Networking 2: Models, Methods and Metal

I.G.Batten@bham.ac.uk

Last Lecture's Problem

- Could people send me a couple of lines with their ideas, so I can use that to kick off the next lecture? Sorry, I should have said: in a room this large, a spin around the room for ideas and my then riffing on them isn't going to work.

Contents

线路交换 vs 包交换

- Circuit Switching v Packet Switching
  - Netheads v Bellheads
- Layered Models

分层模型 传输层与应用层

- Transport and Application Layers

What do we want to do?

我们希望将网络元素(计算机，或运行在计算机上的应用程序)连接在一起，以便它们可以交换数据:

- We want to link network elements (computers, or the applications running on computers) together so they can exchange data:
  - In any chosen amounts
  - At any chosen speed
  - With zero error rates

以任何可选的量，任何可选的速度，以及令错误率传输

- With that, there is no need to make engineering trade-offs, and we can all go and do something more interesting instead.

有了这些，就没有必要在工程上(网络上的问题)进行权衡，我们都可以去做一些更有趣的事情。

现实的一记耳光

Reality Called

正如hifi高玩所幻想的直线才是最牛的 但他们又不能有它们一样，我们也必须接受这样一个事实，
即数据在容量有限的网络中以有限制的速度传播，错误发生的数量不会为零。

- Just as hi-fi nerds fantasise about amplifiers that are “straight wires with gain”, but accept they can’t have them, we have to accept that data travels at a bounded speeds through networks of bounded capacity, and non-zero numbers of errors occur.

我们可以从以下方案来描述数据流：

- We can characterise a flow of data in terms of:

  单位时间内的数据量，或者说带宽(不准确但大家都这样说)，通常以每秒位为单位

  - **volume of data per unit time**, inaccurately but very commonly **bandwidth**, usually in bits per second
  - **latency** (seconds, or more usually) 延迟（秒）
  - **error rate** (errors per bit, or bits per one error)

    错误率（每比特多少错误，或多少比特一个错误）
- With suitably large error bars on these numbers, and suitable prefixes to make the numbers usable.

Volume

- Often **bandwidth, capacity**, etc
- Unhelpfully, often “speed”, which is like asking which is faster, a car travelling at 60mph or a lorry travelling at 60mph.
- How many bits per second can I put in one end and have emerge from the other without loss?
- Ranges from a few hundred bits per second – communication with a Voyager probe – to many terabits per second– long-haul optical backbone transmission.
- Commodity servers have gone from 10Mbps to 40Gbps in thirty years, 4000x increase.

延迟

Latency

- Also **delay** 具体的一个字节穿过网络所需的时间
- How long does it take for a particular bit, or group of bits 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 floor, a lower bound we have to worry about.
  - This has not got much better over time, because the speed of light dominates over everything else over long distances.

Error Rate

- Often **Bit Error Rate, BER**.
- Will be a property of the medium, technology and potential interference
  - Data loss is an error 需要注意数据丢失也算是一种错误
- Cosmic Ray, mains impulse noise, cable and connector issues, etc, etc.
- Determines how much error checking and correction we need, and ultimately there will be an uncorrected error rate we have to accept.
- In 2018, CPU power is cheap, so there is rarely a significant cost to adding more, longer, better error detection and correction.
  - Pro tip: if you are moving files you care about, why not send a SHA512 hash? And a SHA3 hash as well?
  - For email, the failure of the Content-MD5 header to get traction was a huge shame.

Error Rate

顺便说一下:注意在实际设备中，加工过程中产生的误差有很大的影响。

- In passing: note that in real equipment, errors induced in processing have a significant effect.
- BER in non-ECC RAM is of the order of 1 bit per gigabit per day, more as the memory ages.
- Effect of this on (say) TCP is an interesting project, and depend on the nature of the errors, but estimates are around one undetected error in small numbers of billions of packets.

  这对(比方说)TCP的影响是一个有趣的项目，并且取决于错误的性质，但是估计大约是一个未被检测到的错误，它包含数十亿个数据包。

  - This is not reassuring. PhD there for the taking.
- Off topic from off topic: my prediction is that next year’s big problem is going to be memory errors in CPU caches.

大致给你们一点概念

Rough Numbers

今天的实际网络的容量是千兆位或兆比特每秒，延迟是毫秒，未检测到的错误率是一兆字节，实际错误率在检测前是非常可变的。

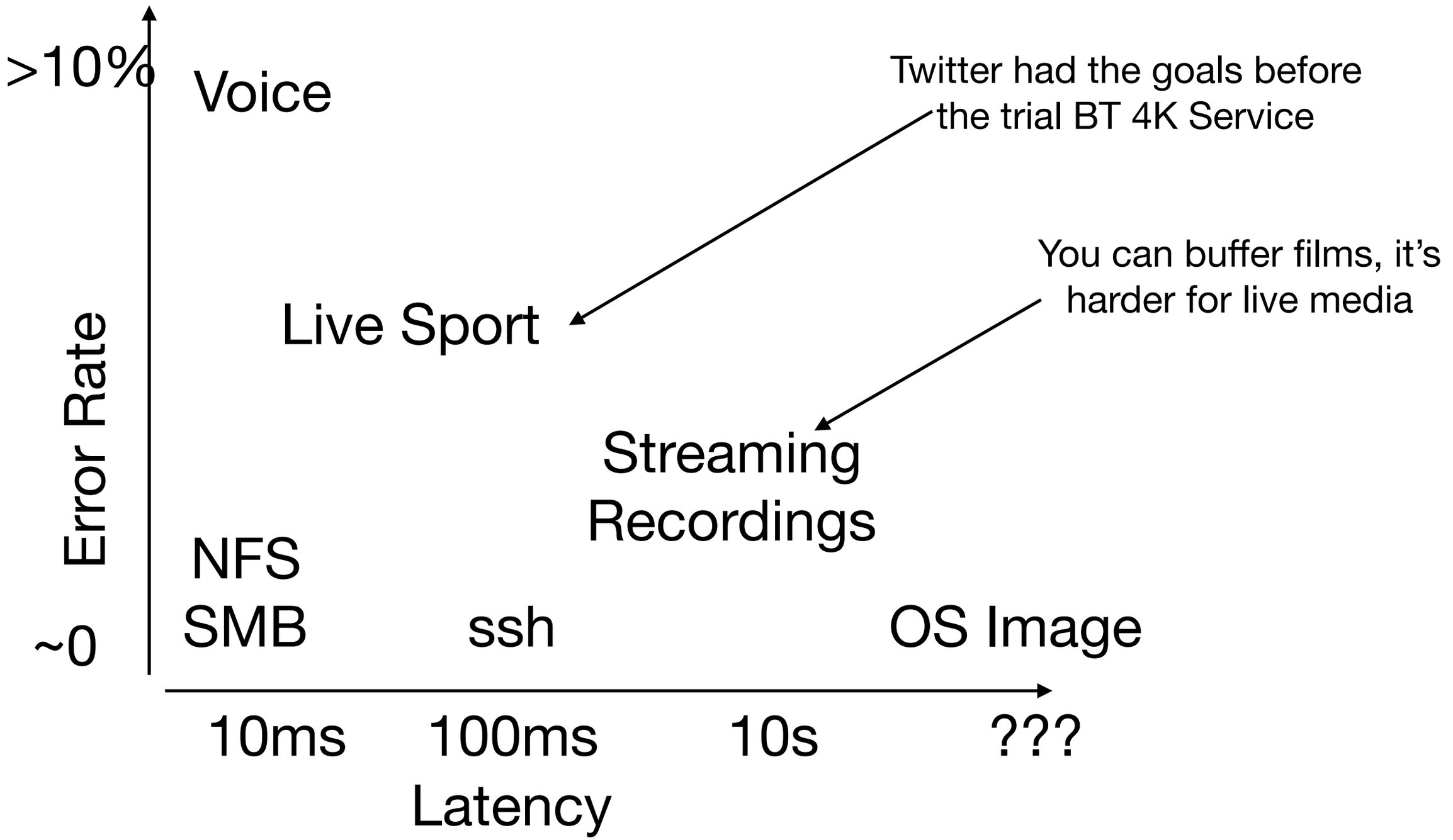
- Practical networks today are volume in **gigabits or terabits per second**, latency in **milliseconds**, undetected error rates of **1 bit in terabytes**, actual error rates before detection very variable.
- Note latency has not improved much in the past thirty years, when bandwidth has increased by between three and five orders of magnitude (10^3–10^5). **Dealing with latency is a big problem today**.

在过去三十年延迟都没什么进步，然而网速却涨了千倍万倍

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

超时

Over time

连接的带宽可以说是无限大的，但依旧受限于成本

- Link bandwidth tends to infinity (terabits per second is affordable with DWDM), but is still limited by cost

延迟既受处理时间的影响(处理时间会减少)，也受始终存在的光速的影响

- 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? $c = 3 \times 10^8\ ms^{-1}$.

- Raw error rates remain about the same, but we have more processing power and spare bandwidth so can do better error correction

原始错误率保持不变，但我们有更多的处理能力和空闲带宽，因此可以做更好的错误更正

- So let’s look at some hardware.

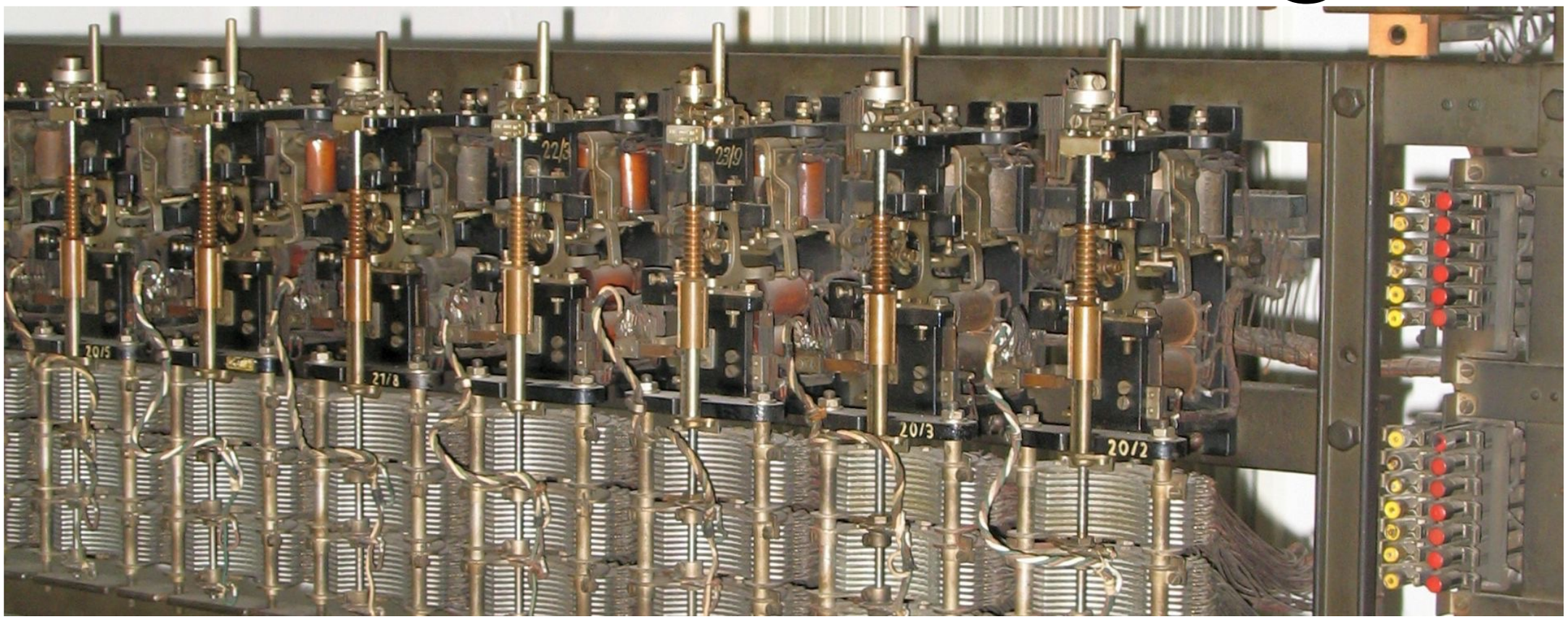
“Never underestimate the bandwidth of a jumbo full of tape”

- 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, or 1TB SD cards, 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

老式电话:物理地将电线连接在一起，使一端的麦克风通过合适的安培和多路复用器连续地与另一端的扬声器相连

Problems

问题：非常低效，即使在一方或双方都处于静止状态时，也会占用双向的电路(duplex circuit)

- Very inefficient: ties up a duplex circuit even when one or both parties are quiescent

在50年代 这样的串线方法是非常复杂和昂贵的

- 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

包交换

Packet Switching

由美国的Paul Baran (RAND)和英国的Donald Davies (NPL)分别提出。

- Proposed independently by Paul Baran in the USA (RAND) and Donald Davies in the UK (NPL).

Davies曾在伯明翰钢管合金公司担任Klaus Fuch的助理

- Davies had worked as an assistant to Klaus Fuchs at Birmingham on Tube Alloys

Baran曾在研究核后通信的可生存性架构

- Baran was working on survivable architectures for post-nuclear communications

Packet Switching

将数据流分成小单元，称为“包” 并为其赋予一些验证信息

- 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 在网络上交换数据包，直到它们到达目的地
  - “Switch” is American railroad usage for what are “points” or “turnouts” in UK railway usage.

Advantages

你是在时间维度上进行多路传输，而不是在频率维度上。这样建造起来更为方便

- You are multiplexing in the **time** domain, rather than multiplexing in the **frequency** domain. It's easier to build, in general.

  你可以将数据包推迟 直到线路上有足够空间传输(也就是后面提及的延迟)

  - 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. 当没有数据传输时就不会消耗资源

- You therefore get **statistical gain** on bandwidth.

  因此在带宽上你就有优势了

- You can re-route data around failed switches, for resilience. 你可以把失败的包交换重新路由，以获得弹性

理解时间域

Time Domain

假设我有些数据流，每个8kbps(8000比特每秒)

- Suppose I have some data streams, each of 8kbps (8000 bits per second).

我有一条一万六比特每秒的网线

- I have a line which runs at 16000 bits per second

我可以先用我的缓存把收到的8k数据先存起来，当缓存满时再以16kbps发送他们，这样输出时有50%时间都是空闲的

- 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).

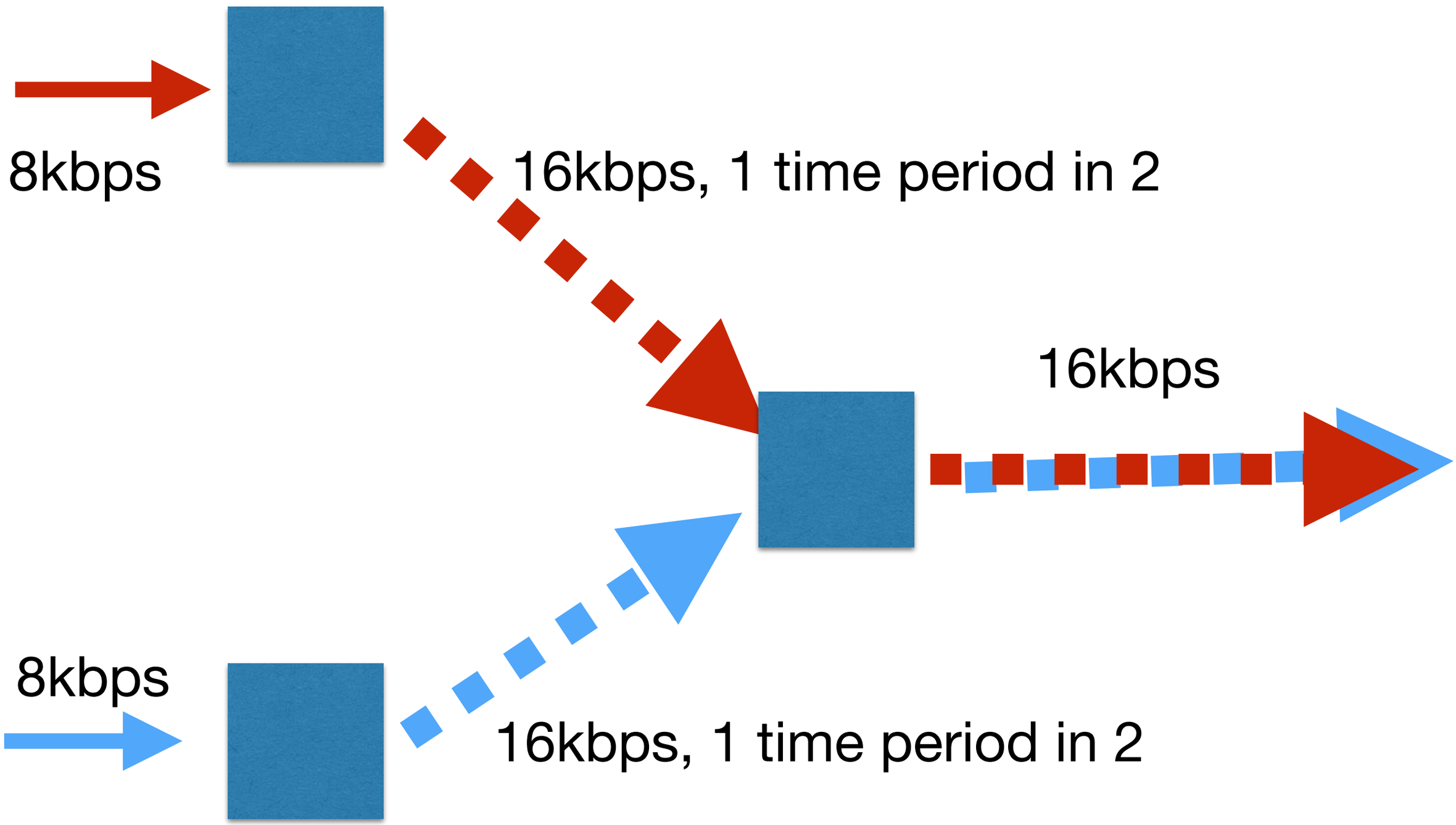
这样我就可以用那50%空闲再做一次这个事

- 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.**

这样会造成延迟：数据块中第一个比特会等到最后一比特被收到时才会发出，然后以更高的速度发送出去

Time Domain


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

Contrast this with Frequency Domain

- I can distinguish two signals on a medium by their frequency. 我能根据频率来分辨同一介质中的俩不同信号。
  - 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. 空气中有高低音调的声音，纤维中有红光和蓝光，真空或电线中有96兆赫和98兆赫的无线电信号。
- I can modulate that signal (or “carrier”) with some data by varying the amplitude (“AM”), the frequency (“FM”), the width of some pulses (“PWM”).

我可以通过改变振幅(“AM”)、频率(“FM”)、一些脉冲宽度(“PWM”)，用一些数据来调制那个信号(或“载波”)。

- On the receiving side I separate the two carriers, and then decode the modulation.

在接收端，我将两个信号(运营商)分离，然后解码调制。

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

- Frequency Domain Multiplexing doesn't introduce systematic latency. 频域复用不会带来系统延迟。
  - Which is why, amongst other reasons, musicians' monitoring systems still use analogue transmission. 这就是为什么，除了其他原因，音乐玩家的监测系统仍然使用模拟传输。
- But is harder to engineer robustly and for real signals the use of available capacity is usually less.
但要稳健地进行设计却比较困难，而且对于实际信号来说，可用容量的使用通常较少。
- All the networking we'll be talking about, apart from at the transmission level, is time-domain.
我们将要讨论的所有网络，除了传输层，都是时域的。

Statistical Gain

- Not many people hammer full line rate 24x7: people pause to sleep, catch breath, etc. 没有多少人能24x7的用满速度:人们停下来睡觉，喘口气，等等。
- Even hammering at full line rate is limited by the other end and other network delays. 即使是全速运作也受到另一端和其他网络延迟的限制
- So you can sell 100 people 10Mbps each, and only provision (100-x) % of (100 x 10M), x in [0, 100).
  所以你可以卖给100个人每人10Mbps，并且只提供(100×10M)的(100×100)%，x在[0,100]中。
- 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).
  通常，对于住宅，x可能高达95(即，您只提供总带宽的5%)，并且一直高达98(2%，1:50)。
- Also called “contention ratio”, “overbooking”, “over commitment”.
  - And “lying” and “deceitful” by the advertising regulator.
  - Another chance to say “and how are things in Selly Oak?”

Problems

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

  您(通常)必须将大于一个包的数据分解成多个包，并在它们到达时将它们粘在一起。
- Dealing with missing packets (usually) means waiting until you have a completed sequence before you can use the data.

  处理丢失的包(通常)意味着要等到有了完整的序列之后才能使用数据。

Presentation

运营商网络，即你购买用来传输数据的服务，可以提供各种各样的服务。

- 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**. 让我们尽量地简化，只看虚拟电路和数据报。
- 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.

  有趣的是，几十年来，这种以数据报命名的电报还没有商业化。我从来没有见过，也没有寄过，我想我的父母(1935年出生)也没有。

Virtual Circuits

端点让网络建立连接

- 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 网络建立一个通往目的地的通道，并且回馈一些token用以验证
- For the duration, that token identifies a “virtual circuit” linking two endpoints
  在此期间，该token标识连接两个端点的“虚拟电路”
- It is then torn down when the endpoint has finished with it 当端点结束时，它将被拆下
- Network has to know about every connection in progress
  网络必须了解每一个正在进行中的连接
- Each packet in a connection follows the same route 连接中的每个包都遵循相同的路由
- Network tries to sort out ordering and packet loss/duplication, but doesn’t always guarantee it. But it will usually tell you when things have gone wrong. 网络试图解决包丢失/dup 以及排序的问题，但并不总是保证如此。但它通常会告诉你什么时候出了问题。
- The user of the circuit doesn’t have to worry about the fine details, but a checksum would be a good idea once in a while.
  电路的使用者不必担心细节问题，但每隔一段时间进行一次校验和checksum是个比较好的解决方式。

Datagram Services

- Each packet contains complete addressing information 每个包都包含完整的地址信息
- Each packet is considered as a separate item by the network (conceptually, at least: more later)
每个包被网络认为是一个单独的项目(至少在概念上)
- Endpoints are responsible for dealing with issues of loss, duplication, corruption: your packet might get delivered at some point, that is all you know.
端点负责处理丢失、复制和损坏的问题:包可能在某个时候被交付，这就是您需要知道的全部。
- Network doesn’t (need to) know about connections: it just routes packets
网络不需要知道连接，它只用路由包就行了

Netheads v Bellheads

给电信工作的人。如虚拟电路：他们可以塑造和梳理流量，提供附加值，运行复杂且有趣的协议

- “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…

参加IETF的人，喜欢数据报服务：它阻止电信公司去塑造，修饰流量

- Last thirty years are a history of conflict between the two camps.

过去三十年是这两个阵营冲突的历史。

Layering

我刚才描述的是网络提供的最基本的服务:虚拟电路和数据报。

- What I’ve just described is the very basic services made available by networks: virtual circuits and datagrams.

但是应用程序通常需要一些有保证的东西:您发送一个文件，它会完好无损地到达那里，或者您知道出了问题。

- 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.

您希望您的程序能够在不同类型的网络上运行，而不需要进行根本的更改。

Layering

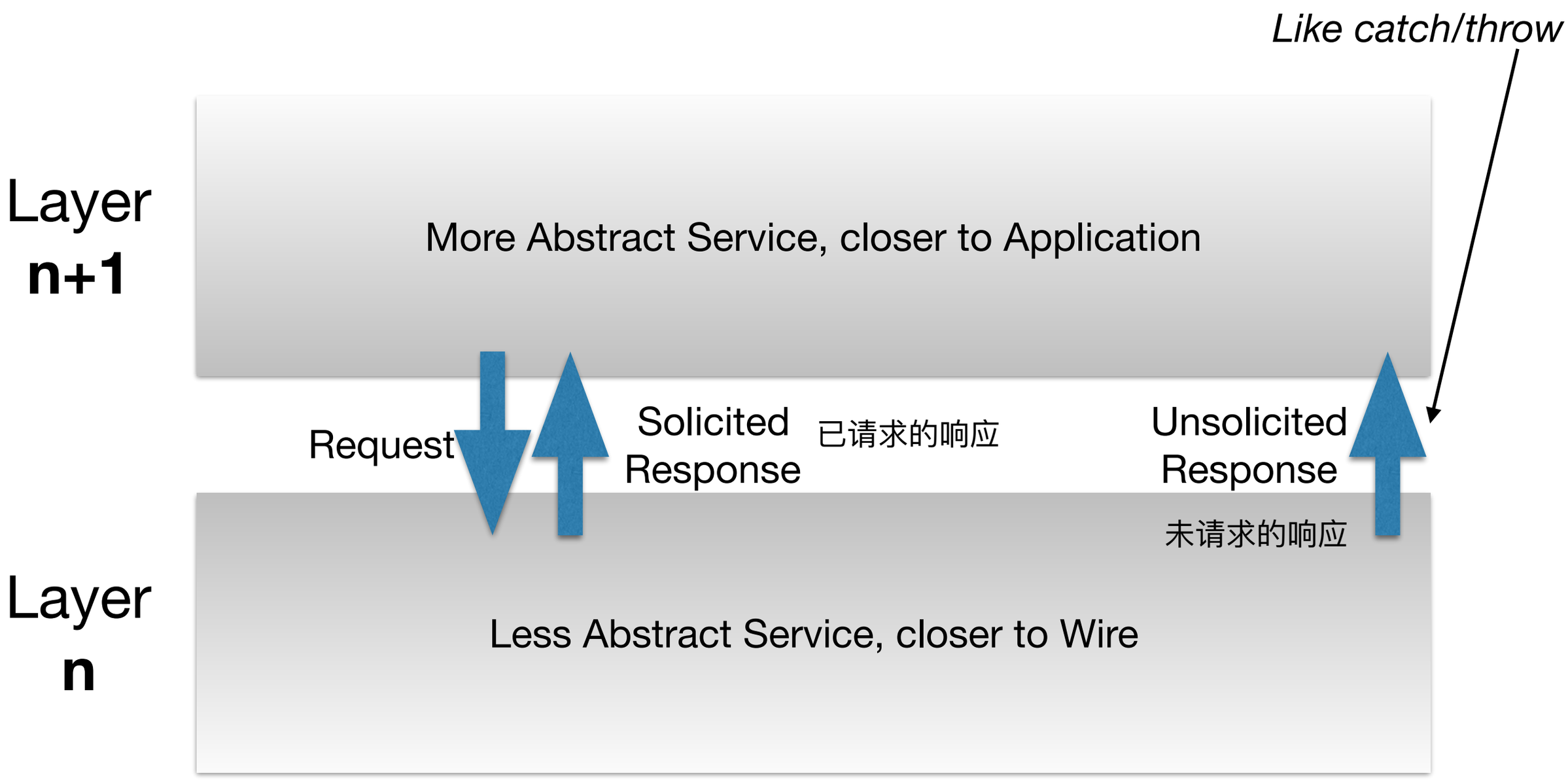
因此，从“层”或“堆栈”的角度来考虑网络的想法就产生了:一系列接口从实际应用程序所需的服务开始，逐渐接近伏特和闪烁的激光二极管。

- 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.

每一层都向它上面的层提供服务，并利用它下面的层为本层提供服务。每个任务只出现在一个层中。

Layers



Anything that obeys the requests can be used as a replacement

Competing Models

“OSI”(开放系统互连)是一个失败的项目，主要是在欧洲，从电信社区内部构建一套标准的网络协议。我们稍后将了解它失败的原因。它与专有系统和……

- “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

这种模式在任何模式取得成功之前很久就出现了

- The model came long before any successful implementation.

  一个愤世嫉俗的历史学家会说，从来没有任何成功的实现。

  - A ~~cynic~~ historian would argue there were never any successful implementations.

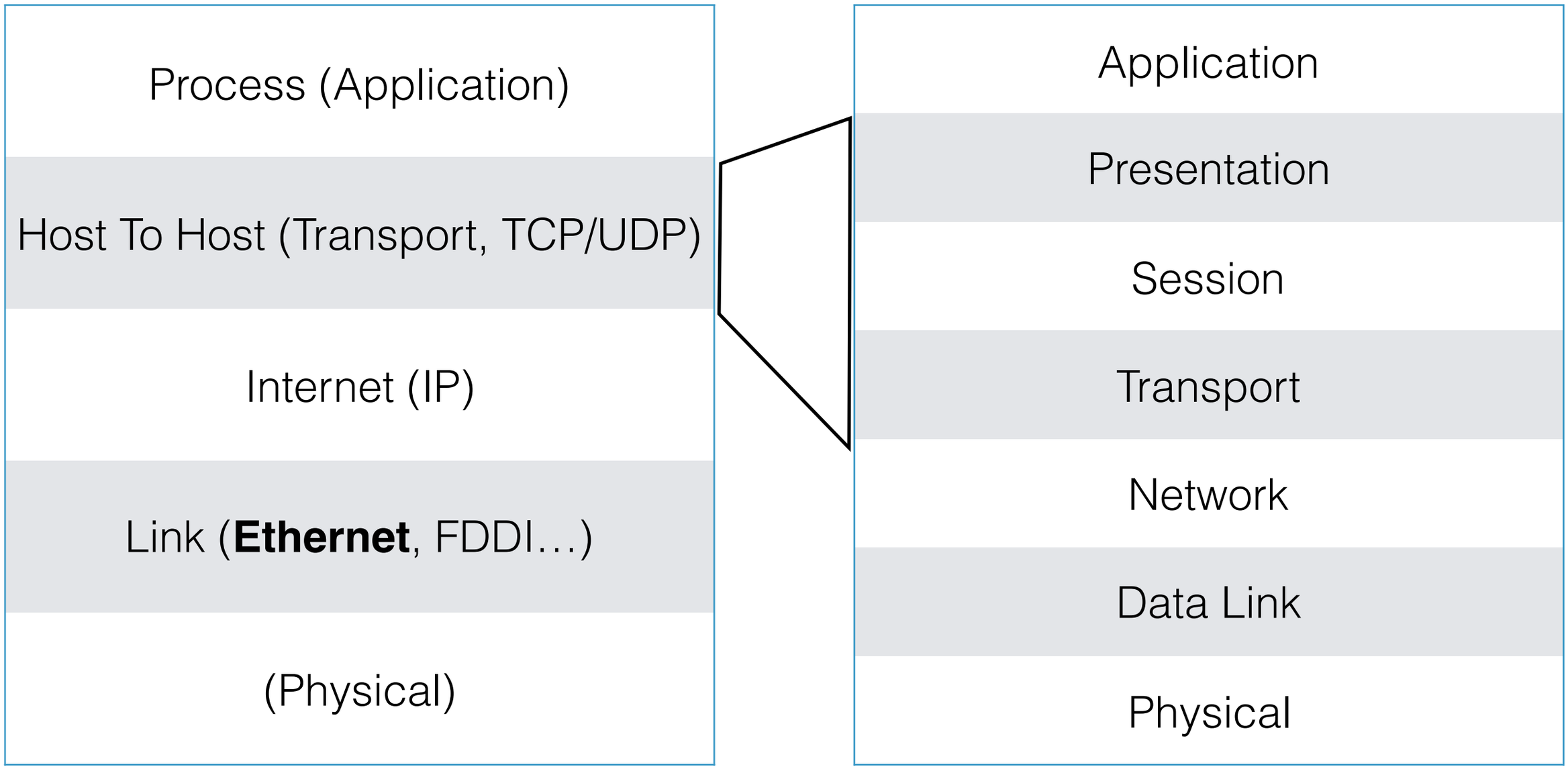
- But the model achieved widespread traction as the model of how computer networks either *should* or *do* work. 但是这个模型作为计算机网络应该或者应该如何工作的模型获得了广泛的关注。

DoD Model

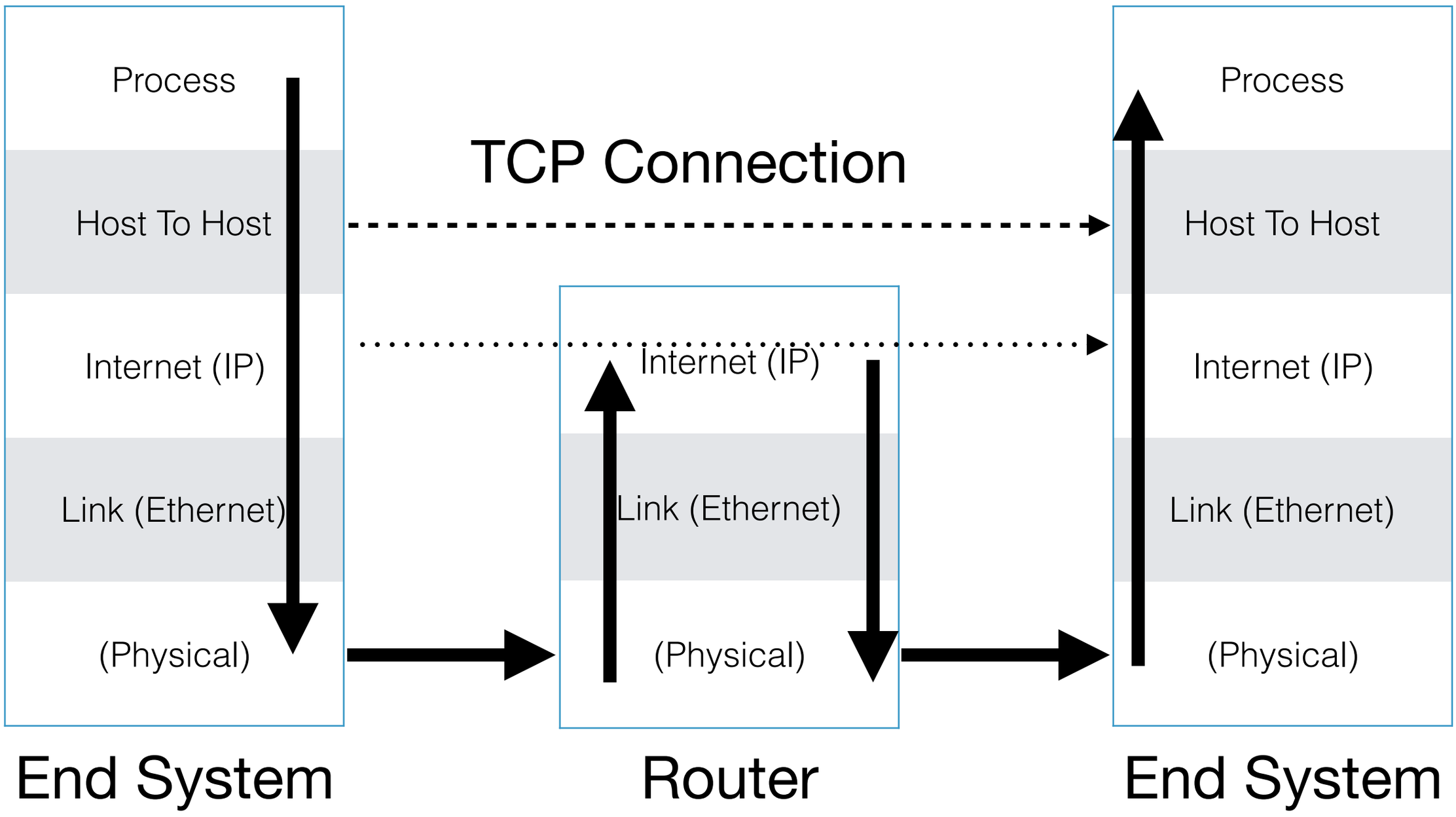
DoD模型是对已建立的实践的事后合理化，部分是为了提供一种方法来比较TCP/IP架构和OSI的提案。

- 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. 实践和实现是第一位的，模型是第二位的。

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

应用程序运行的代码可以做真正有用的工作，以及它们使用的协议。

- 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… 用于邮件的SMTP和IMAP，用于web的HTTP，用于远程登录的ssh……
- They need services to move data from computer to computer. 这些都需要将数据从一台计算机转移到另一台计算机的服务。

The DoD Model: Transport

传输层通过网络在两个终端系统之间移动数据，该网络的拓扑结构和传输层不太了解的其他属性。

- 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

传输层将保证各种特性:可靠性、顺序交付等等(或者将放弃对这些特性的责任)。

- 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.

传输层将允许在同一个终端系统上运行的多个实体(通常是应用程序)之间进行通信。

The DoD Model: Internet / Subnet / Network Layer

这一层在终端系统之间移动包，但不提供任何可靠性保证。它处理选择使用哪个链接来使数据更接近其目的地。

- 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

  这一层也不处理多个应用程序的任何概念:在两个不同的应用程序之间分离数据是上述层的职责

  - IPv4, IPv6

The DoD Model: Link Layer

该层负责从一个网络元素移动数据到下一个元素。这一层涉及到分组、寻址等等。

- 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 使用的协议可能有多种
  - Ethernet is going to be our main focus, but ATM will sometimes still appear 以太网将是我们关注的焦点，但ATM有时仍会出现

(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…

这是用于移动数据的实际编码。以太网可以通过同轴电缆(如果你是老人的话)、双绞线、各种光纤、无线电……

Layers are Encapsulated in Packets

Ethernet Header	IP Header	TCP or UDP Header	**Data**	TCP or UDP Trailer	IP Trailer	Ethernet Trailer

The OSI Model: extras

OSI模型的主要区别在于比DoD多了三层: 传输层 = 表示层、会话层和传输层。

- The OSI model's main difference is that there are three layers where the DoD has one: transport's equivalent is presentation, session, transport.

表示层用于处理数据编码(例如，将整数转换为每个人都可以使用的中立格式)。

- Presentation is intended to handle data encoding (converting integers to a neutral format everyone can use, for example).
- Session is meant to handle the relationship between multiple connections (say, restarting a failed file transfer)

会话用于处理多个连接之间的关系(例如，重新启动失败的文件传输)

- Transport is transport.

传输层则就是传输，Dod把上面所有的都放进了库里

  - 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.

经验和历史表明，额外的层是无用的:应用程序需要的东西过于具体，无法作为通用服务提供。但那是另一堂课。

Abstraction is **hard**

当更接近应用程序的一层被要求提供不适合低层的服务时(比如尝试用Haskell编写操作系统)，问题就出现了。

- 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)

如果你想自己写一个传输数据的服务 像TCP一样可靠，是非常非常困难的

- Conversely, if your network just does virtual circuits, sending a single datagram is very expensive.

TCP/IP Wins…

这就是为什么TCP/IP赢的原因

- Because there is one transport service for connections and one transport service for datagrams

  因为只有一个用于连接的服务和一个用于数据传输服务，其它的就没有去管了（比如另一端的电脑将数据以另一个方式去呈现，那就不管我的事了，你自己去修理）

  - TCP/IP has various experimental transports for special purposes, but they are not widely deployed.

    TCP/IP有各种用于特殊目的的实验性传输，但是它们没有得到广泛的部署。

- 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…

另一个原因是其只有一个网络层

一个杀死osi的原因是在商业上愚蠢地去完美适配全世界所有电信商的设备
与其只有一个网络层IP来说，osi想要不同网络层在顶端传输不同服务，导致其过于复杂

- 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…

OSI试图要“高效”，并尝试提供多种传输服务，从非常细微的服务到虚拟电路的传输，再到非常复杂的datagram服务

- 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

大型电信公司BT之类 它们牵光纤 然后（通常）出售一个服务级协议

- 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)	Consumer ISPs
Sale of Internet Connections to consumers and businesses, with large amounts of statistical gain and vague SLAs	
Carrying the data belonging to ISPs over a distance: they want an SLA, and only rent bandwidth they need <u>after</u> 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

那设备又是如何被构造的呢？它们是被分层的

其实现在所有的设备都是一个电脑了，你不论搞台什么电脑都可以当做路由器用

- In two different ways
  - Different pieces of equipment do different jobs in the stack (although the distinctions are becoming blurred)

    不同的设备来做不同的工作（尽管它们的区别正在变模糊）
  - There are aspects of different layers in the same piece of equipment, and separating those makes design, construction and security easier.

    同一件设备中有不同的层，将这些层分开可以使设计、施工和安全更容易。

Planes within Elements

你可以把设备视为其具有一个管理层，一个控制层 和一个交通（数据）层

- 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

这些中间还会穿插一些安全层

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.

一个包含界面的管理层

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.

一个设置路线的控制层

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")

数据层：传送数据，与其它组件完全独立 (completely isolated)

Different Hardware

我们可以根据其模型来将网络硬件分开来　　这些硬件作用于不同的层

- We can divide network hardware up by which part of the model they implement.

交换机：连接层，在连接中交换以太网packets

- “**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

现在这些东西的很模糊了，尤其是交换机，几乎被包含在了路由器中

- 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.