Computer Networking

Slide Set 1

Andrew W. Moore
Andrew.Moore@cl.cam.ac.uk

Topic 1 Foundation

- Administrivia
- Networks
- Channels
- Multiplexing
- Performance: loss, delay, throughput

Course Administration

Commonly Available Texts

- ❑ Computer Networks: A Systems Approach
  Peterson and Davie
  https://book.systemsapproach.org
  https://github.com/SystemsApproach/book

- ❑ Computer Networking : Principles, Protocols and Practice
  Olivier Bonaventure (and friends)
  Less GitHub but more practical exercises
  https://www.computer-networking.info/
  Version 3 draft (UCAM access only)

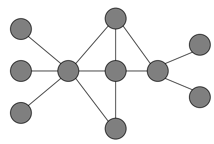
Other textbooks are available.

Thanks

- Slides are a fusion of material from
  to Stephen Strowes, Tilman Wolf & Mike Zink, Ashish Padalkar , Evangelia Kalyvianaki, Brad Smith, Ian Leslie, Richard Black, Jim Kurose, Keith Ross, Larry Peterson, Bruce Davie, Jen Rexford, Ion Stoica, Vern Paxson, Scott Shenker, Frank Kelly, Stefan Savage, Jon Crowcroft , Mark Handley, Sylvia Ratnasamy, Adam Greenhalgh, and Anastasia Courtney.
- Supervision material is drawn from
  Stephen Kell, Andy Rice, and the TA teams of 144 and 168
- Finally thanks to the fantastic past Part 1b students and Andrew Rice for all the tremendous feedback.

What is a network?

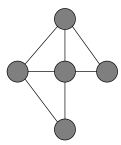
- A system of “links” that interconnect “nodes” in order to move “information” between nodes



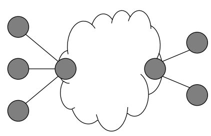
- Yes, this is all rather abstract

What is a network?

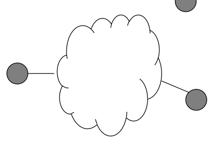
- We also talk about



or



or even



- Yes, abstract, vague, and under-defined….

There are *many* different types of networks

- Internet
- Telephone network
- Transportation networks
- Cellular networks
- Supervisory control and data acquisition networks
- Optical networks
- Sensor networks

We will focus almost exclusively on the Internet

The Internet has transformed everything

- The way we do business
  - E-commerce, advertising, cloud-computing
- The way we have relationships
  - Facebook friends, E-mail, IM, virtual worlds
- The way we learn
  - Wikipedia, search engines
- The way we govern and view law
  - E-voting, censorship, copyright, cyber-attacks

The Internet transforms everything


Taking the dissemination of information to the next level

The Internet is big business

- Many large and influential networking companies
  - Huawei, Broadcom, AT&T, Verizon, Akamai, Cisco, …
  - $132B+ industry (carrier and enterprise alone)
- Networking central to most technology companies
  - Apple, Google, Facebook, Intel, Amazon, VMware, …

But why is the Internet *interesting*?

"What's your formal model for the Internet?" -- *theorists*

"Aren't you just writing software for networks" – *hackers*

"You don't have performance benchmarks???" – *hardware folks*

"Isn't it just another network?" – *old timers at BT*

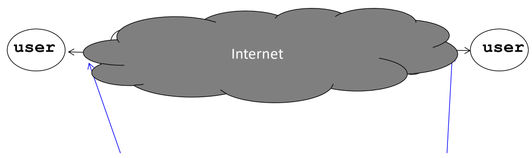
"What's with all these TLA protocols?" – *all*

"But the Internet seems to be working…" – *my mother*

A few defining characteristics of the Internet

A federated system

- The Internet ties together different networks
  - >20,000 ISP networks (the definition is fuzzy)



Tied together by IP -- the "Internet Protocol" : a single common interface between users and the network and between networks

A federated system

- The Internet ties together different networks
  - >20,000 ISP networks

- A single, common interface is great for interoperability…
- …but tricky for business

- Why does this matter?
  - ease of interoperability is the Internet's most important goal
  - practical realities of incentives, economics and real-world trust, drive topology, route selection and service evolution

Tremendous scale (2020 numbers – so some 'weird')

- 4.57 Billion users (58% of world population)
- 1.8 Billion web sites
  - 34.5% of which are powered by the WordPress!
- 4.88 Billion smartphones (45.4% of population)
- 500 Million Tweets a day
- 100 Billion WhatsApp messages per day
- 1 Billion hours of YouTube video watched per day
- 500 hours of Youtube video added per minute
- 2+ billion TikTok installs
- 60% video streaming
  - 12.5% of the Internet traffic is native Netflix

Tremendous scale (2020 numbers – so some 'weird')

- 4.57 Billion users (58% of world populatio
- 1.8 Billion web sites
  - 34.5% of which are power ess!
- 4.88 Billion smar % of population)
- 500 Mil ay
- natsApp messages per day
- illion hours of YouTube video watched per day
- 500 hours of Youtube video added per minute
- 2+ billion TikTok installs
- 60% video streaming
  - 12.5% of the Internet traffic is native Netflix

"Internet Scale" refers to such systems

Enormous diversity and dynamic range

- Communication latency: microseconds to seconds (10^6)
- Bandwidth: 1Kbits/second to 400 Gigabits/second (10^7)
- Packet loss: 0 – 90%

- Technology: optical, wireless, satellite, copper

- Endpoint devices: from sensors and cell phones to datacenters and supercomputers
- Applications: social networking, file transfer, skype, live TV, gaming, remote medicine, backup, IM
- Users: the governing, governed, operators, malicious, naïve, savvy, embarrassed, paranoid, addicted, cheap …

Constant Evolution

1970s:
- 56kilobits/second "backbone" links
- <100 computers, a handful of sites in the US (and one UK)
- Telnet and file transfer are the "killer" applications

Today
- 400+Gigabits/second backbone links
- 40B+ devices, all over the globe
  - 27B+ IoT devices alone

Asynchronous Operation

- Fundamental constraint: **speed of light**

- Consider:
  - How many cycles does your 3GHz CPU in Cambridge execute before it can possibly get a response from a message it sends to a server in Palo Alto?
    - Cambridge to Palo Alto: 8,609 km
    - Traveling at 300,000 km/s: 28.70 milliseconds
    - Then back to Cambridge: 2 x 28.70 = 57.39 milliseconds
    - 3,000,000,000 cycles/sec * 0.05739 = 172,179,999 cycles!

- Thus, communication feedback is always *dated*

Prone to Failure

- To send a message, **all** components along a path must function correctly
  - software, wireless access point, firewall, links, network interface cards, switches,…
  - Including human operators
- Consider: 50 components, that work correctly 99% of time → 39.5% chance communication will fail
- Plus, recall
  - scale → lots of components
  - asynchrony → takes a long time to hear (bad) news
  - federation (**inter**net) → hard to identify fault or assign blame

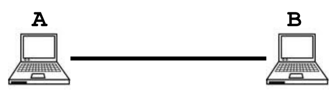
Recap: The Internet is…

- A complex federation
- Of enormous scale
- Dynamic range
- Diversity
- Constantly evolving
- Asynchronous in operation
- Failure prone
- Constrained by what's practical to engineer
- Too complex for (simple) theoretical models
- "Working code" doesn't mean much
- Performance benchmarks are too narrow

An Engineered System

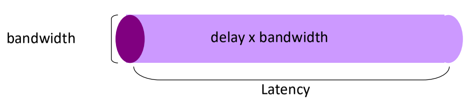
- Constrained by what technology is practical
  - Link bandwidths
  - Switch port counts
  - Bit error rates
  - Cost
  - …

Nodes and Links



Channels = Links
Peer entities = Nodes

Properties of Links (Channels)

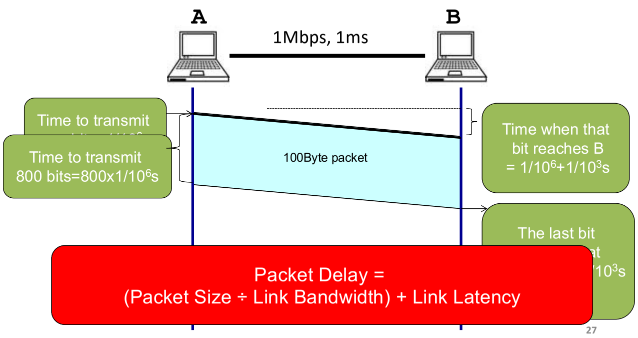


- Bandwidth (capacity): "width" of the links
  - number of bits sent (or received) per unit time (bits/sec or bps)
- Latency (delay): "length" of the link
  - propagation time for data to travel along the link (seconds)
- Bandwidth-Delay Product (BDP): "volume" of the link
  - amount of data that can be "in flight" at any time
  - propagation delay × bits/time = total bits in link

Examples of Bandwidth-Delay

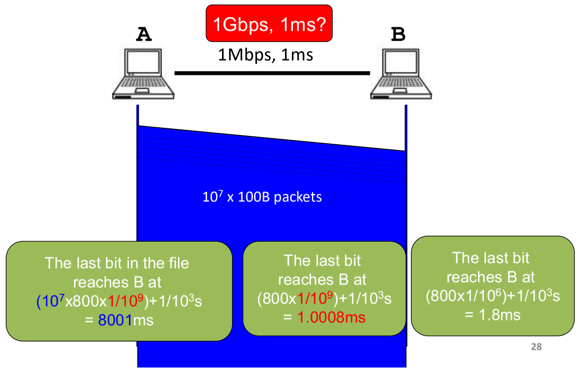
- Same city over a slow link:
  - BW~100Mbps
  - Latency~10msec
  - BDP ~ 10^6bits ~ 125KBytes
- Intra Datacenter:
  - BW~100Gbps
  - Latency~30usec
  - BDP ~ 10^6bits ~ 375KBytes
- Cross-Atlantic over fast link:
  - BW~10Gbps
  - Latency~100msec
  - BDP ~ 10^9bits ~ 125MBytes
- Intra Host:
  - BW~100Gbps
  - Latency~16nsec
  - BDP ~ 1600bits ~ 200Bytes

Packet Delay
Sending a 100B packet from A to B?

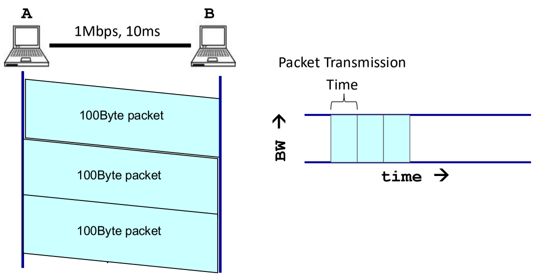


Packet Delay =
(Packet Size ÷ Link Bandwidth) + Link Latency

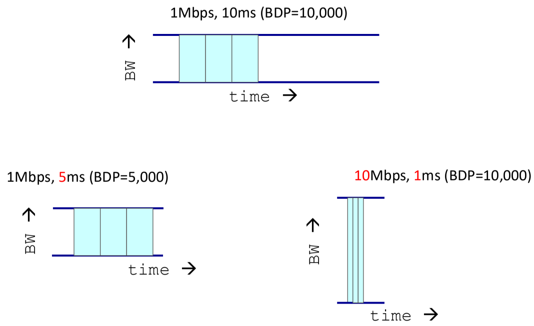
1GB file in 100B packets ay
Sending a 100B packet from A to B?



Packet Delay: The "pipe" view
Sending 100B packets from A to B?

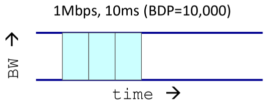


Packet Delay: The "pipe" view
Sending 100B packets from A to B?



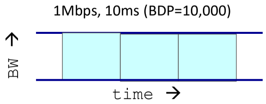
Packet Delay: The "pipe" view
Sending 100B packets from A to B?

1Mbps, 10ms (BDP=10,000)



What if we used *200Byte packets??*

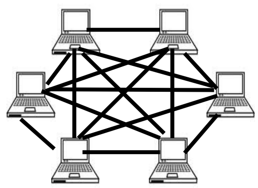
1Mbps, 10ms (BDP=10,000)



Recall Nodes and Links

What if we have more nodes?

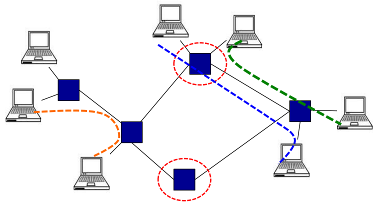
One link for every node?



Need a <u>scalable</u> way to interconnect nodes

Solution: A switched network

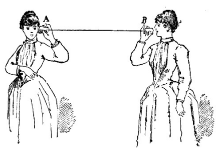
Nodes <u>share</u> network link resources



How is this sharing implemented?

Two forms of switched networks

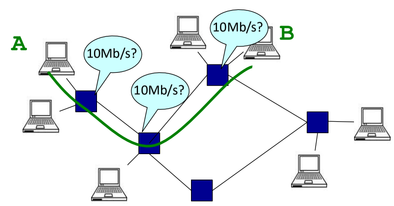
- Circuit switching (used in the *POTS*: Plain Old Telephone system)



- Packet switching (used in the Internet)

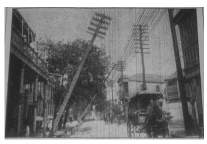
Circuit switching

Idea: source reserves network capacity along a path



(1) Node A sends a reservation request
(2) Interior switches establish a connection -- i.e., "circuit"
(3) A starts sending data
(4) A sends a "teardown circuit" message

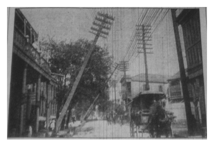
Multiplexing



Sharing makes things efficient (cost less)

- One airplane/train for 100's of people
- One telephone for many calls
- One lecture theatre for many classes
- One computer for many tasks
- One network for many computers
- One datacenter many applications

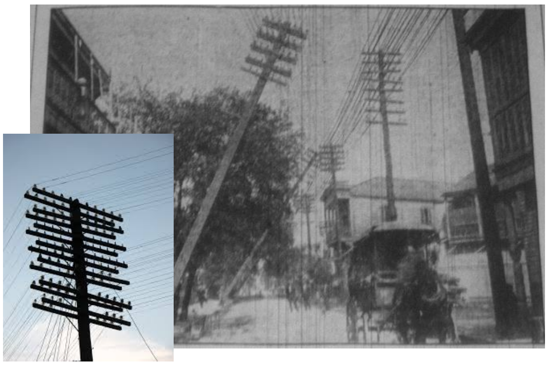
Multiplexing



Sharing makes things efficient (cost less)

- One airplane/train for 100's of people
- One telephone for many calls
- One ~~lecture theatre~~ Lecturer? for many classes
- One computer for many tasks
- One network for many computers
- One datacenter many applications

Old Time Multiplexing

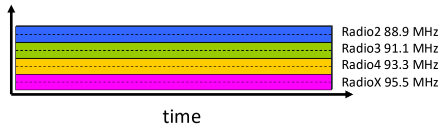


Circuit Switching: FDM and TDM

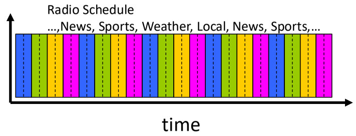
Example:

4 users

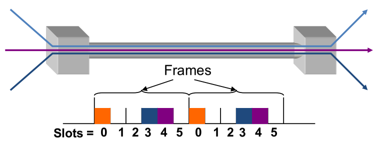
Frequency Division Multiplexing



Time Division Multiplexing

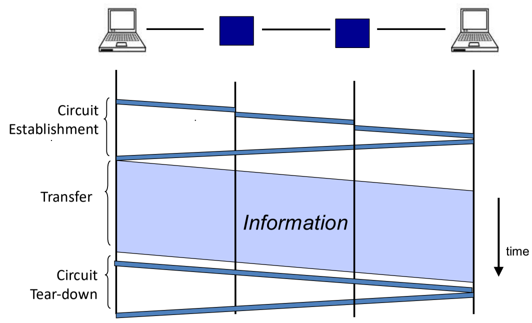


Time-Division Multiplexing/Demultiplexing



- Time divided into frames; frames into slots
- Relative slot position inside a frame determines to which conversation data belongs
  - e.g., slot 0 belongs to orange conversation
- Slots are reserved (released) during circuit setup (teardown)
- If a conversation does not use its circuit **capacity is lost!**

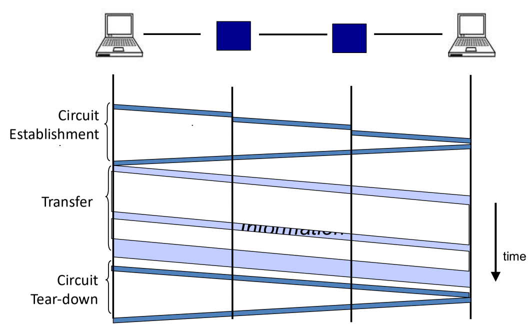
Timing in Circuit Switching



Circuit switching: pros and cons

- Pros
  - guaranteed performance
  - fast transfer (once circuit is established)
- Cons

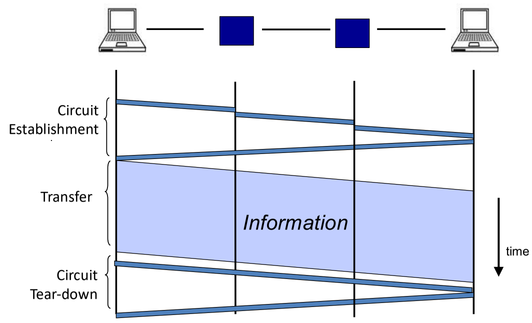
Timing in Circuit Switching



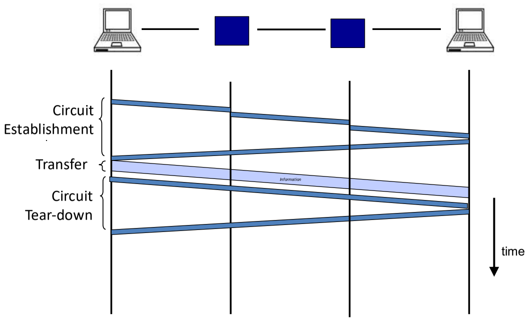
Circuit switching: pros and cons

- Pros
  - guaranteed performance
  - fast transfer (once circuit is established)
- Cons
  - **wastes bandwidth if traffic is "bursty"**

Timing in Circuit Switching



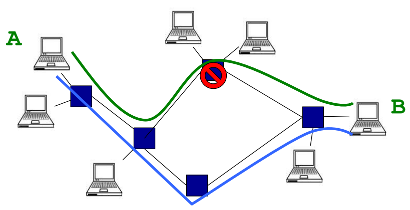
Timing in Circuit Switching



Circuit switching: pros and cons

- Pros
  - guaranteed performance
  - fast transfers (once circuit is established)
- Cons
  - wastes bandwidth if traffic is "bursty"
  - **connection setup time is overhead**

Circuit switching



Circuit switching doesn't "route around failure"

Circuit switching: pros and cons

- Pros
  - guaranteed performance
  - fast transfers (once circuit is established)
- Cons
  - wastes bandwidth if traffic is "bursty"
  - connection setup time is overhead
  - **recovery from failure is slow**

Numerical example

- How long does it take to send a file of 640,000 bits from host A to host B over a circuit-switched network?
  - All links are 1.536 Mbps
  - Each link uses TDM with 24 slots/sec
  - 500 msec to establish end-to-end circuit

Let's work it out!

51

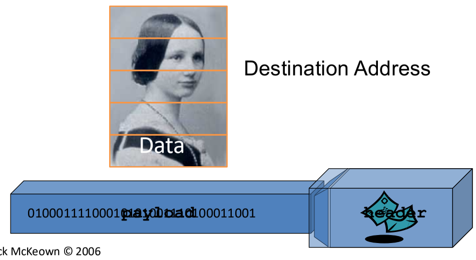
Two forms of switched networks

- Circuit switching (e.g., telephone network)
- Packet switching (e.g., Internet)

52

Packet Switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a "header" and "payload"*



After Nick McKeown © 2006

54

Packet Switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a "header" and "payload"*
  - payload is the data being carried
  - header holds instructions to the network for how to handle packet (think of the header as an API)
  - In this example, the header has a destination address
  - More complex headers may include
    - How this traffic should be handled? (first class, second class, etc)
    - Who signed for it?
    - Were the contents ok?

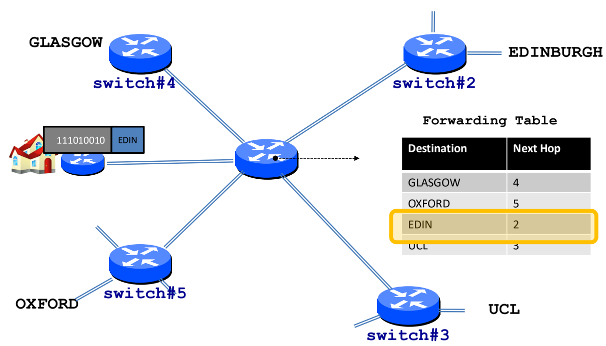
55

Packet Switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a "header" and "payload"
- Switches "forward" packets based on their headers

56

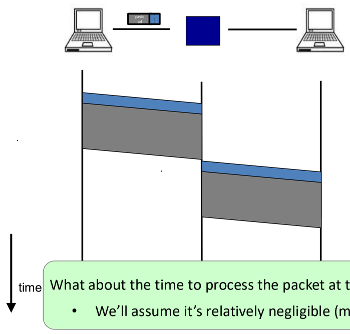
Switches forward packets



Destination	Next Hop
GLASGOW	4
OXFORD	5
EDIN	2
UCL	3

57

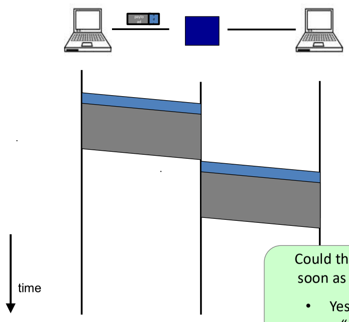
Timing in Packet Switching



What about the time to process the packet at the switch?

- We'll assume it's relatively negligible (mostly true)

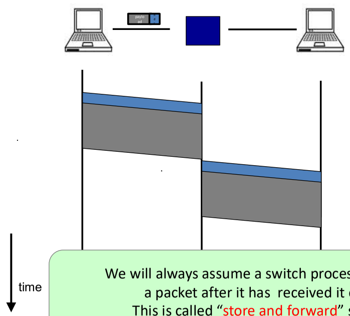
Timing in Packet Switching



Could the switch start transmitting as soon as it has processed the header?

- Yes! This would be called a "cut through" switch

Timing in Packet Switching



We will always assume a switch processes/forwards a packet after it has received it entirely. This is called "store and forward" switching

Packet Switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a "header" and "payload"
- Switches "forward" packets based on their headers

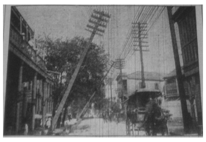
Packet Switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a "header" and "payload"
- Switches "forward" packets based on their headers
- Each packet travels independently
  - no notion of packets belonging to a "circuit"

Packet Switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a "header" and "payload"
- Switches "forward" packets based on their headers
- Each packet travels independently
- No link resources are reserved in advance. Instead packet switching leverages statistical multiplexing (stat muxing)

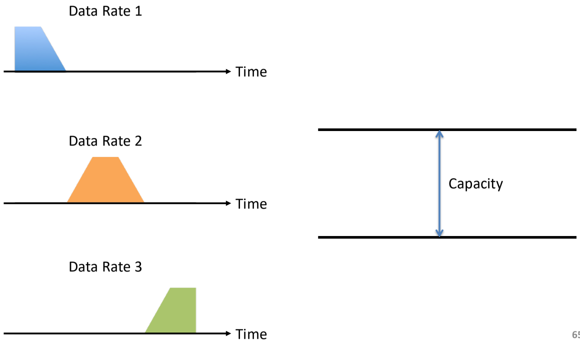
Multiplexing



Sharing makes things efficient (cost less)

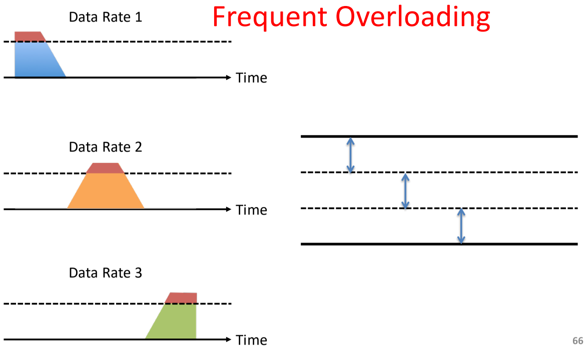
- One airplane/train for 100's of people
- One telephone for many calls
- One lecture theatre for many classes
- One computer for many tasks
- One network for many computers
- One datacenter many applications

Three Flows with Bursty Traffic



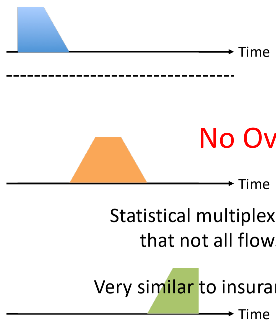
When Each Flow Gets 1/3rd of Capacity

Frequent Overloading



When Flows Share Total Capacity

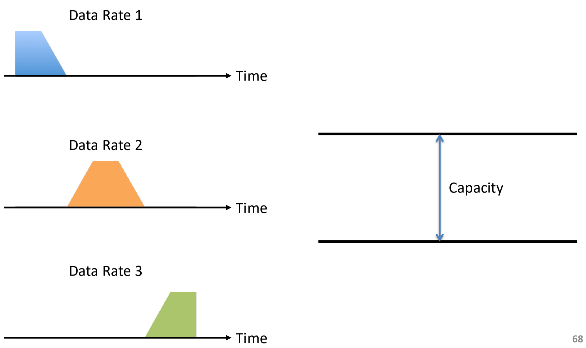
No Overloading



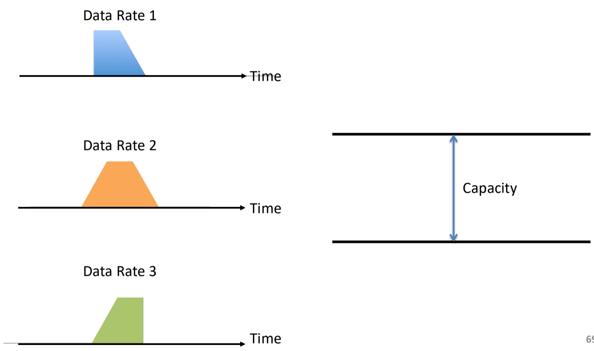
Statistical multiplexing relies on the assumption that not all flows burst at the same time.

Very similar to insurance, and has same failure case

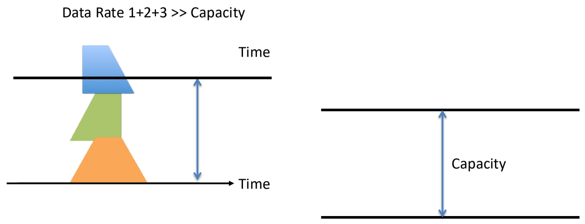
Three Flows with Bursty Traffic



Three Flows with Bursty Traffic



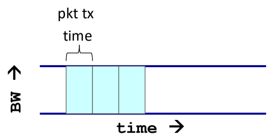
Three Flows with Bursty Traffic



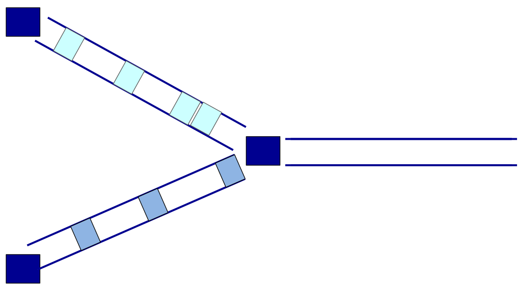
What do we do under overload?

Sorry we don't carry https here….

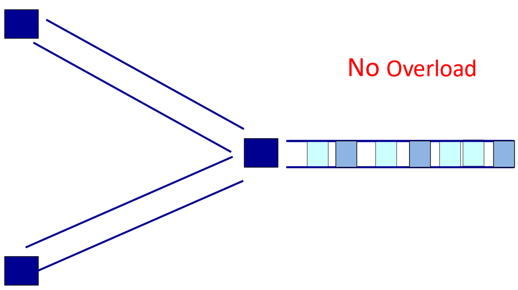
Statistical multiplexing: pipe view



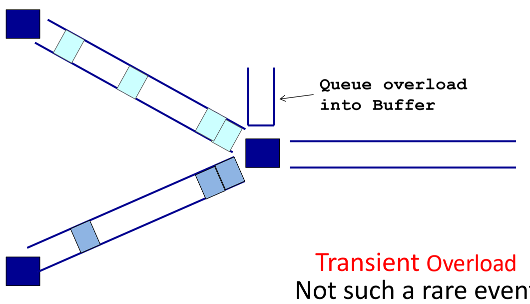
Statistical multiplexing: pipe view



Statistical multiplexing: pipe view

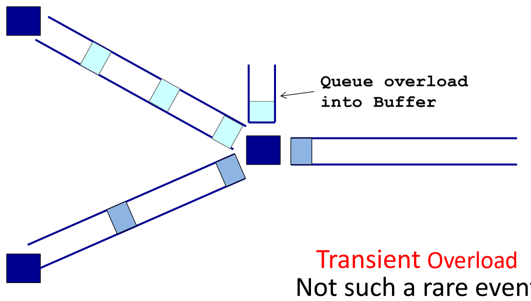


Statistical multiplexing: pipe view



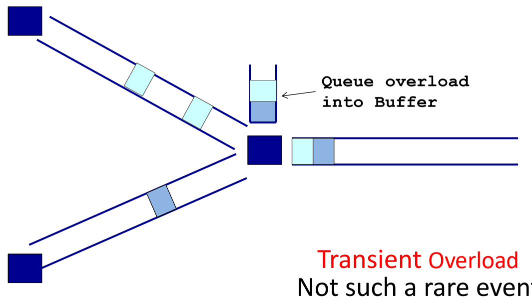
Transient Overload
Not such a rare event

Statistical multiplexing: pipe view



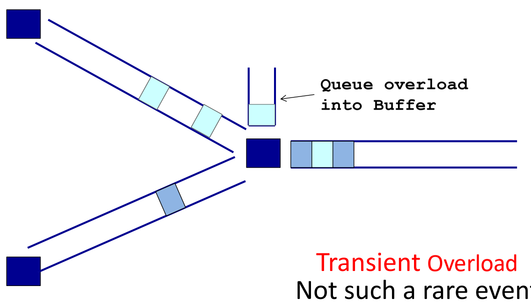
Transient Overload
Not such a rare event

Statistical multiplexing: pipe view



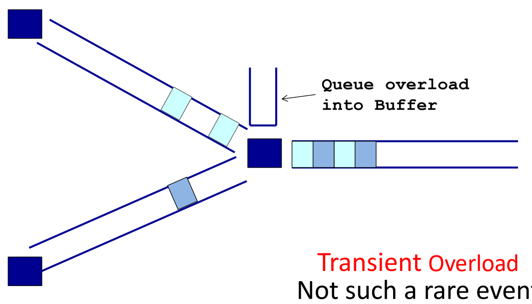
Transient Overload
Not such a rare event

Statistical multiplexing: pipe view



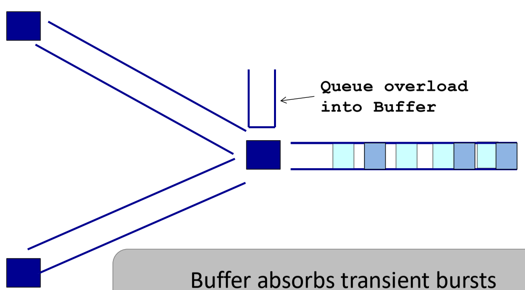
Transient Overload
Not such a rare event

Statistical multiplexing: pipe view



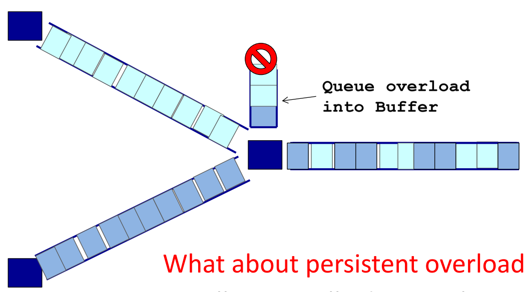
Transient Overload
Not such a rare event

Statistical multiplexing: pipe view



Buffer absorbs transient bursts
But *NOT* additional capacity

Statistical multiplexing: pipe view



What about persistent overload?
Will eventually drop packets

Queues introduce queuing delays

- Recall,

  packet delay = transmission delay + propagation delay (*)

- With queues (statistical multiplexing)

  packet delay = transmission delay + propagation delay + queuing delay (*)

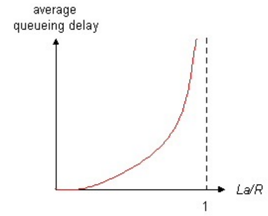
- Queuing delay caused by "packet interference"

- Made worse at high load
  - less "idle time" to absorb bursts
  - think about traffic jams at rush hour or rail network failure

(* plus per-hop *processing* delay that we define as negligible)

Queuing delay extremes

- R=link bandwidth (bps)
- L=packet length (bits)
- a=average packet arrival rate

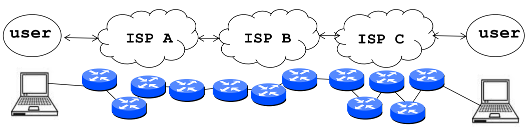


traffic intensity = La/R

- La/R ~ 0: average queuing delay small
- La/R -> 1: delays become large
- La/R > 1: more "work" arriving than can be serviced, average delay infinite – or data is lost (*dropped*).

Recall the Internet *federation*

- The Internet ties together different networks
  - >20,000 ISP networks



We can see (hints) of the nodes and links using traceroute…

"Real" Internet delays and routes

traceroute: rio.cl.cam.ac.uk to people.eng.unimelb.edu.au

(tracepath on winows is similar)

Three delay measurements from rio.cl.cam.ac.uk to gatwick.net.cl.cam.ac.uk

```
awm22@rio:~$ traceroute people.eng.unimelb.edu.au
traceroute to people.eng.unimelb.edu.au (128.250.59.37), 30 hops max, 60 byte packets
 1  vlan101.gatwick.net.cl.cam.ac.uk (128.232.32.2)  1.520 ms  1.822 ms  0.709 ms
 2  cl-wgb.d-mw.net.cam.ac.uk (193.60.89.5)  0.259 ms  0.256 ms  0.227 ms
 3  d-mw.c-ce.net.cam.ac.uk (131.111.6.53)  0.231 ms  0.381 ms  0.357 ms
 4  c-ce.b-ec.net.cam.ac.uk (131.111.6.82)  0.317 ms  0.481 ms  0.476 ms
 5  ae0.lowdss-ban1.ja.net (146.97.41.37)  2.842 ms  2.846 ms  2.821 ms
 6  ae26.lowdss-sbr1.ja.net (146.97.35.245)  2.877 ms  2.805 ms  2.795 ms
 7  ae28.londhx-sbr1.ja.net (146.97.33.17)  6.191 ms  6.109 ms  6.325 ms
 8  janet.mx1.lon.uk.geant.net (62.40.124.197)  6.319 ms  6.245 ms  6.258 ms
 9  138.44.226.6 (138.44.226.6)  169.704 ms  169.722 ms  169.682 ms
10  et-7-3-0.pe1.wmlb.vic.aarnet.net.au (113.197.15.28)  250.954 ms  251.163 ms  251.116 ms
11  * * *
12  4000v-eng-web-people-l.eng.unimelb.edu.au (128.250.59.37)  251.943 ms  251.952 ms  251.962 ms
13  4000v-eng-web-people-l.eng.unimelb.edu.au (128.250.59.37)  252.053 ms  252.018 ms  251.966 ms
14  * * *
15  4000v-eng-web-people-l.eng.unimelb.edu.au (128.250.59.37)  252.215 ms  252.088 ms  252.118 ms
16  4000v-eng-web-people-l.eng.unimelb.edu.au (128.250.59.37)  253.361 ms  253.109 ms  253.461 ms
17  4000v-eng-web-people-l.eng.unimelb.edu.au (128.250.59.37)  253.077 ms  253.832 ms  253.298 ms
18  * * *
….
29  * * *
30  * * *
```

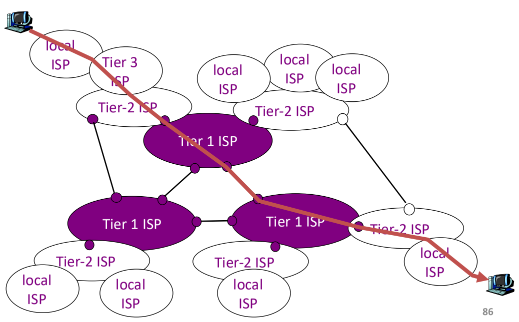
Direct London-Perth

Australian link

* means no response (probe or reply lost, router not replying)

Internet structure: network of networks

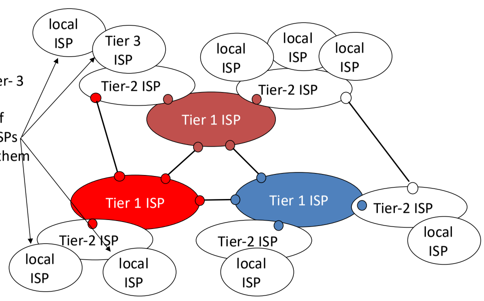
- a packet passes through many networks!



Internet structure: network of networks

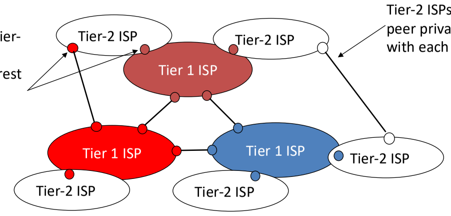
- "Tier-3" ISPs and local ISPs
  - last hop ("access") network (closest to end systems)

Local and tier- 3 ISPs are *customers* of higher tier ISPs connecting them to rest of Internet



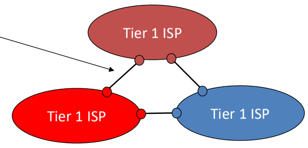
Internet structure: network of networks

- "Tier-2" ISPs: smaller (often regional) ISPs
  - Connect to one or more tier-1 ISPs, possibly other tier-2 ISPs

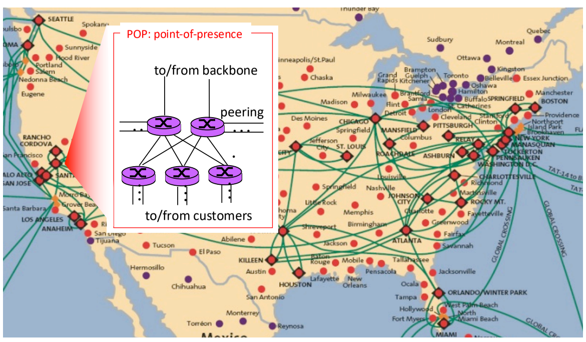


Internet structure: network of networks

- roughly hierarchical
- at center: "tier-1" ISPs (e.g., Verizon, Sprint, AT&T, Cable and Wireless), national/international coverage
  - treat each other as equals



Tier-1 ISP: e.g., Sprint



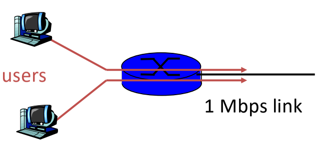
Packet Switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a "header" and "payload"
- Switches "forward" packets based on their headers
- Each packet travels independently
- No link resources are reserved in advance. Instead packet switching leverages statistical multiplexing
  - allows efficient use of resources
  - but introduces queues and queuing delays

Packet switching versus circuit switching

Packet switching may (does!) allow more users to use network

- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time
- *circuit-switching:*
  - 10 users
- *packet switching:*
  - with 35 users, probability > 10 active at same time is less than .0004



Q: how did we get value 0.0004?

Packet switching versus circuit switching

Q: how did we get value 0.0004?

- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time
- *circuit-switching:*
  - 10 users
- *packet switching:*
  - with 35 users, probability > 10 active at same time is less than .0004

Let U be number of users active
N the total users
P is 0.1 in our example to get 0.0004

$$P(u=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$\left[\therefore P(u \le K) = \sum_{k=0}^{K} \binom{n}{k} p^k (1-p)^{n-k}\right] \therefore \left[P(u>K) = 1 - \sum_{k=0}^{K} \binom{n}{k} p^k (1-p)^{n-k}\right]$$

for $n = 35,\ K = 10$

$$P(u \le 10) = \sum_{k=0}^{10} \binom{35}{k} p^k (1-p)^{35-k}$$

where $p = 0.1$:

$P(u \le 10) = 0.99958$

$\therefore P(u > 10) = 0.00042$

Circuit switching: pros and cons

- Pros
  - guaranteed performance
  - fast transfers (once circuit is established)
- Cons
  - wastes bandwidth if traffic is "bursty"
  - connection setup adds delay
  - recovery from failure is slow

97

Packet switching: pros and cons

- Cons
  - no guaranteed performance
  - header overhead per packet
  - queues and queuing delays
- Pros
  - efficient use of bandwidth (stat. muxing)
  - no overhead due to connection setup
  - resilient -- can `route around trouble'

98

Summary

- A sense of how the basic `plumbing' works
  - links and switches
  - packet delays= transmission + propagation + queuing + (negligible) per-switch processing
  - statistical multiplexing and queues
  - circuit vs. packet switching

99

Topic 2 – Architecture and Philosophy

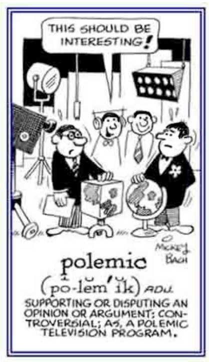
- Abstraction
- Layering
- Layers and Communications
- Entities and Peers
- What is a protocol?
- Protocol Standardization
- The architects process
  - How to break system into modules
  - Where modules are implemented
  - Where is state stored
- Internet Philosophy and Tensions

1

TRIGGER WARNING

- Philosophy,
- Bad Analogies, and
- RANTS verging on POLEMIC

Will follow.....



Abstraction Concept

A mechanism for breaking down a problem

what not *how*

- eg Specification *versus* implementation
- eg Modules in programs

Allows replacement of implementations without affecting system behavior

Vertical versus *Horizontal*

"Vertical" what happens in a box "How does it attach to the network?"

"Horizontal" the communications paths running through the system

Hint: paths are built ("layered") on top of other paths

Computer System Modularity

Partition system into modules & abstractions:

- Well-defined interfaces give flexibility
  - ***Hides*** implementation - can be freely changed
  - Extend functionality of system by adding new modules
- E.g., libraries encapsulating set of functionality
- E.g., programming language + compiler abstracts away how the particular CPU works …

Computer System Modularity (cnt' d)

- Well-defined interfaces hide information
  - Isolate assumptions
  - Present high-level abstractions
- **But can impair performance!**
- Ease of implementation vs worse performance

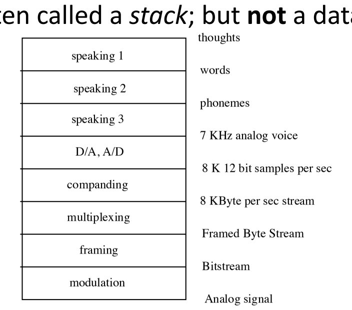
Network System Modularity

Like software modularity, but:

- Implementation is distributed across many machines (routers and hosts)
- Must decide:
  - How to break system into modules
    - **Layering**
  - Where modules are implemented
    - **End-to-End Principle**
  - Where state is stored
    - **Fate-sharing**

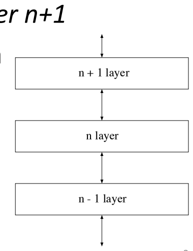
Layering Concept

- A restricted form of abstraction: system functions are divided into layers, one built upon another
- Often called a *stack*; but **not** a data structure!



Layers and Communications

- Interaction only between adjacent layers
- *layer n* uses services provided by *layer n-1*
- *layer n* provides service to *layer n+1*
- Bottom layer is physical media
- Top layer is application



Entities and Peers

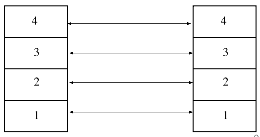
Entity – a *thing* (an independent existence)

Entities *interact* with the layers above and below

Entities *communicate* with *peer* entities

- same level but different place (eg different person, different box, different host)

Communications between peers is supported by entities at the lower layers



Entities and Peers

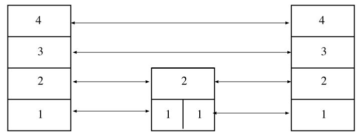
Entities usually do something useful

- Encryption – Error correction – Reliable Delivery
- Nothing at all is also reasonable

Not all communications is end-to-end

Examples for things in the middle

- IP Router – Mobile Phone Cell Tower
- Person translating French to English



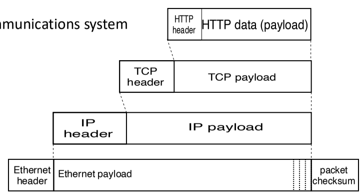
Layering and Embedding

In Computer Networks we often see higher-layer information embedded within lower-layer information

- Such embedding can be considered a form of layering
- Higher layer information is generated by stripping off headers and trailers of the current layer
- eg an IP entity only looks at the IP headers

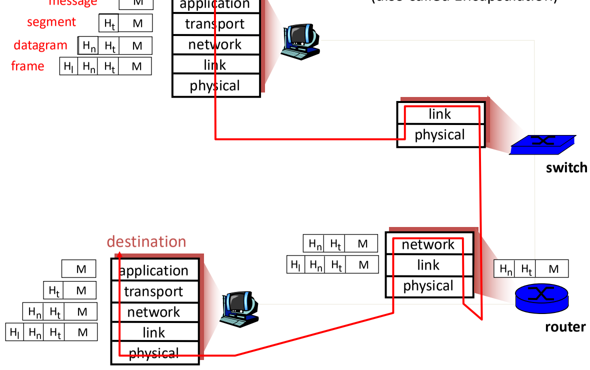
BUT embedding is not the only form of layering

Layering is to help understand a communications system
NOT
determine implementation strategy

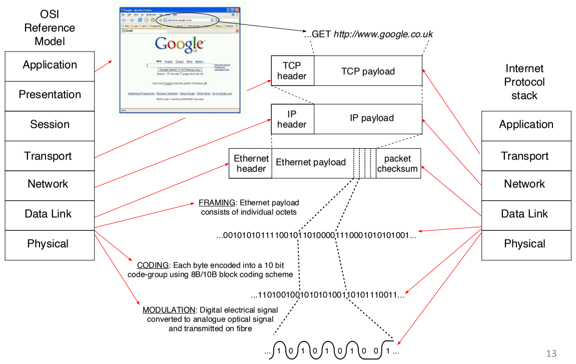


Example Embedding

(also called Encapsulation)

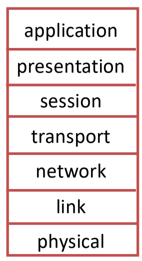


Internet protocol stack *versus* OSI Reference Model

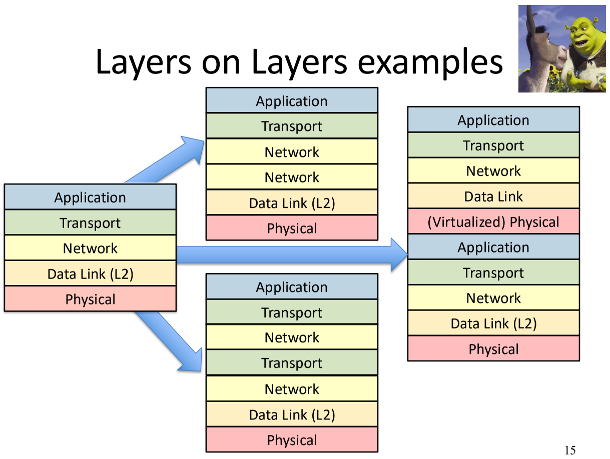


ISO/OSI reference model

- presentation: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application



Layers on Layers examples



What is a protocol?

human protocols:

- "what's the time?"
- "I have a question"
- introductions

... specific msgs sent

... specific actions taken when msgs received, or other events

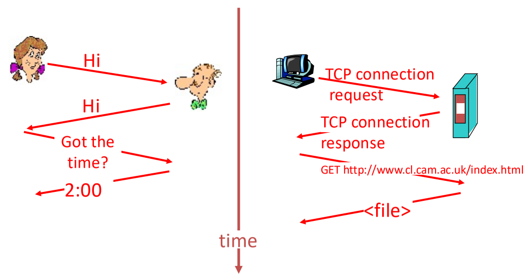
network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt

What is a protocol?

a human protocol and a computer network protocol:



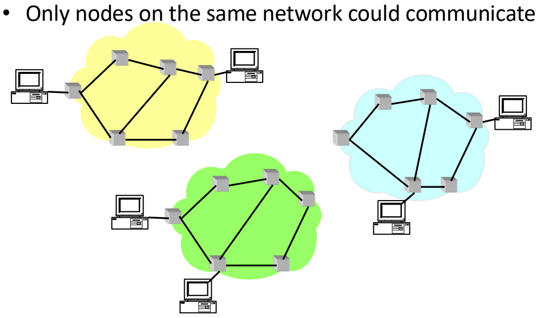
Q: Other human protocols?

Protocol Standardization

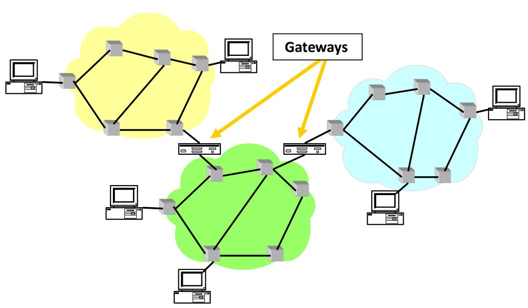
- All hosts must follow same protocol
  - Very small modifications can make a big difference
  - Or prevent it from working altogether
- This is why we have standards
  - Can have multiple implementations of protocol
- Internet Engineering Task Force (IETF)
  - Based on working groups that focus on specific issues
  - Produces "Request For Comments" (RFCs)
  - IETF Web site is ***http://www.ietf.org***
  - RFCs archived at ***http://www.rfc-editor.org***

So many Standards Problem

- Many different packet-switching networks
- Each with its own Protocol
- Only nodes on the same network could communicate



INTERnet Solution



Internet Design Goals (Clark '88)

- **Connect existing networks**
- Robust in face of failures
- Support multiple types of delivery services
- Accommodate a variety of networks
- Allow distributed management
- Easy host attachment
- Cost effective
- Allow resource accountability

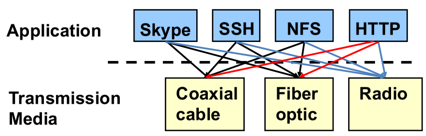
Real Goals

Internet Motto

We reject kings , presidents, and voting. We believe in rough consensus and running code." – David Clark

- **Build something that works!**
- Connect existing networks
- Robust in face of failures
- Support multiple types of delivery services
- Accommodate a variety of networks
- Allow distributed management
- Easy host attachment
- Cost effective
- ~~Allow resource accountability~~

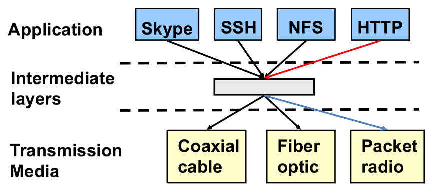
A Multitude of Apps Problem



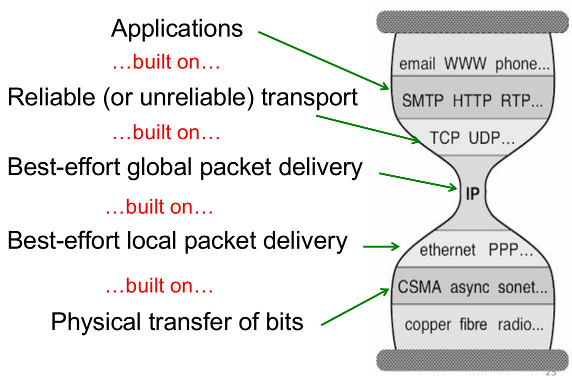
- Re-implement every application for every technology?
- No! But how does the Internet design avoid this?

Solution: Intermediate Layers

- Introduce intermediate layers that provide set of abstractions for various network functionality and technologies
  - A new app/media implemented only once
  - Variation on "add another level of indirection"

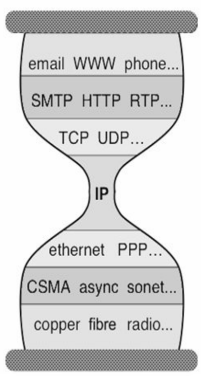


In the context of the Internet



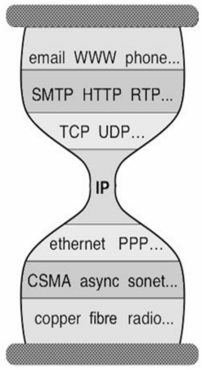
Three Observations

- Each layer:
  - Depends on layer below
  - Supports layer above
  - Independent of others
- Multiple versions in layer
  - Interfaces differ somewhat
  - Components pick which lower-level protocol to use
- But only one IP layer
  - Unifying protocol



Layering Crucial to Internet's Success

- Reuse
- Hides underlying detail
- Innovation at each level can proceed in parallel
- Pursued by very different communities



What are some of the drawbacks of protocols and layering?

Drawbacks of Layering

- Layer N may duplicate lower layer functionality
  - e.g., error recovery to retransmit lost data
- Information hiding may hurt performance
  - e.g., packet loss due to corruption vs. congestion
- Headers start to get really big
  - e.g., typical TCP+IP+Ethernet is 54 bytes
- Layer violations when the gains too great to resist
  - e.g., TCP-over-wireless
- Layer violations when network doesn't trust ends
  - e.g., firewalls

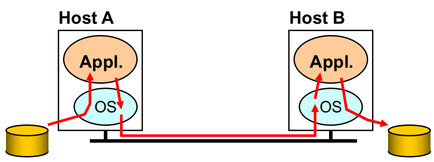
Placing Network Functionality

- Hugely influential paper: "End-to-End Arguments in System Design" by Saltzer, Reed, and Clark ('84)
  - articulated as the "End-to-End Principle" (E2E)
- Endless debate over what it means
- Everyone cites it as supporting their position
  (regardless of the position!)

Basic Observation

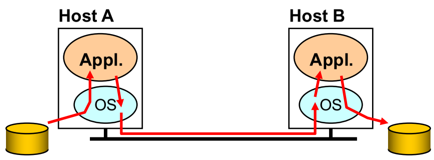
- Some application requirements can only be correctly implemented **end-to-end**
  - reliability, security, *etc.*
- Implementing these in the network is hard
  - every step along the way must be fail proof
- Hosts
  - **Can** satisfy the requirement without network's help
  - **Will/must** do so, since they can't rely on the network

Example: Reliable File Transfer



- Solution 1: make each step reliable, and string them together to make reliable end-to-end process

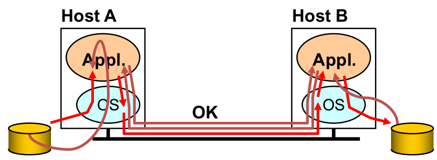
Example: Reliable File Transfer



- Solution 1: make each step reliable, and string them together to make reliable end-to-end process

So what is the problem?
each component is 0.9 reliable
leads to total system failure of >0.4*

Example: Reliable File Transfer



- Solution 1: make each step reliable, and string them together to make reliable end-to-end process
- Solution 2: end-to-end **check** and retry

Discussion

- Solution 1 is incomplete
  - What happens if any network element misbehaves?
  - Receiver has to do the check anyway!
- Solution 2 is complete
  - Full functionality can be entirely implemented at application layer with no need for reliability from lower layers
- Is there any need to implement reliability at lower layers?

Summary of End-to-End Principle

- Implementing functionality (e.g., reliability) in the network
  - Doesn't reduce host implementation complexity
  - Does increase network complexity
  - Probably increases delay and overhead on all applications even if they don't need the functionality (e.g. VoIP)
- However, implementing in the network can improve performance in some cases
  - e.g., consider a very lossy link

"Only-if-Sufficient" Interpretation

- Don't implement a function at the lower levels of the system unless it can be completely implemented at this level
- *Unless you can relieve the burden from hosts, don't bother*

"Only-if-Necessary" Interpretation

- Don't implement *anything* in the network that can be implemented correctly by the hosts
- Make network layer absolutely minimal
  - This E2E interpretation trumps performance issues
  - Increases flexibility, since lower layers stay **simple**

“Only-if-Useful” Interpretation

- If hosts can implement functionality correctly, implement it in a lower layer only as a performance enhancement
- But do so only if it does not impose burden on applications that do not require that functionality

We have some tools:

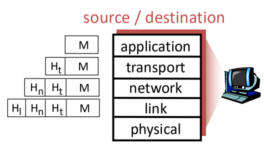
- Abstraction
- Layering
- Layers and Communications
- Entities and Peers
- Protocol as motivation
- Examples of the architects process
- Internet Philosophy and Tensions

Distributing Layers Across Network

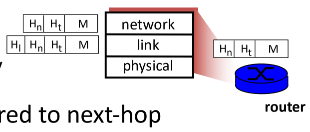
- Layers are simple if only on a single machine
  - Just stack of modules interacting with those above/below
- But we need to implement layers across machines
  - Hosts
  - Routers (switches)
- What gets implemented where?

What Gets Implemented on Host?

- Bits arrive on wire, must make it up to application
- Therefore, all layers must exist at the host



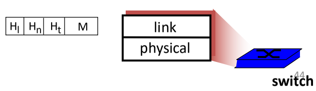
What Gets Implemented on a Router?



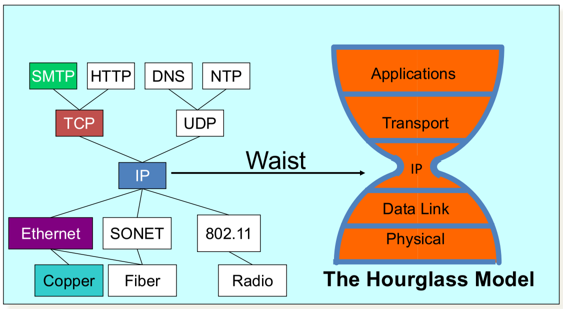
- Bits arrive on wire
  - Physical layer necessary
- Packets must be delivered to next-hop
  - Datalink layer necessary
- Routers participate in global delivery
  - Network layer necessary
- Routers don’t support reliable delivery
  - Transport layer (and above) ***not*** supported

What Gets Implemented on Switches?

- Switches do what routers do, except they don’t participate in global delivery, just local delivery
- They only need to support Physical and Datalink
  - Don’t need to support Network layer
- Won’t focus on the router/switch distinction
  - Almost all boxes support network layer these days
  - Routers have switches but switches do not have routers

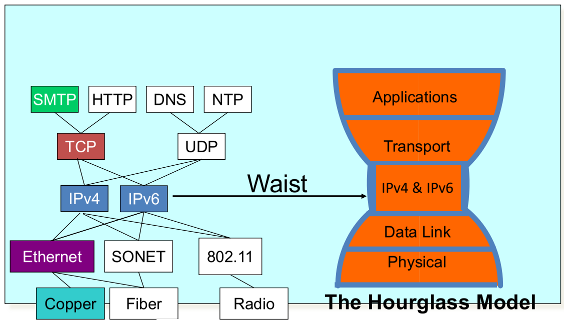


The Internet *Hourglass*



There is just one network-layer protocol, **IP**.
The "narrow waist" facilitates interoperability.

The middle-age Internet *Hourglass*



There is just ~~one~~ TWO network-layer protocol, **IP**v4 + v6
The "narrow waist" facilitates interoperability(???).

Alternative to Standardization?

- Have one implementation used by everyone
- Open-source projects
  - Which has had more impact, Linux or POSIX?
- Or just sole-sourced implementation
  - Skype, Signal, FaceTime, etc.