Computer Networking

Lent Term M/W/F 11:00-12:00
LT1 in Gates Building

Slide Set 1

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

January 2014

Topic 1 Foundation

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

Course Administration

Commonly Available Texts

- Computer Networking: A Top-Down Approach

  Kurose and Ross, 6th edition 2013, Addison-Wesley

  (5th edition is also commonly available)
- Computer Networks: A Systems Approach

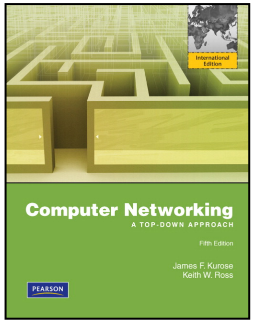
  Peterson and Davie, 5th edition 2011, Morgan-Kaufman

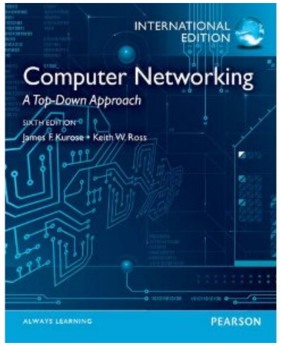
Other Selected Texts (non-representative)

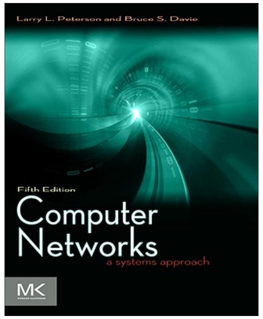
- Internetworking with TCP/IP, vol. I + II

  Comer & Stevens, Prentice Hall
- UNIX Network Programming, Vol. I

  Stevens, Fenner & Rudoff, Prentice Hall







Thanks

- Slides are a fusion of material from

  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, and Adam Greenhalgh (and to those others I've forgotten, sorry.)

- Supervision material is drawn from

  Stephen Kell, Andy Rice

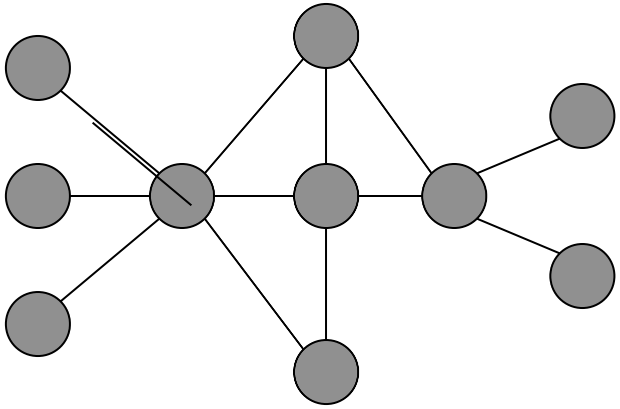
- Practical material will become available through this year

  But would be impossible without Nick McKeown, David Underhill, Matthew Ireland, Andrew Ryrie and Antanas Uršulis

- Finally thanks to the Part 1b students past 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 very vague

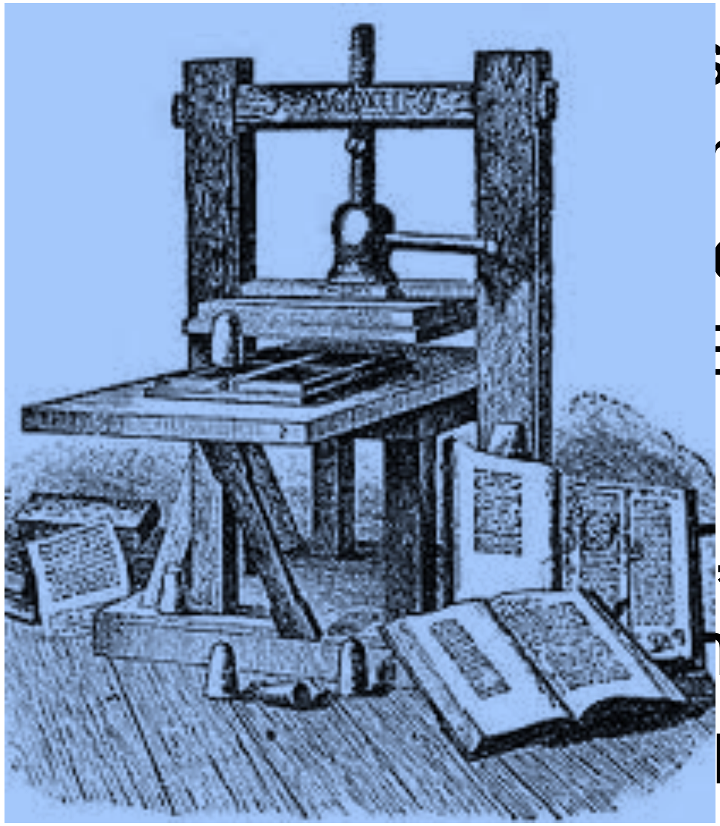
There are *many* different types of networks

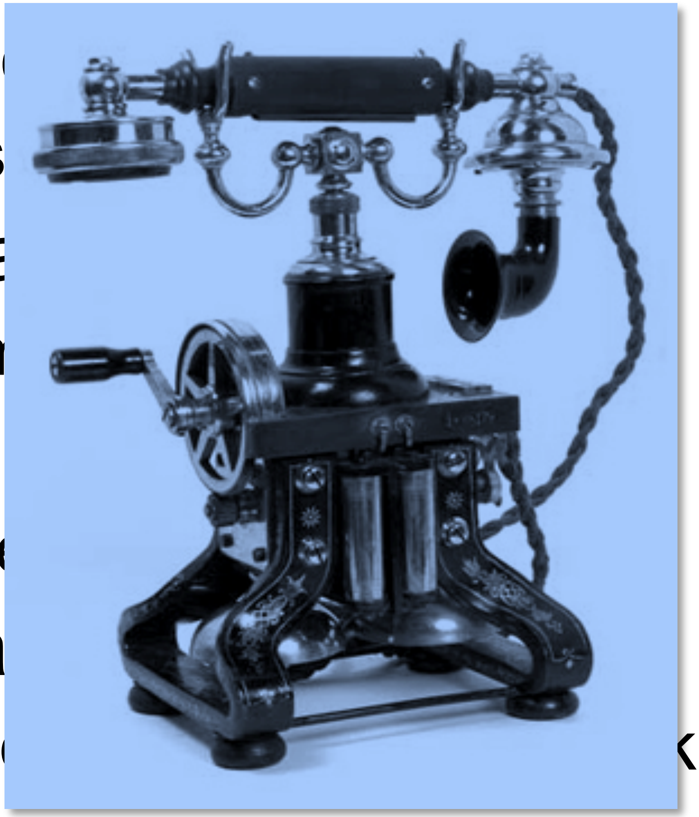
- Internet
- Telephone network
- Transportation networks
- Cellular networks
- Supervisory control and data acquisition networks
- Optical networks
- Sensor networks

We will focus almost exclusively on the Internet

The Internet is transforming everything

- sin
  rtis
- ela
  E-m
-
  , se
- n a
  p, (ks





Took the dissemination of information to the next level

The Internet is big business

- Many large and influential networking companies
  - Cisco, Broadcom, AT&T, Verizon, Akamai, Huawei, …
  - $120B+ industry (carrier and enterprise alone)

- Networking central to most technology companies
  - Google, Facebook, Intel, HP, Dell, VMware, …

Internet research has impact

- The Internet started as a research experiment!
- 4 of 10 most cited authors work in networking
- *Many* successful companies have emerged from networking research(ers)

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 AT&T*

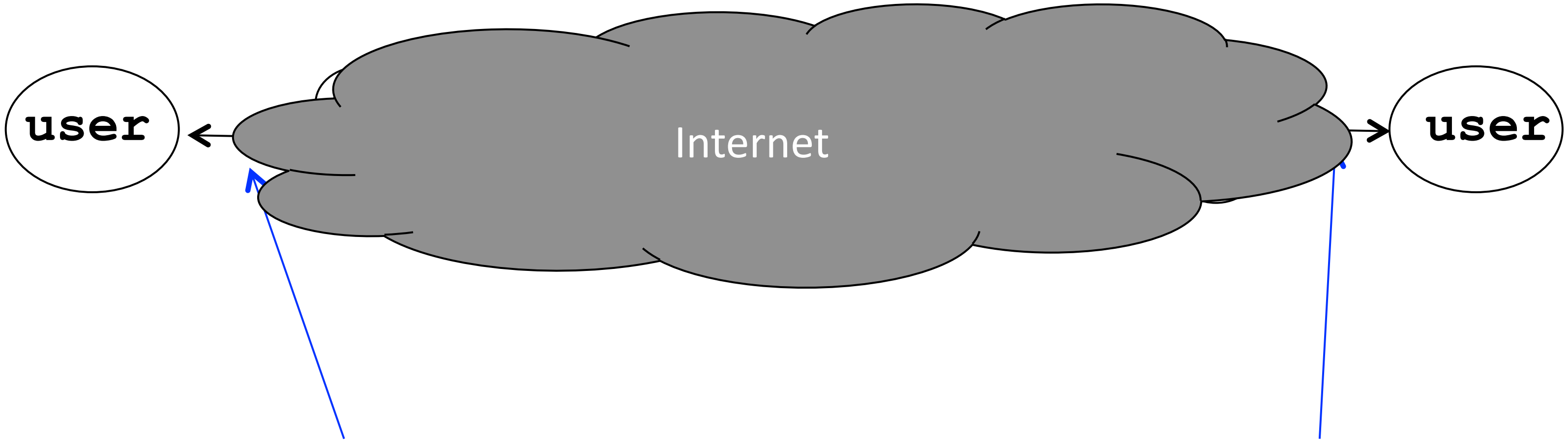
“What’s with all these TLA protocols?” – *all*

“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
  - >18,000 ISP networks



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
  - >18,000 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

- 2.4 Billion users (34% of world population)
- 1 Trillion unique URLs
- 294 Billion emails sent per day
- 1 Billion smartphones
- 937 Million Facebook users
- 2 Billion YouTube videos watched per day
- Routers that switch 10Terabits/second
- Links that carry 100Gigabits/second

Enormous diversity and dynamic range

- Communication latency: microseconds to seconds (10^6)
- Bandwidth: 1Kbits/second to 100 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

- 100+Gigabits/second backbone links
- 5B+ devices, all over the globe
- 20M Facebook apps installed per day

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, modem, 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

An Engineered System

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

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 theoretical models
- "Working code" doesn't mean much
- Performance benchmarks are too narrow

Performance – not just bits per second

Second order effects

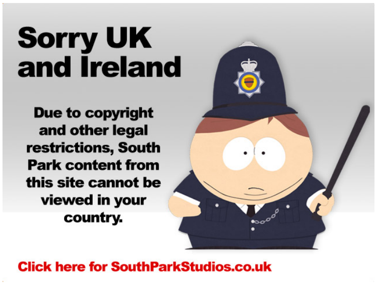
- Image/Audio quality

Other metrics…

- Network efficiency (good-put *versus* throughput)

- User Experience? (World Wide Wait)

- Network connectivity expectation



Sorry UK and Ireland

Due to copyright and other legal restrictions, South Park content from this site cannot be viewed in your country.

Click here for SouthParkStudios.co.uk

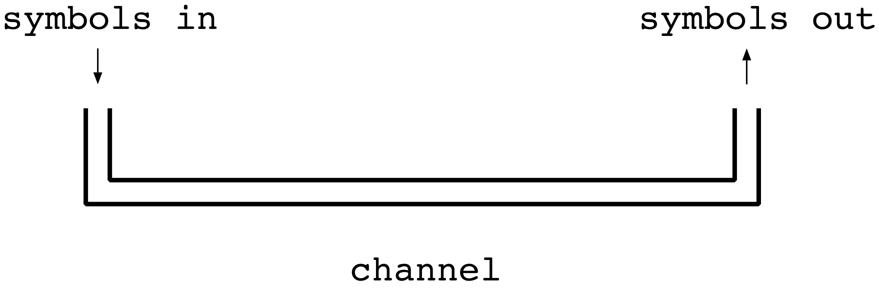
- Others?

Channels Concept

(This channel definition is very abstract)

- Peer entities communicate over channels
- Peer entities provide higher-layer peers with higher-layer channels

A channel is that into which an entity puts symbols and which causes those symbols *(or a reasonable approximation) to appear somewhere else at a later point in time.*



Channel Characteristics

Symbol type: bits, packets, waveform

Capacity: bandwidth, data-rate, packet-rate

Delay: fixed or variable

Fidelity: signal-to-noise, bit error rate, packet error rate

Cost: per attachment, for use

Reliability

Security: privacy, unforgability

Order preserving: always, almost, usually

Connectivity: point-to-point, to-many, many-to-many

Examples:

- Fibre Cable
- 1 Gb/s channel in a network
- Sequence of packets transmitted between hosts
- A telephone call (handset to handset)
- The audio channel in a room
- Conversation between two people

Example Physical Channels

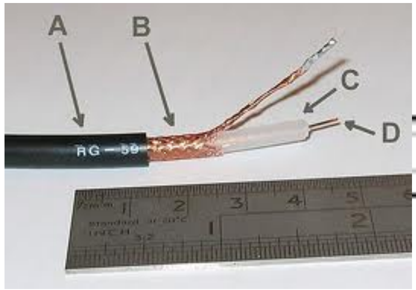
these example physical channels are also known as *Physical Media*

Twisted Pair (TP)

- two insulated copper wires
  - Category 3: traditional phone wires, 10 Mbps Ethernet
  - Category 6: 1Gbps Ethernet
- Shielded (STP)
- Unshielded (UTP)

Coaxial cable:

- two concentric copper conductors
- bidirectional
- baseband:
  - single channel on cable
  - legacy Ethernet
- broadband:
  - multiple channels on cable
  - HFC (Hybrid Fiber Coax)



Fiber optic cable:

- high-speed operation
- point-to-point transmission
- (10’s-100’s Gps)
- low error rate
- immune to electromagnetic noise

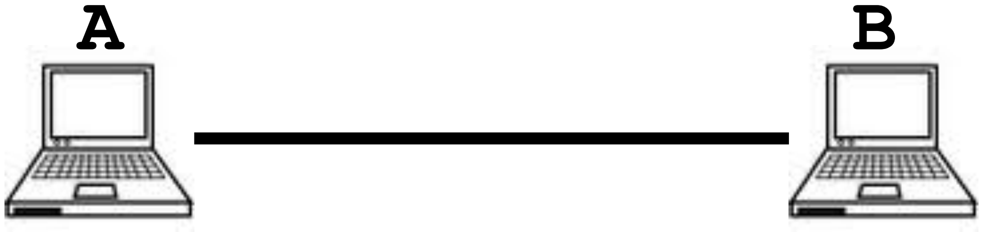
More Physical media: Radio

- Bidirectional and multiple access
- propagation environment effects:
  - reflection
  - obstruction by objects
  - interference

Radio link types:

- terrestrial microwave
  - e.g. 45 Mbps channels
- LAN (e.g., Wifi)
  - 11Mbps, 54 Mbps, 200 Mbps
- wide-area (e.g., cellular)
  - 4G cellular: ~ 4 Mbps
- satellite
  - Kbps to 45Mbps channel (or multiple smaller channels)
  - 270 msec end-end delay
  - geosynchronous versus low altitude

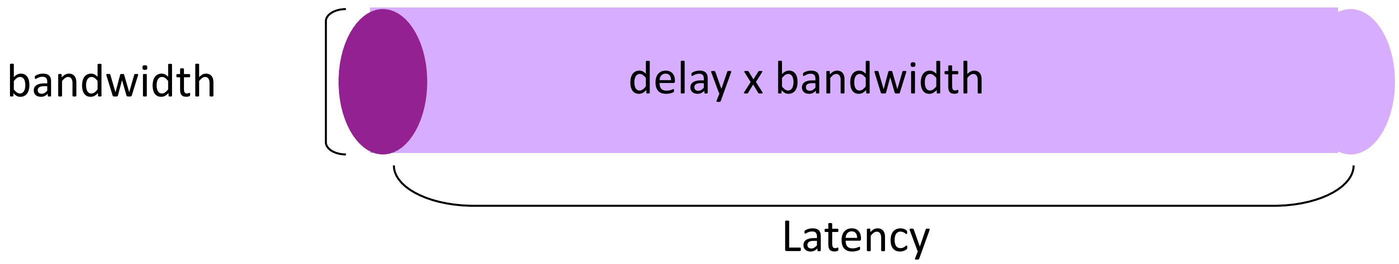
Nodes and Links



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

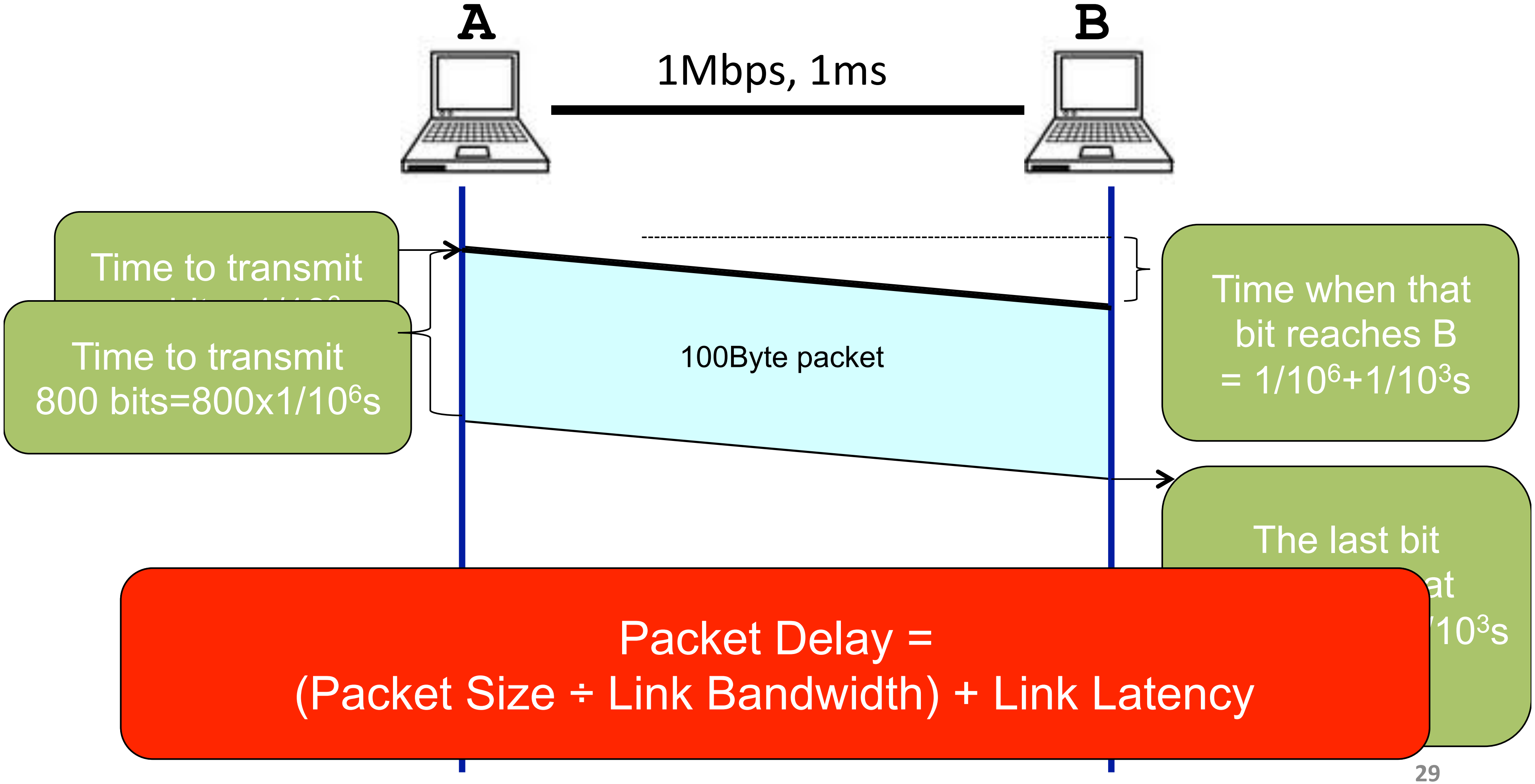
Examples of Bandwidth-Delay

- Same city over a slow link:
  - BW~100Mbps
  - Latency~0.1msec
  - BDP ~ 10,000bits ~ 1.25KBytes

- Cross-country over fast link:
  - BW~10Gbps
  - Latency~10msec
  - BDP ~ 10^8bits ~ 12.5GBytes

Packet Delay

Sending a 100B packet from A to B?



Time to transmit

Time to transmit 800 bits=800x1/10^6s

Time when that bit reaches B = $1/10^6+1/10^3$s

The last bit

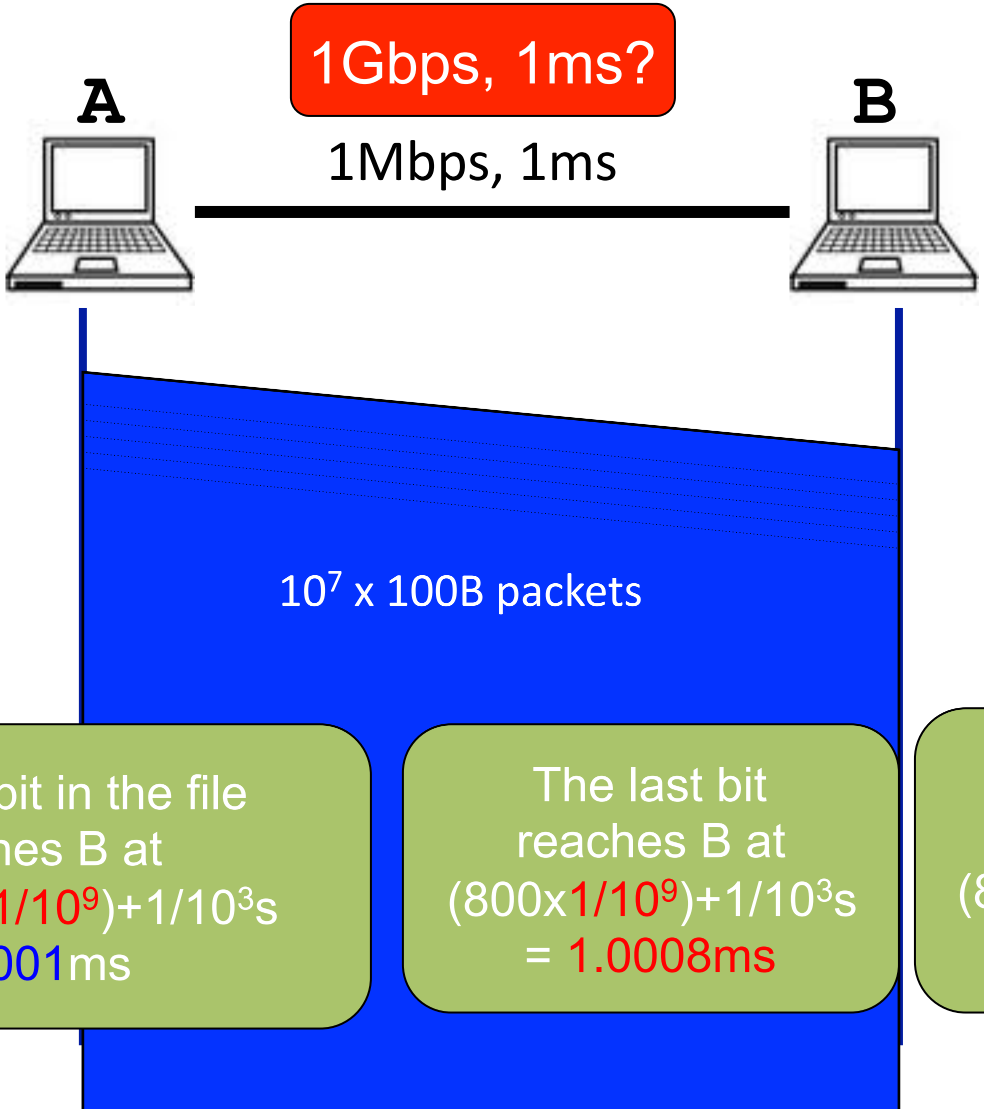
at

$/10^3$s

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

1GB file in 100B packets ay

Sending a 100B packet from A to B?



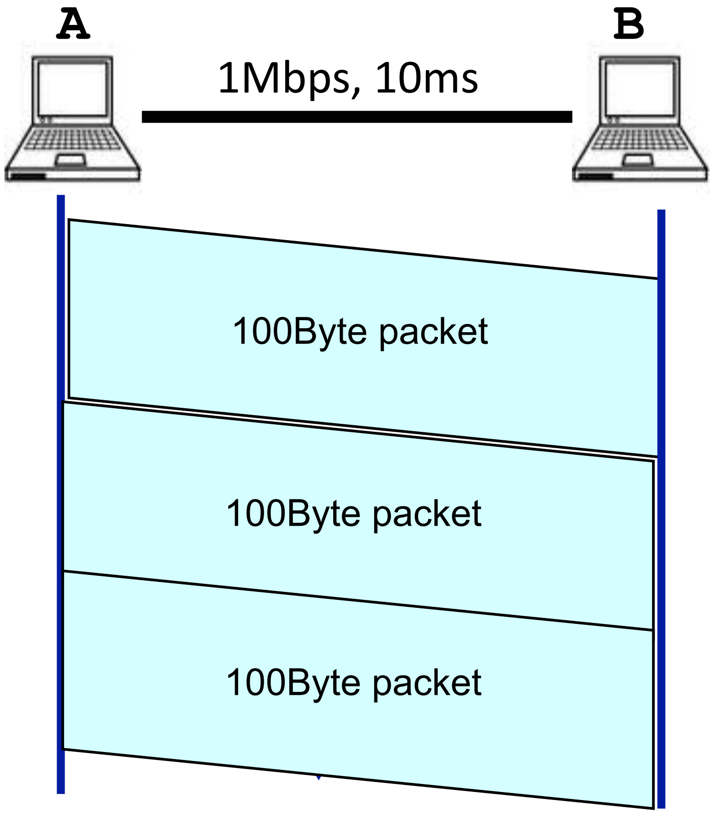
The last bit in the file reaches B at $(10^7\text{x}800\text{x}1/10^9)+1/10^3$s = 8001ms

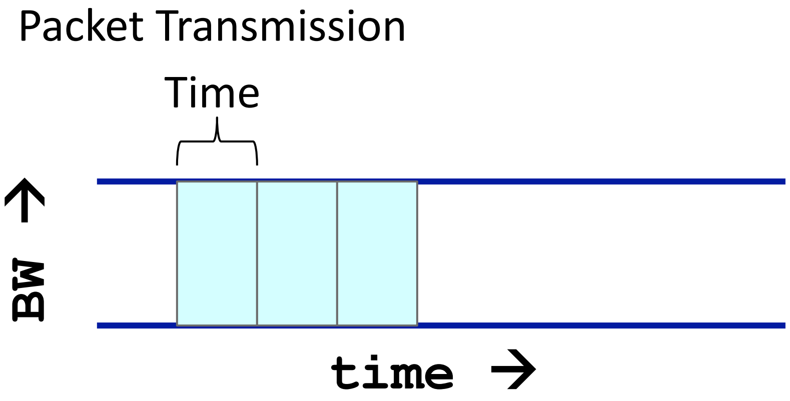
The last bit reaches B at $(800\text{x}1/10^9)+1/10^3$s = 1.0008ms

The last bit reaches B at $(800\text{x}1/10^6)+1/10^3$s = 1.8ms

Packet Delay: The “pipe” view

Sending 100B packets from A to B?

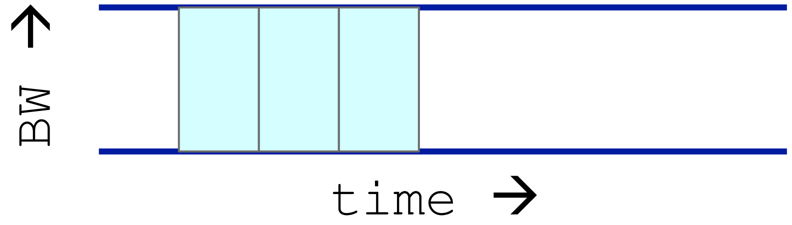




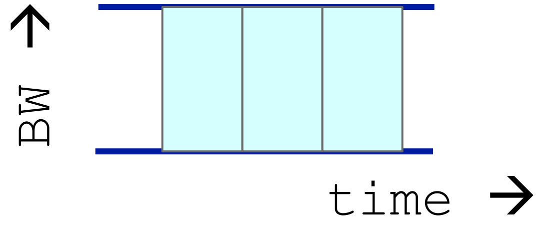
Packet Delay: The “pipe” view

Sending 100B packets from A to B?

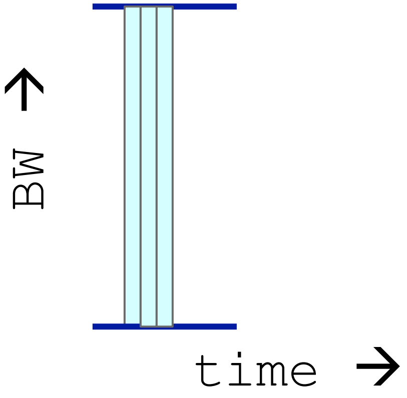
1Mbps, 10ms (BDP=10,000)



1Mbps, 5ms (BDP=5,000)



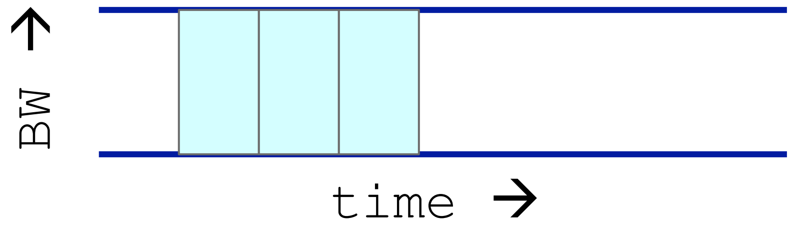
10Mbps, 1ms (BDP=10,000)



Packet Delay: The “pipe” view

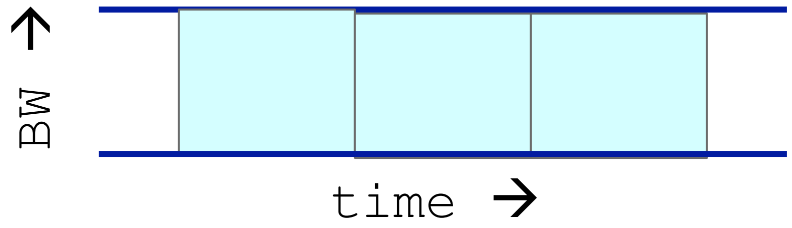
Sending 100B packets from A to B?

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

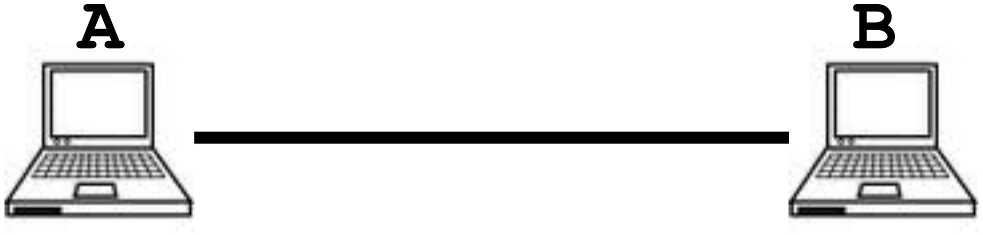


What if we used *200Byte packets??*

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

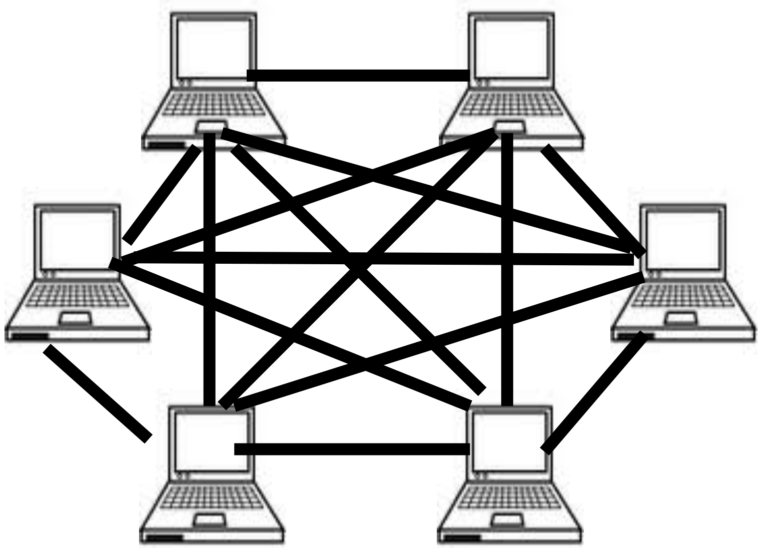


Recall Nodes and Links



What if we have more nodes?

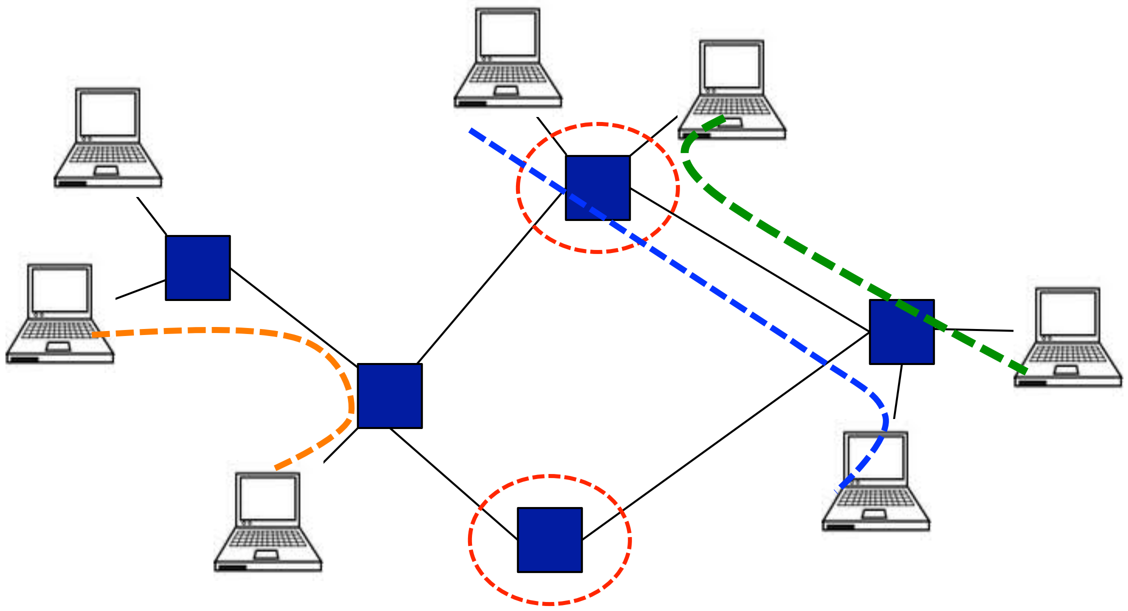
One link for every node?



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

Solution: A switched network

Nodes share network link resources



How is this sharing implemented?

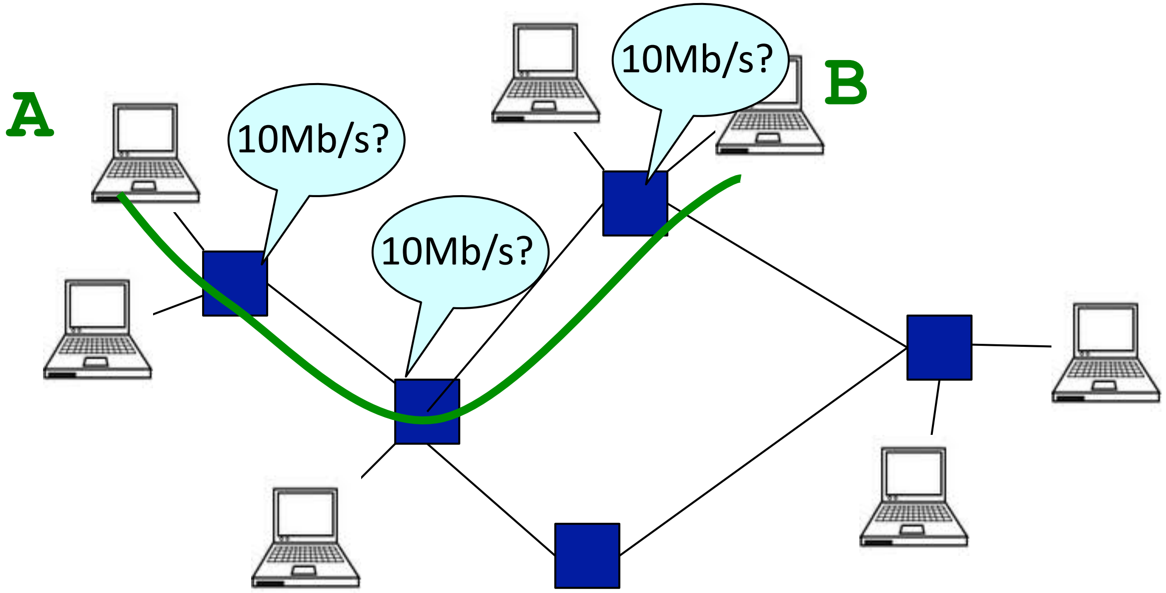
Two forms of switched networks

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

- Packet switching (used in the Internet)

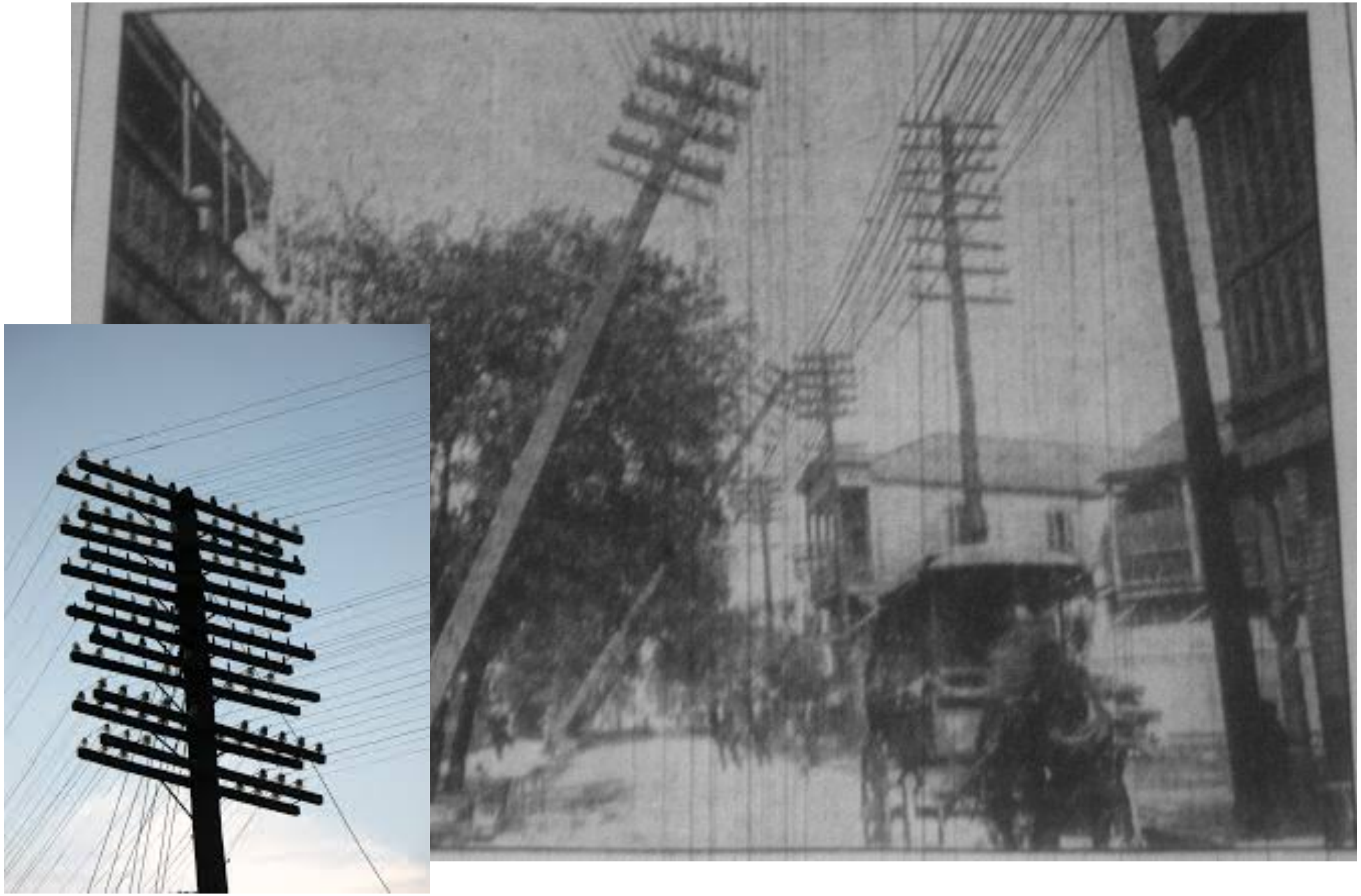
Circuit switching

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

Old Time Multiplexing

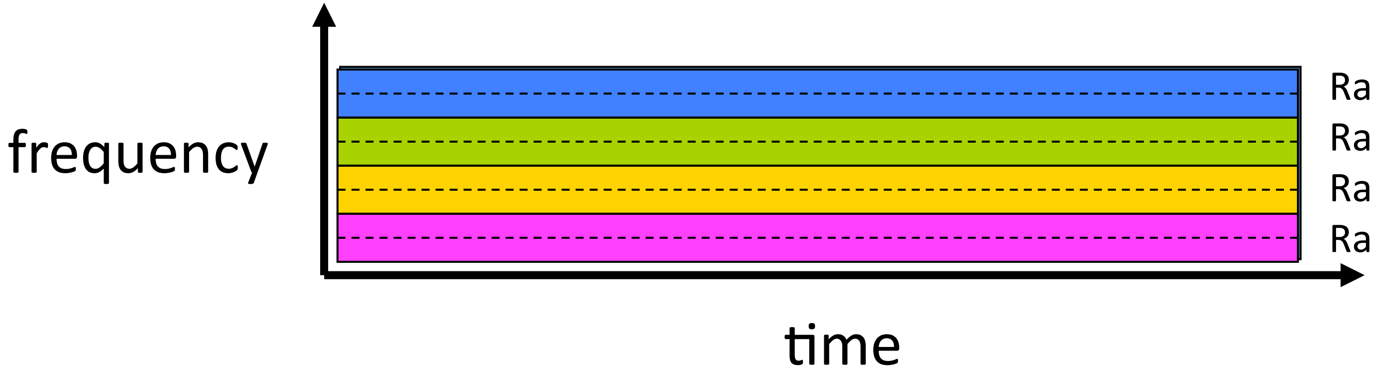


Circuit Switching: FDM and TDM

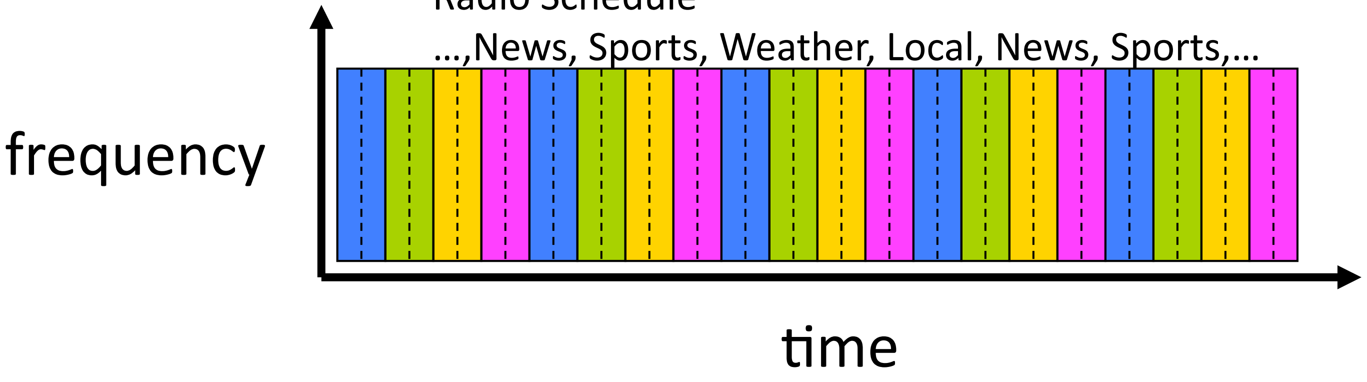
Example:

4 users 🟦🟩🟨🟪

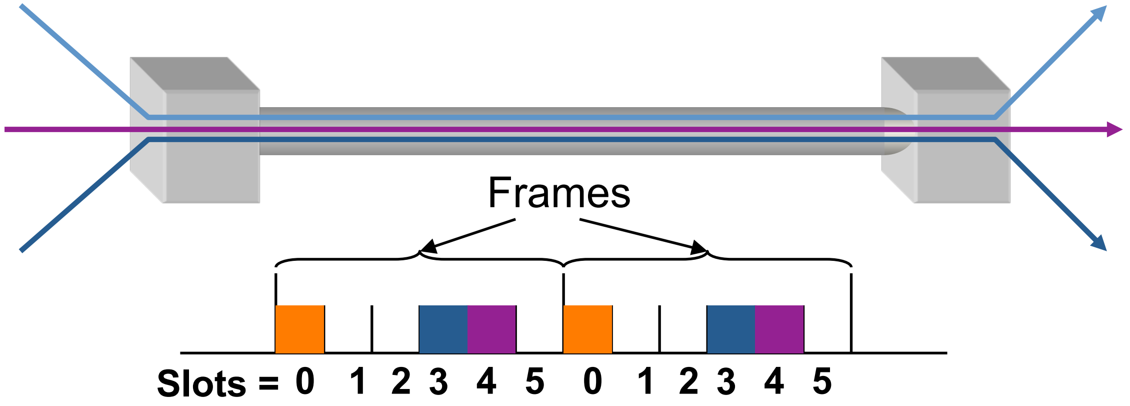
Frequency Division Multiplexing



Time Division Multiplexing

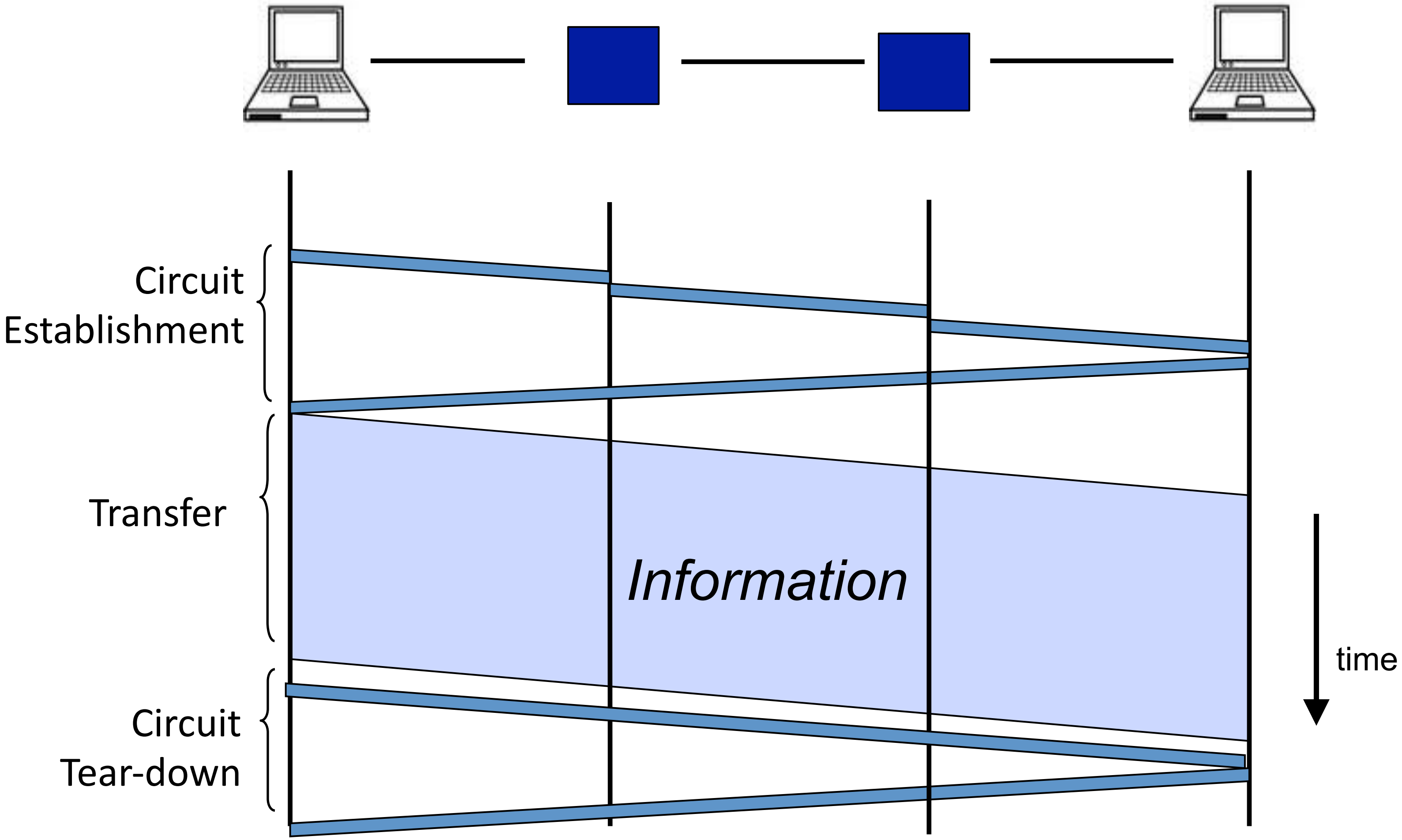


Time-Division Multiplexing/Demultiplexing



- 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

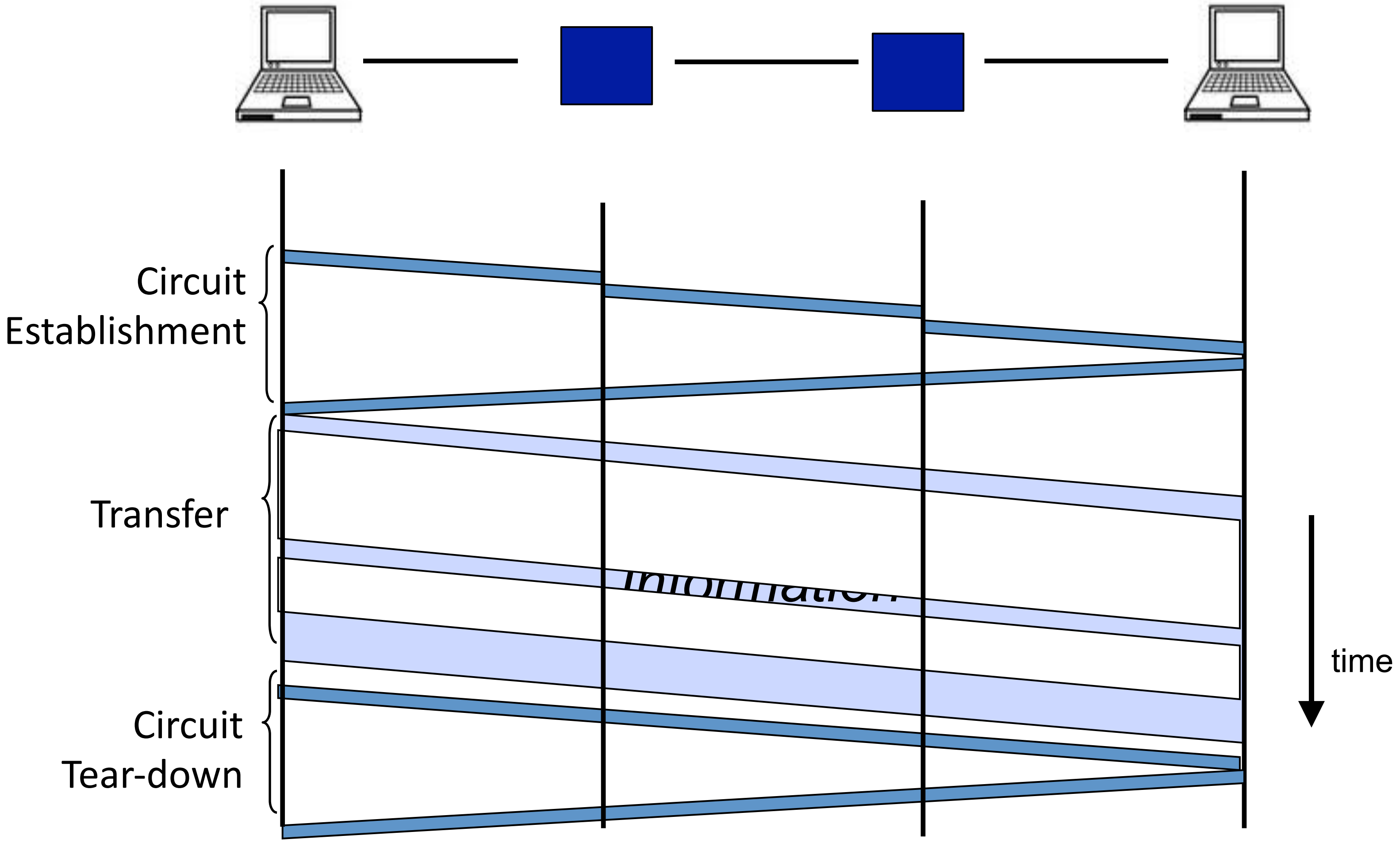


Circuit switching: pros and cons

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

- Cons

Timing in Circuit Switching

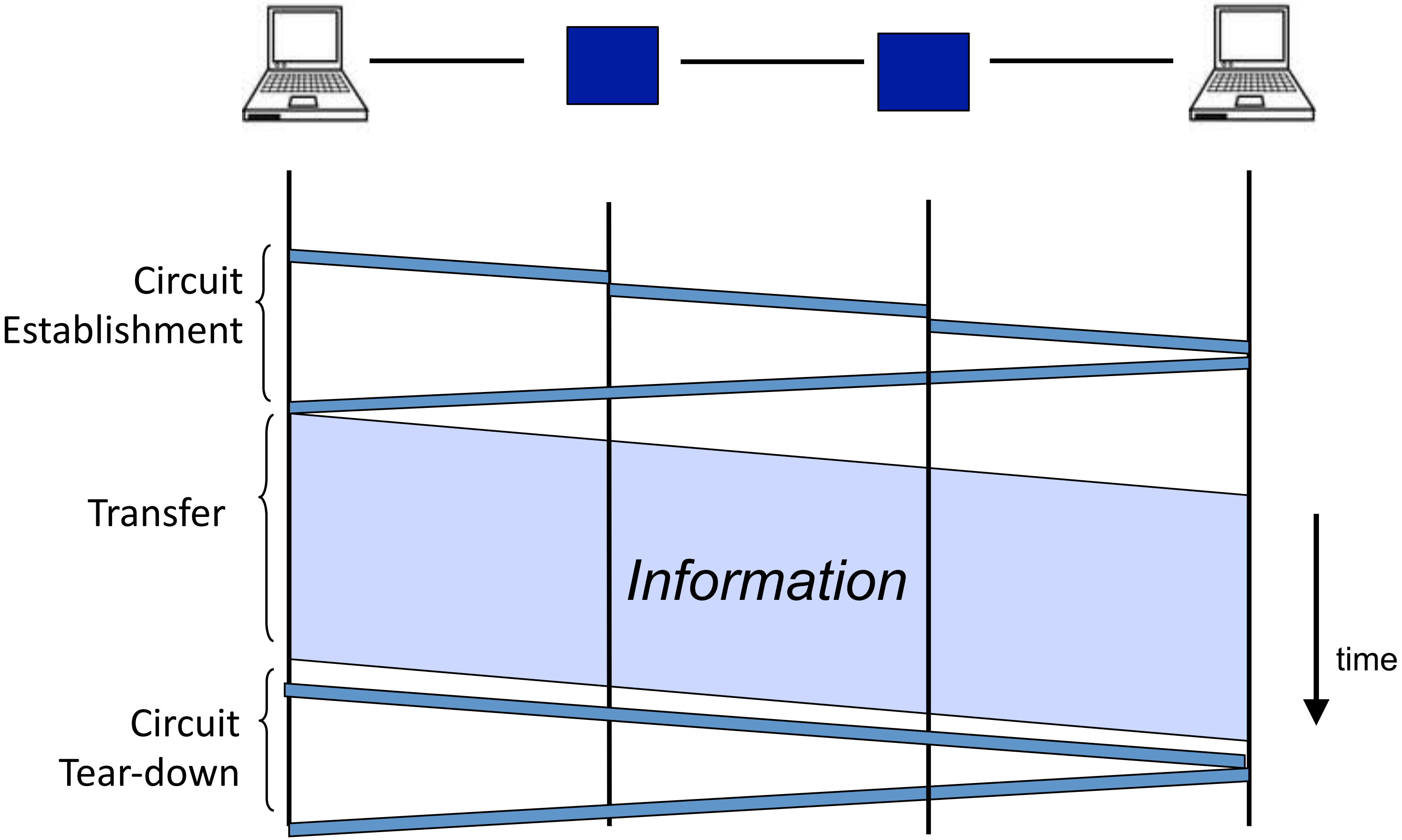


Circuit switching: pros and cons

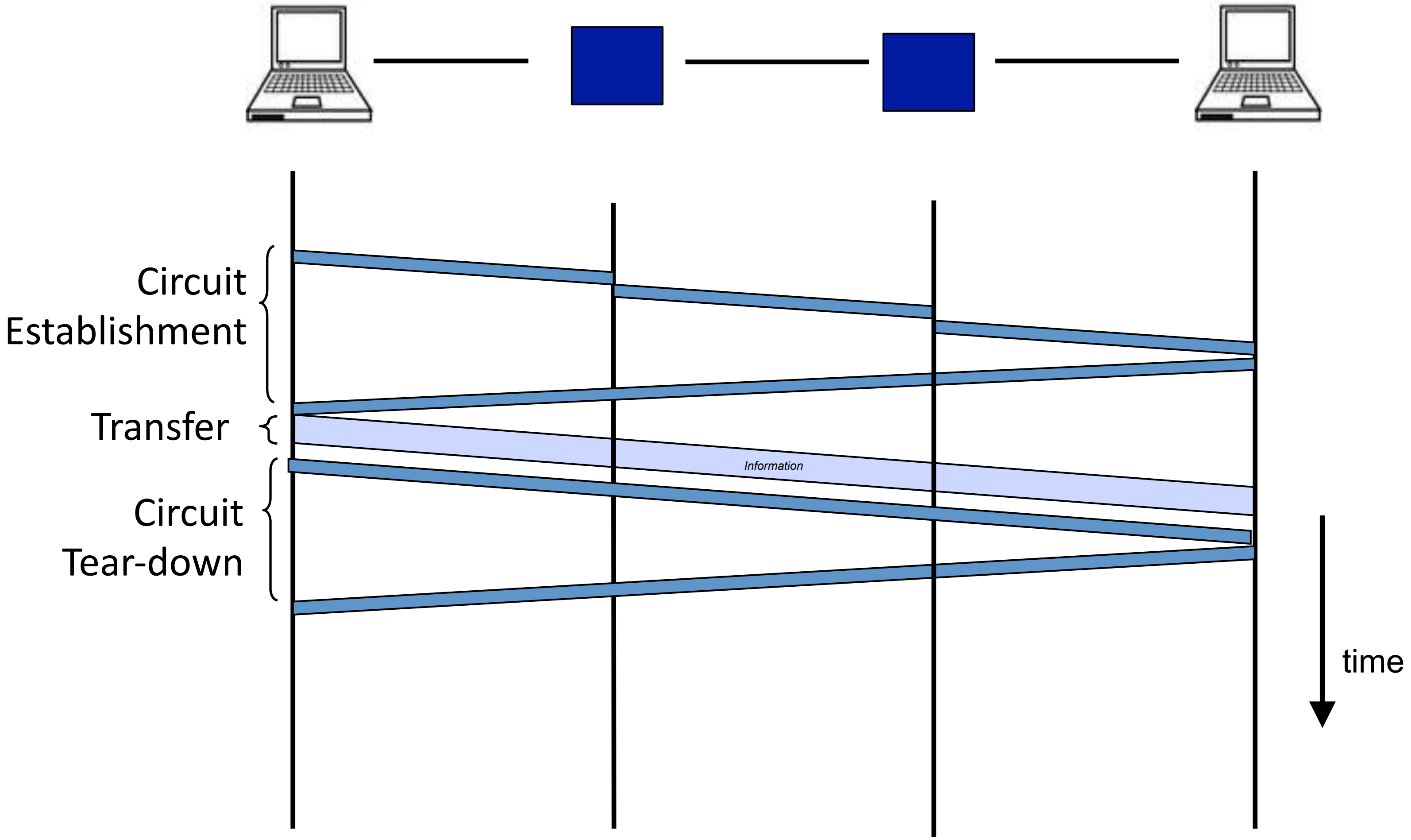
- Pros
  - guaranteed performance
  - fast transfer (once circuit is established)

- Cons
  - **wastes bandwidth if traffic is "bursty"**

Timing in Circuit Switching



Timing in Circuit Switching



Circuit Establishment

Transfer

Information

Circuit Tear-down

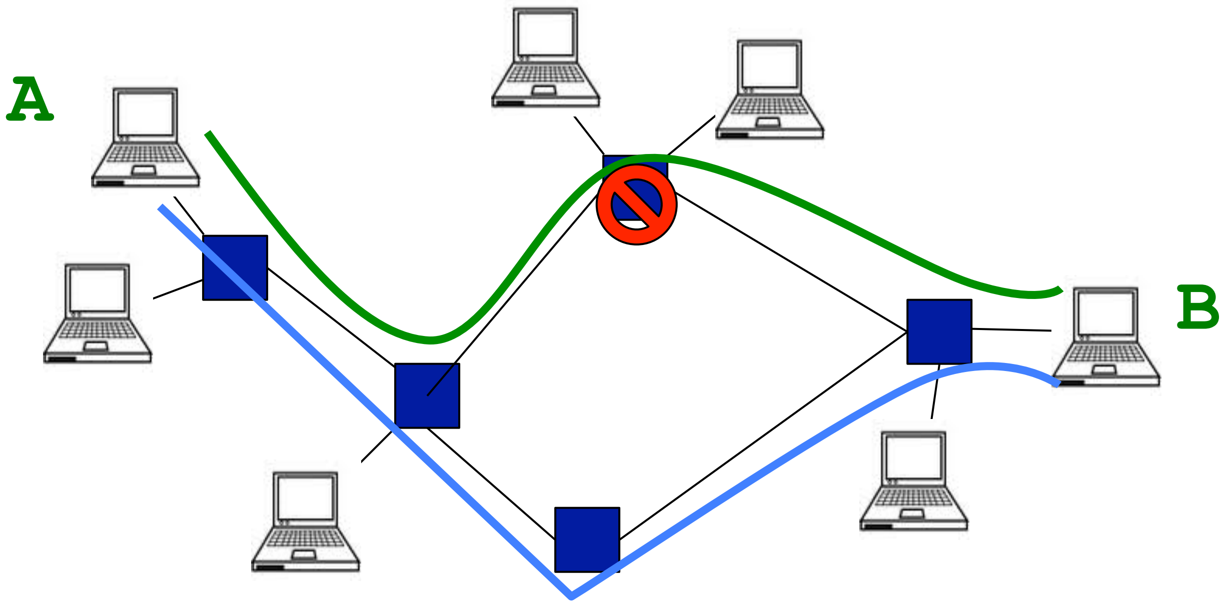
time

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 trouble”

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!

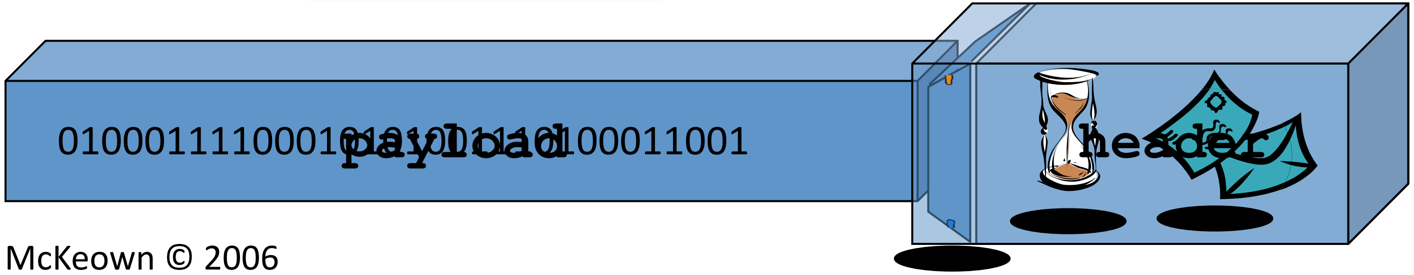
Two forms of switched networks

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

Packet Switching

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

1. Internet Address
2. Age (TTL)
3. Checksum to protect header



After Nick McKeown © 2006

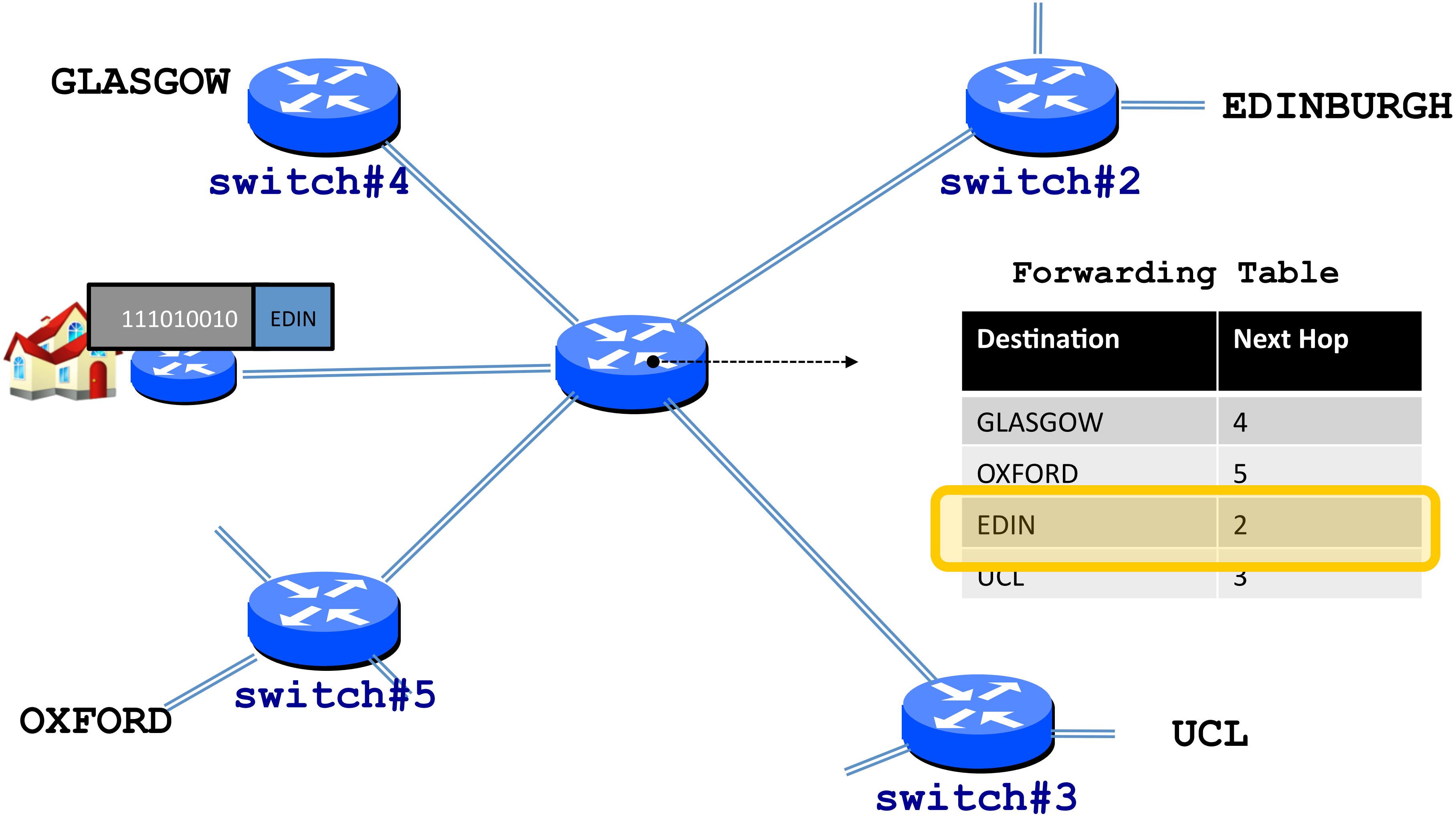
Packet Switching

- 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)

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

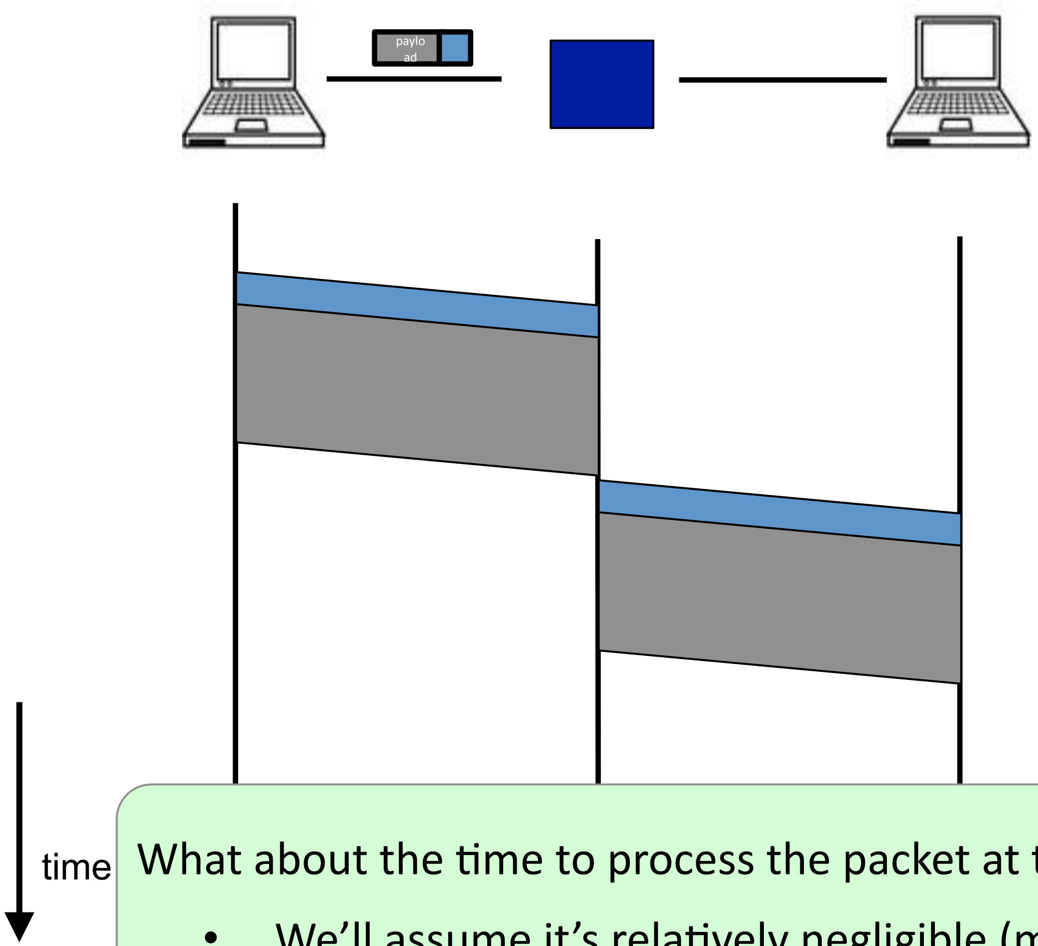
Switches forward packets



Forwarding Table

Destination	Next Hop
GLASGOW	4
OXFORD	5
EDIN	2
UCL	3

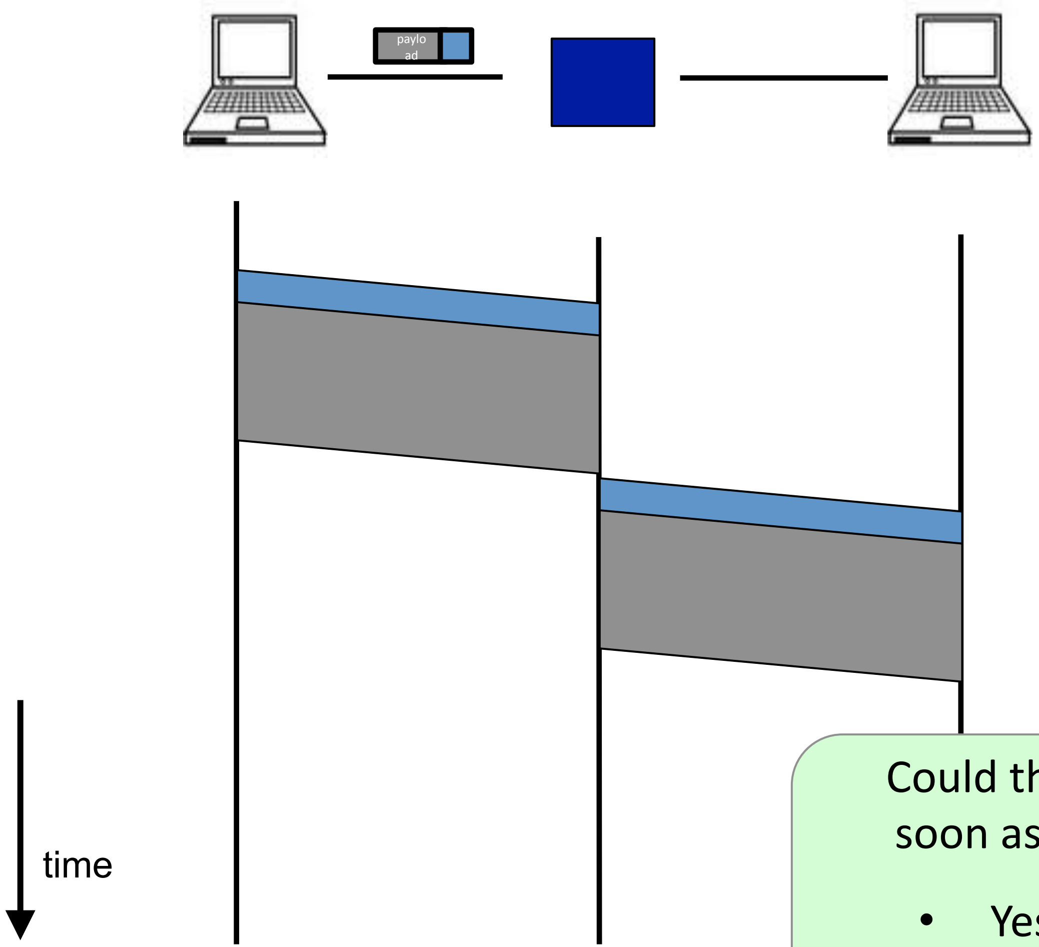
Timing in Packet Switching



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

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

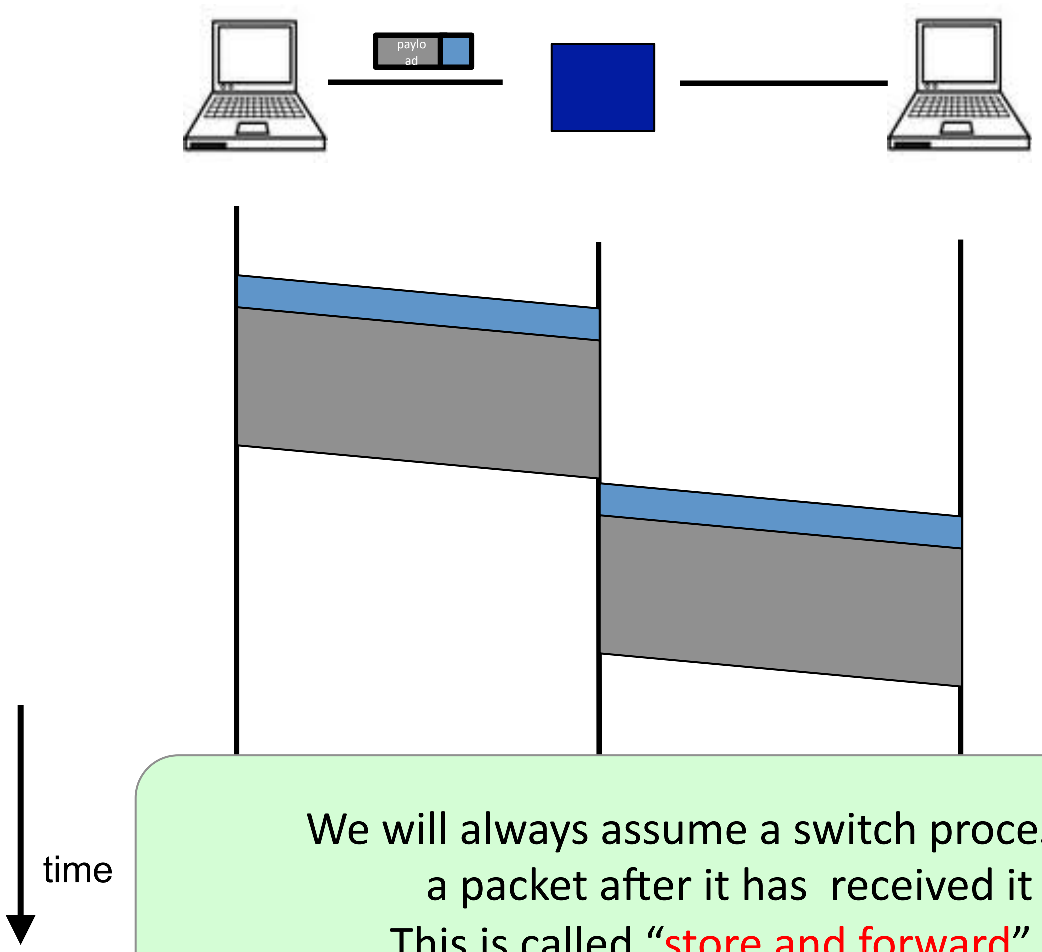
Timing in Packet Switching



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

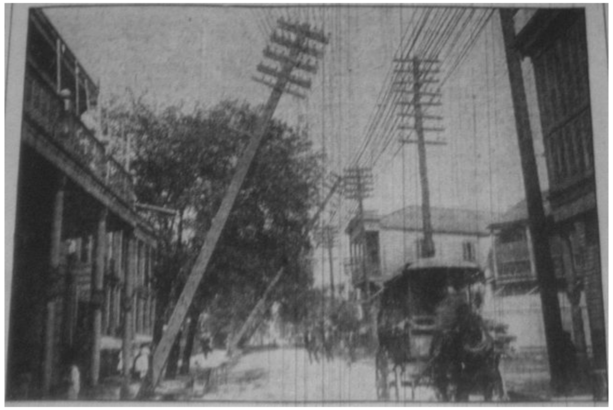
Packet Switching

- 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)

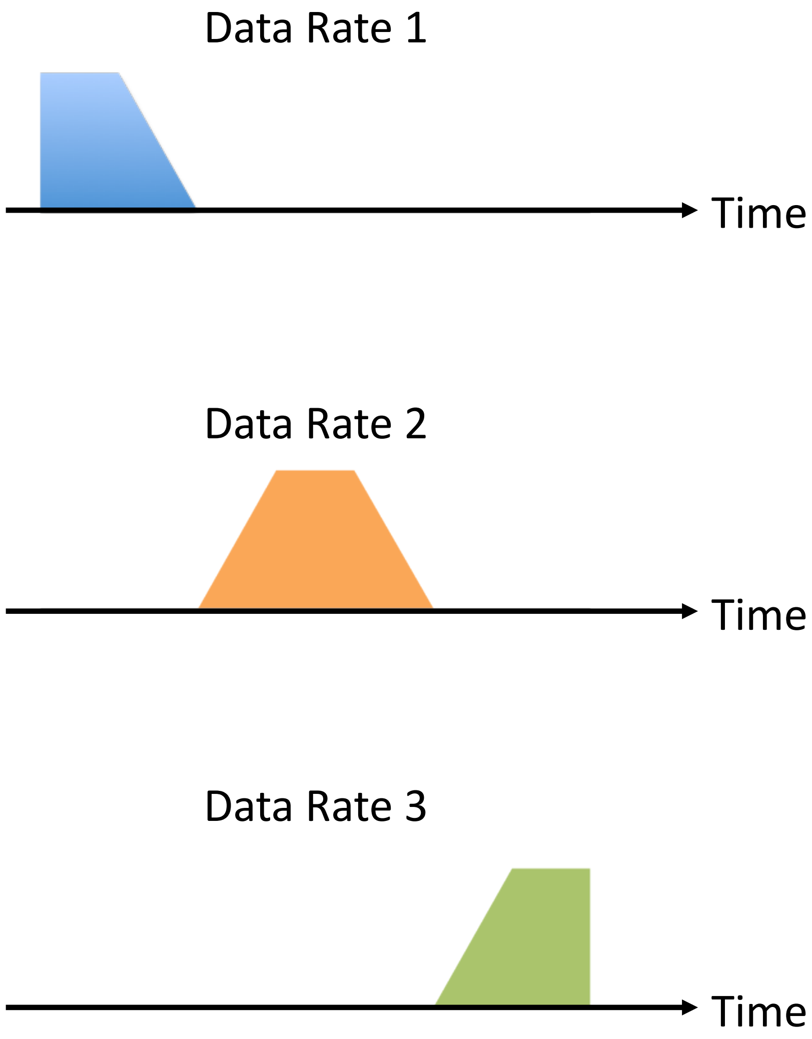
Multiplexing

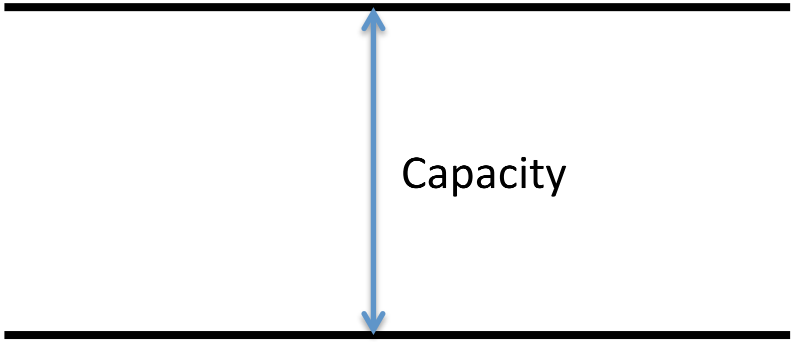


Sharing makes things efficient (cost less)

- One airplane/train for 100 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

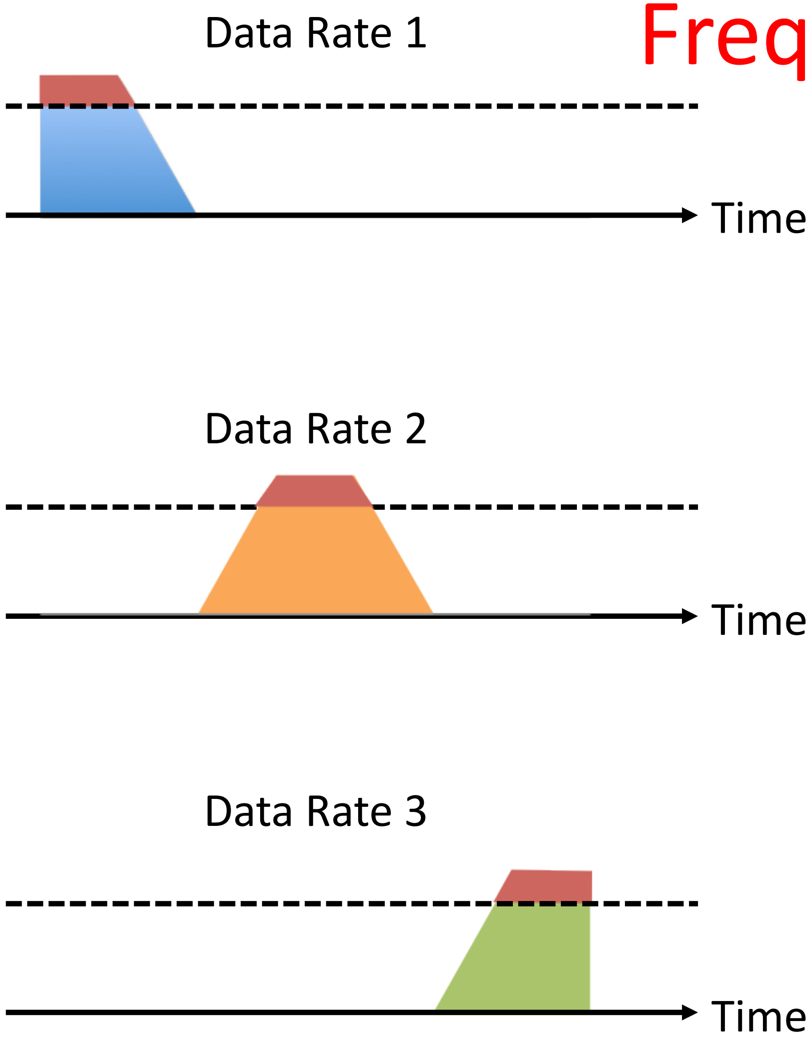
Three Flows with Bursty Traffic

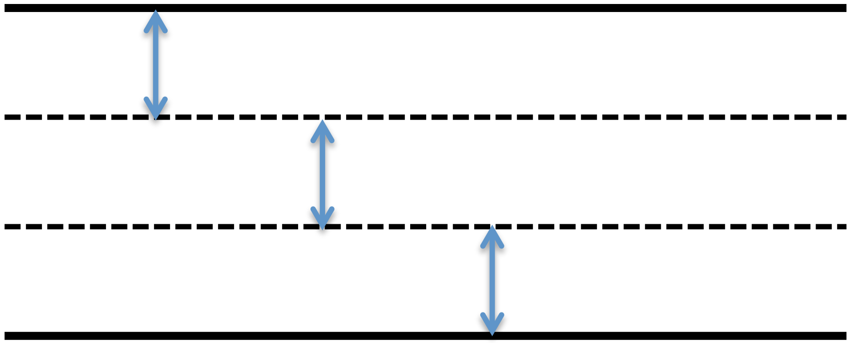




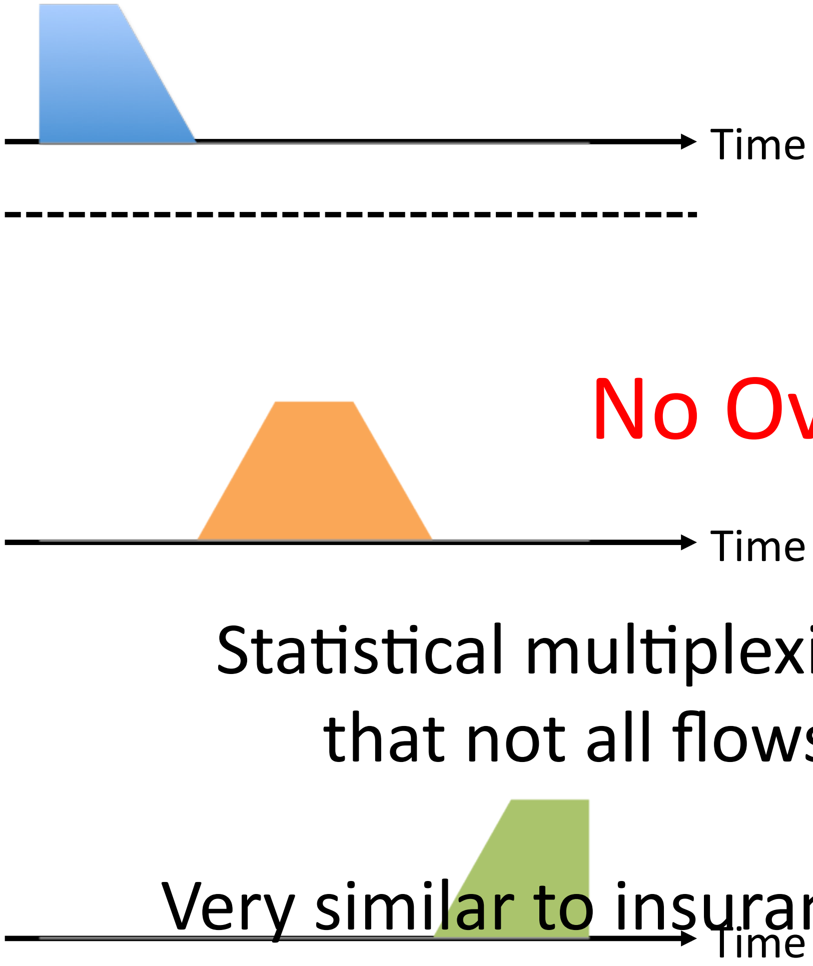
When Each Flow Gets 1/3rd of Capacity

Frequent Overloading





When Flows Share Total Capacity

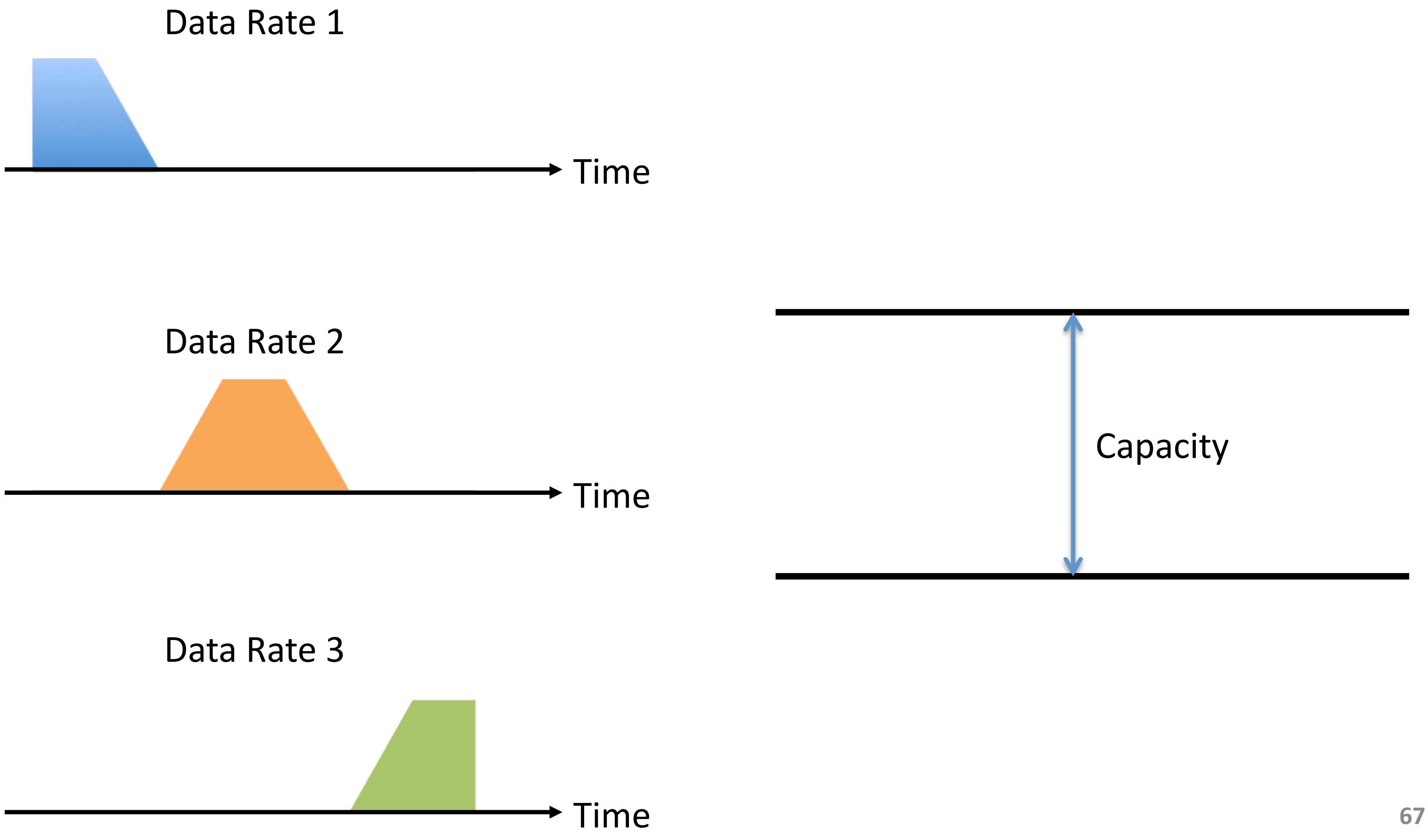


No Overloading

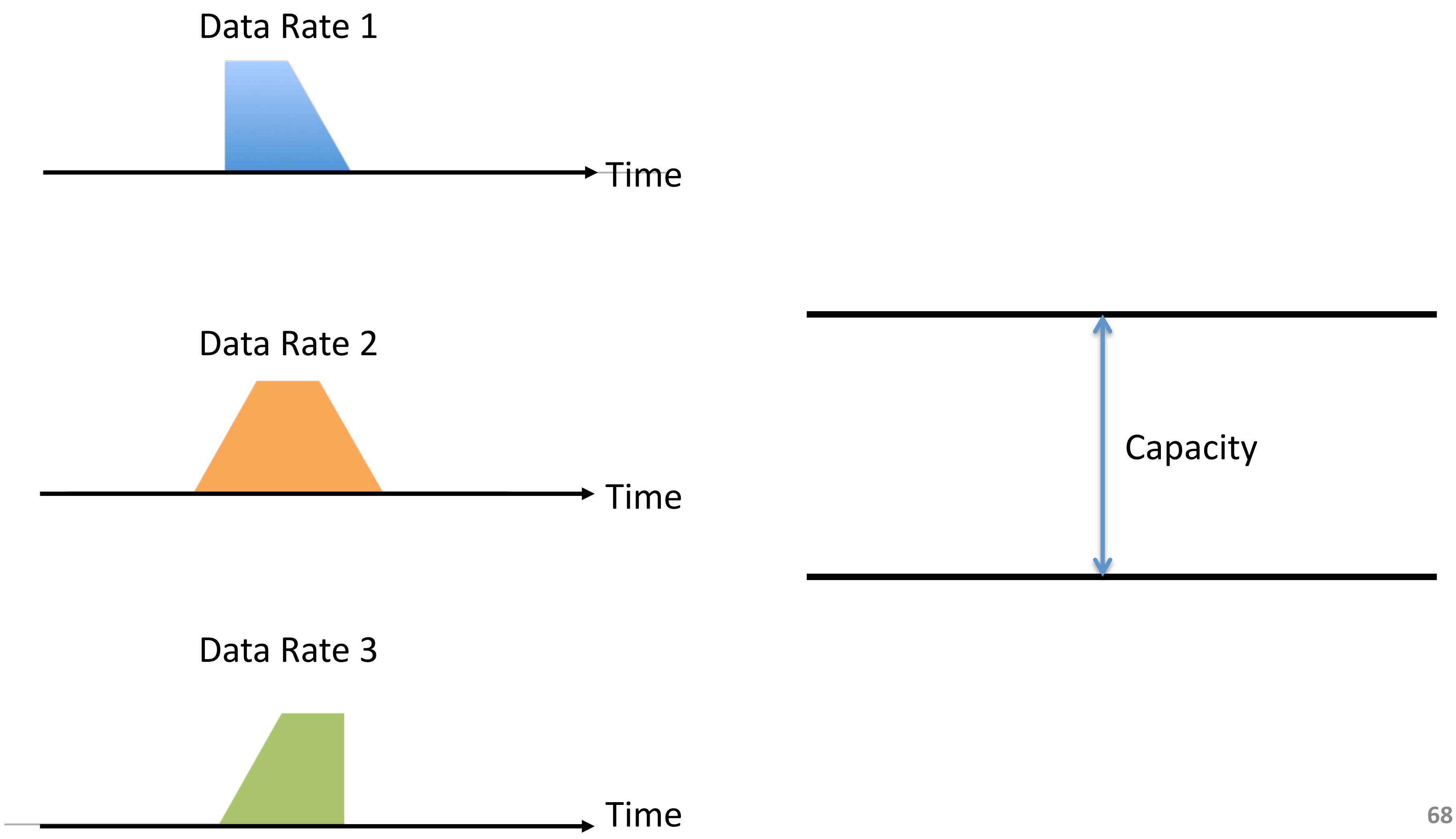
Statistical multiplexing relies on the assumption that not all flows burst at the same time.

Very similar to insurance, and has same failure case

Three Flows with Bursty Traffic



Three Flows with Bursty Traffic



Data Rate 1

Time

Data Rate 2

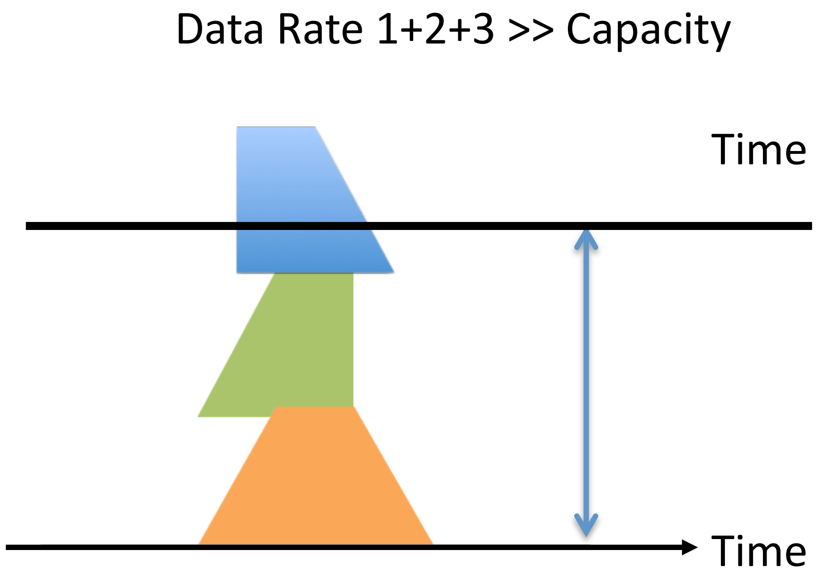
Time

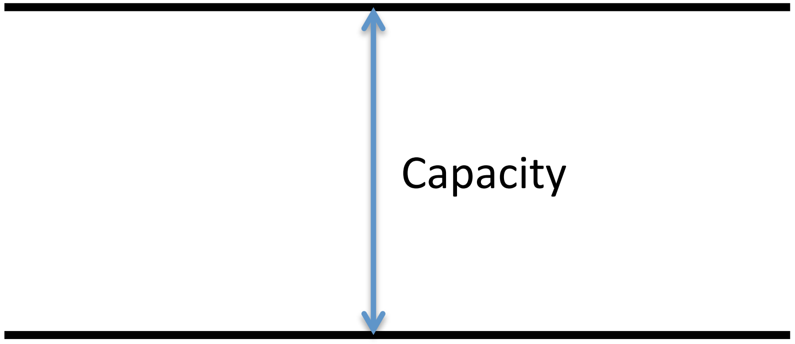
Data Rate 3

Time

Capacity

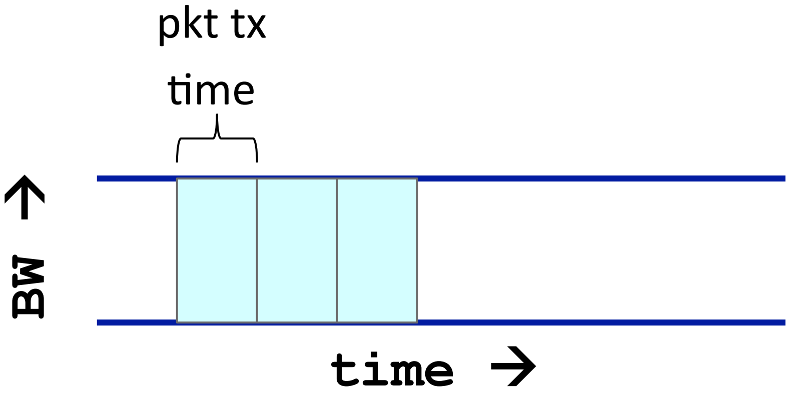
Three Flows with Bursty Traffic



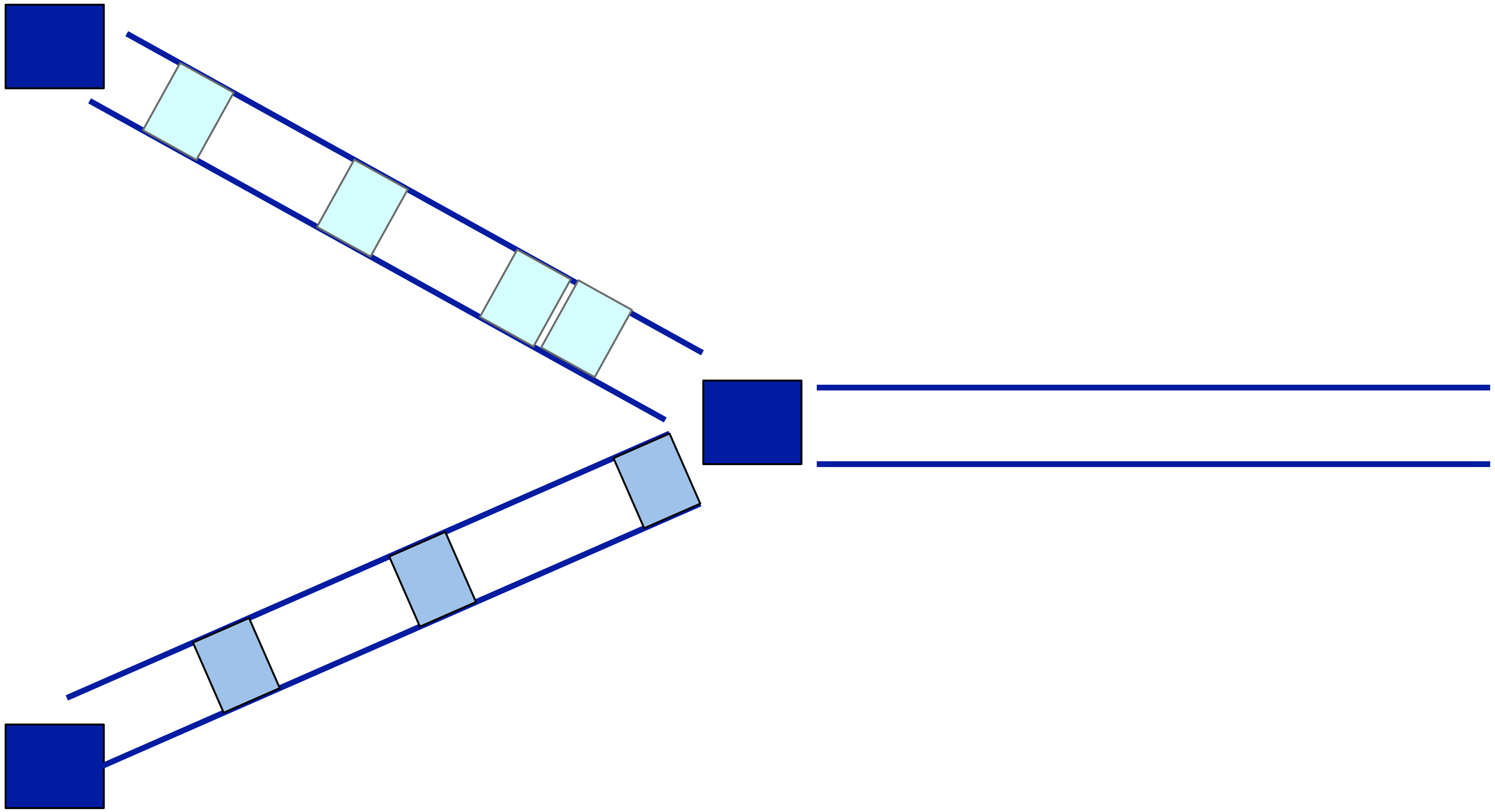


What do we do under overload?

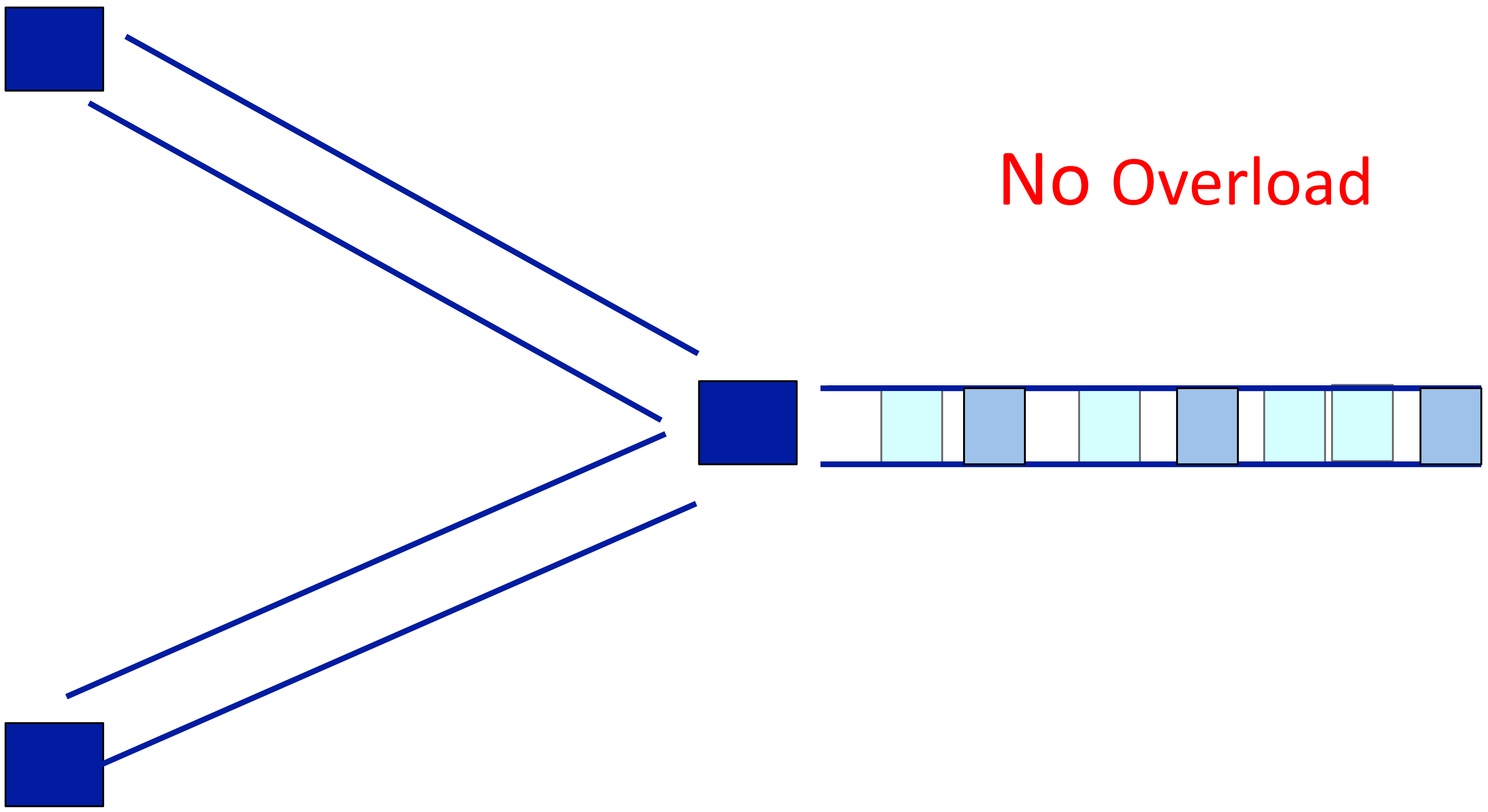
Statistical multiplexing: pipe view



Statistical multiplexing: pipe view

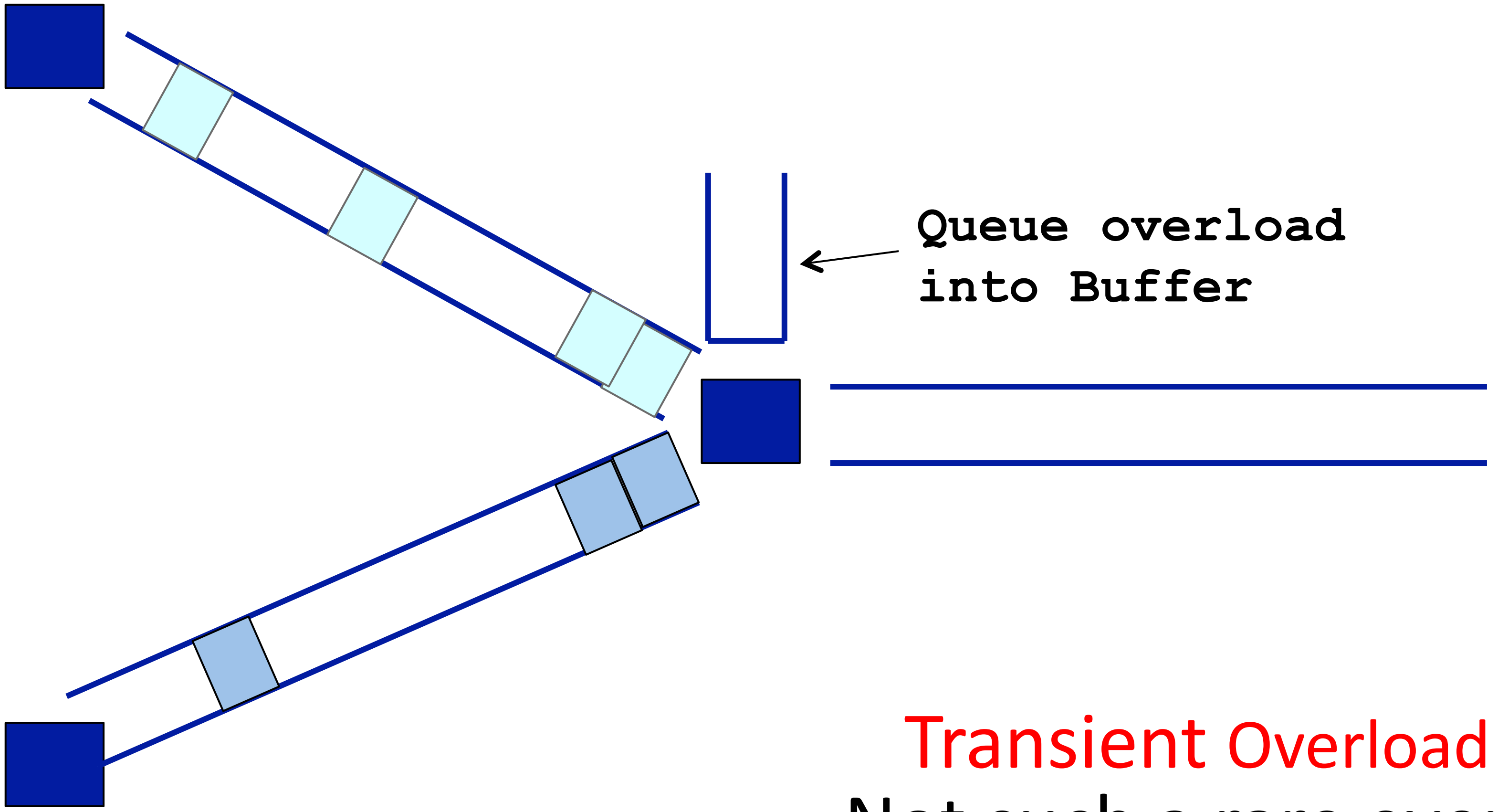


Statistical multiplexing: pipe view



No Overload

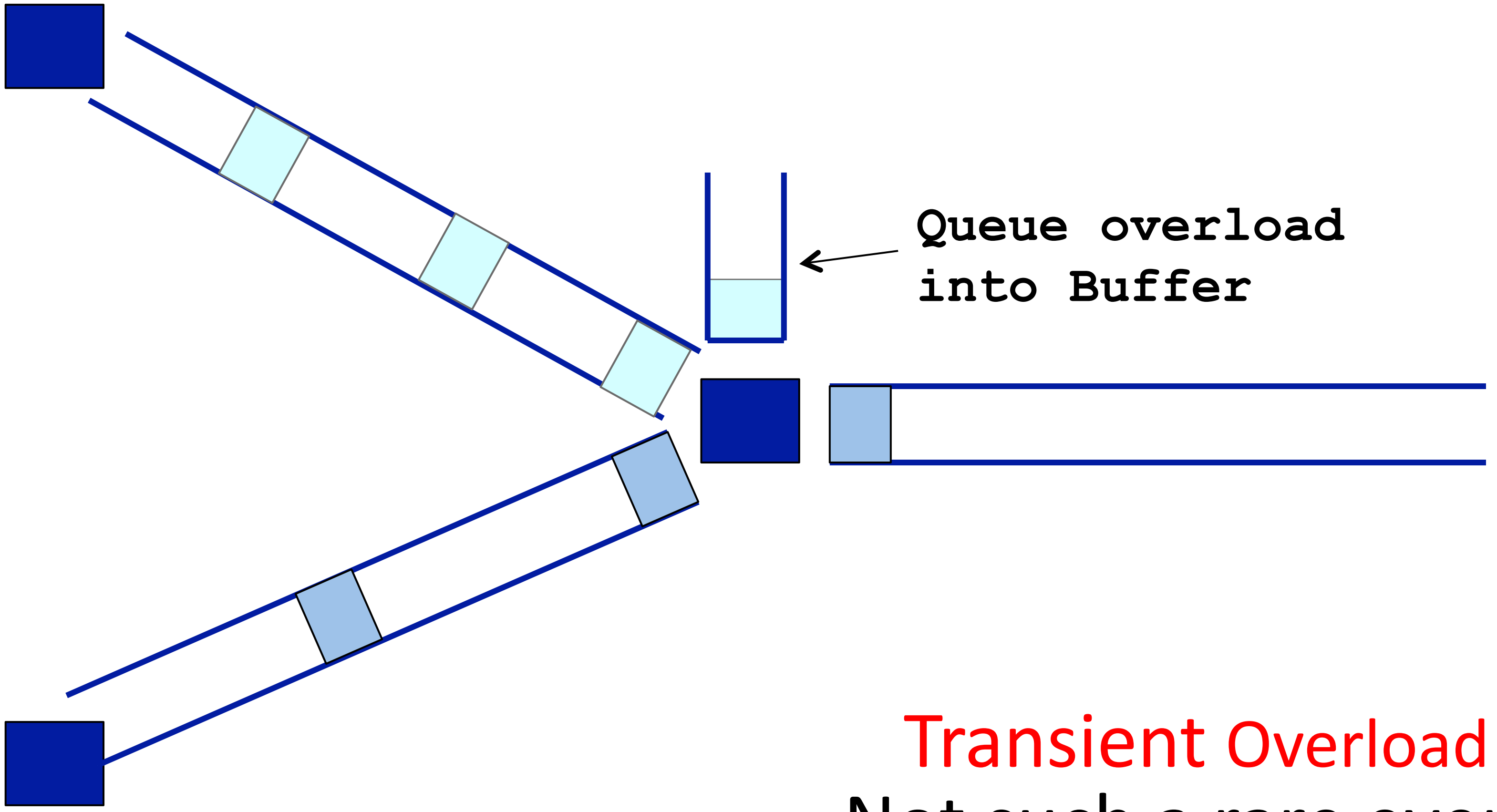
Statistical multiplexing: pipe view



Transient Overload

Not such a rare event

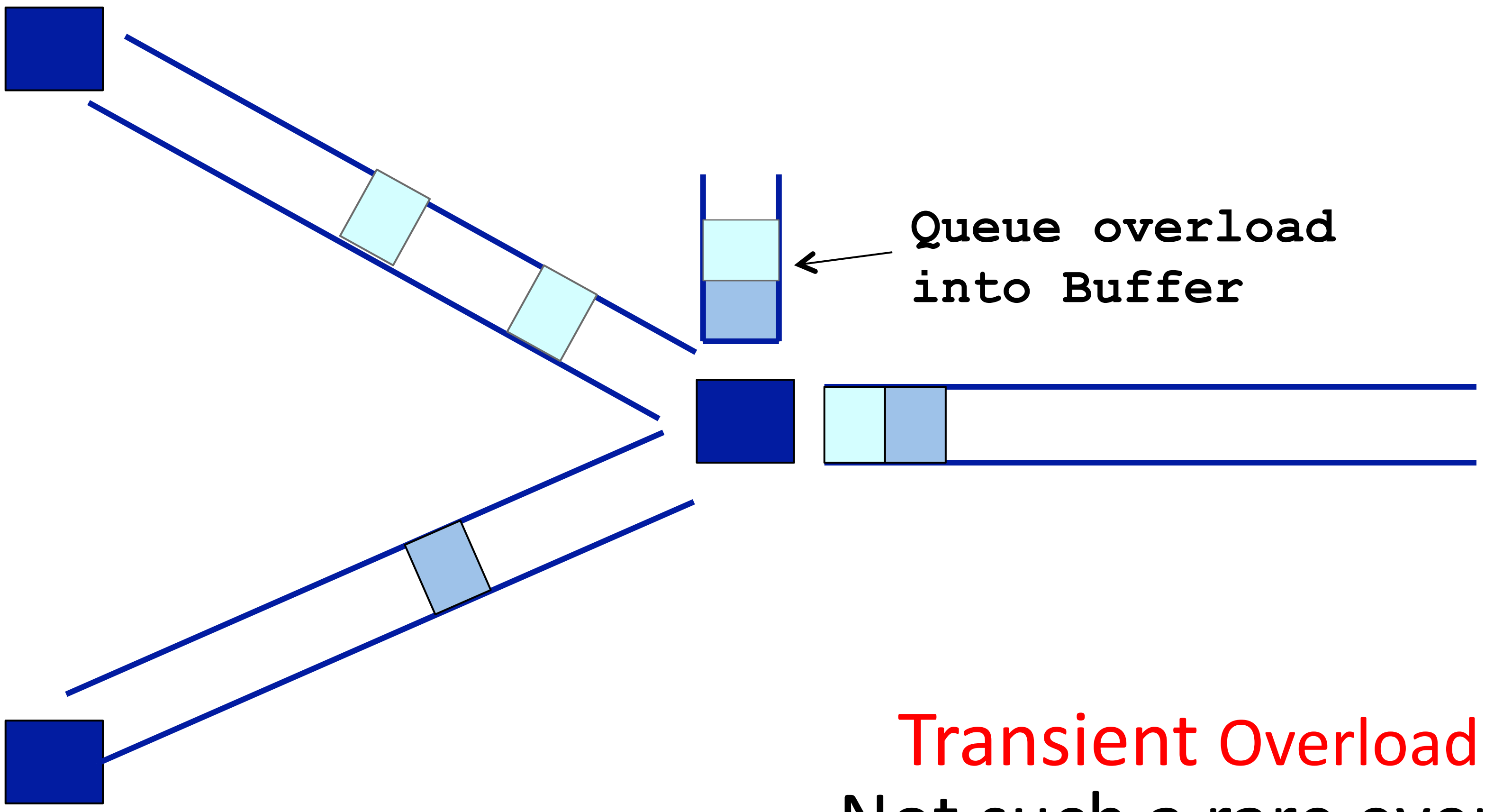
Statistical multiplexing: pipe view



Transient Overload

Not such a rare event

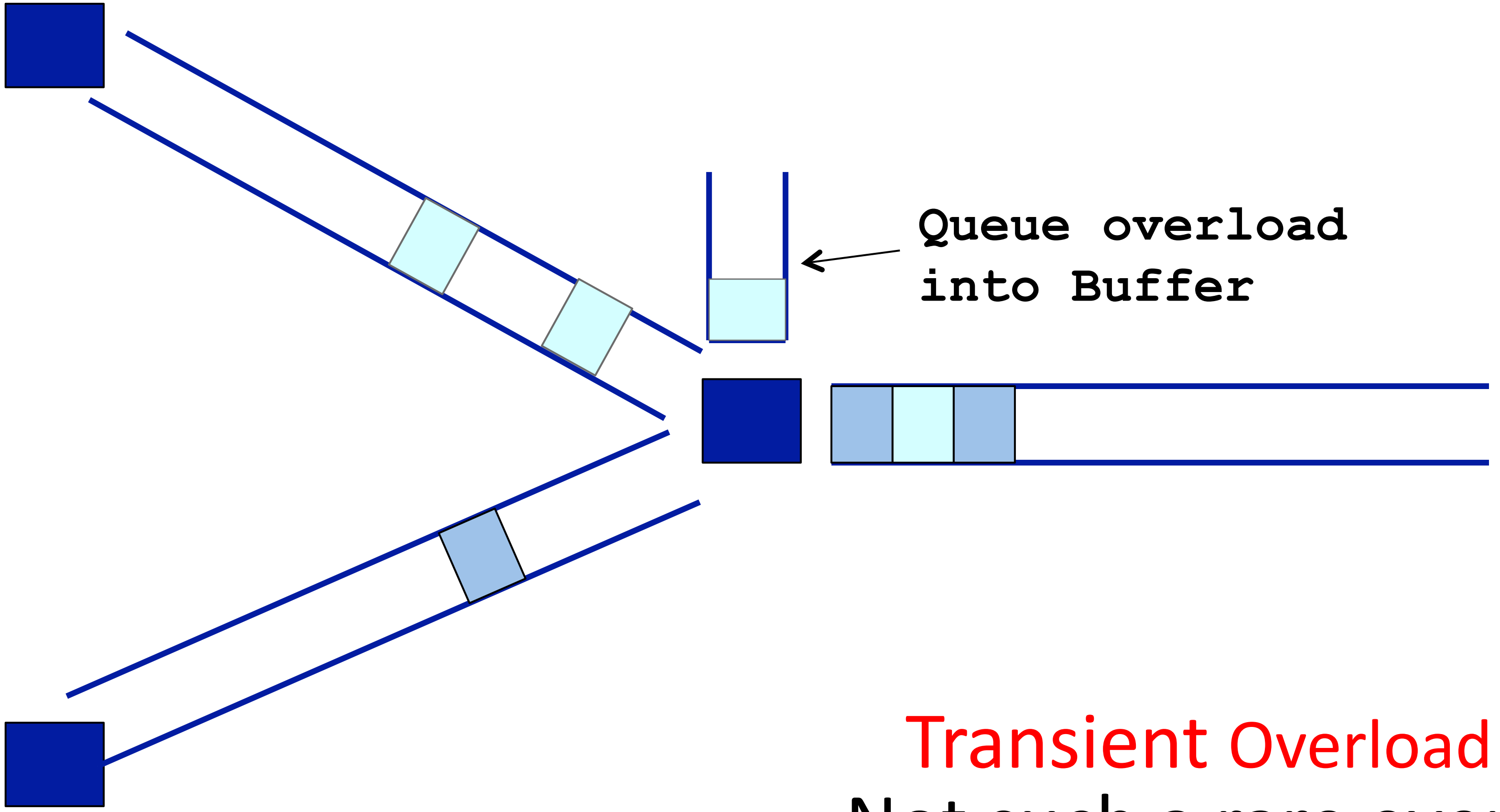
Statistical multiplexing: pipe view



Transient Overload

Not such a rare event

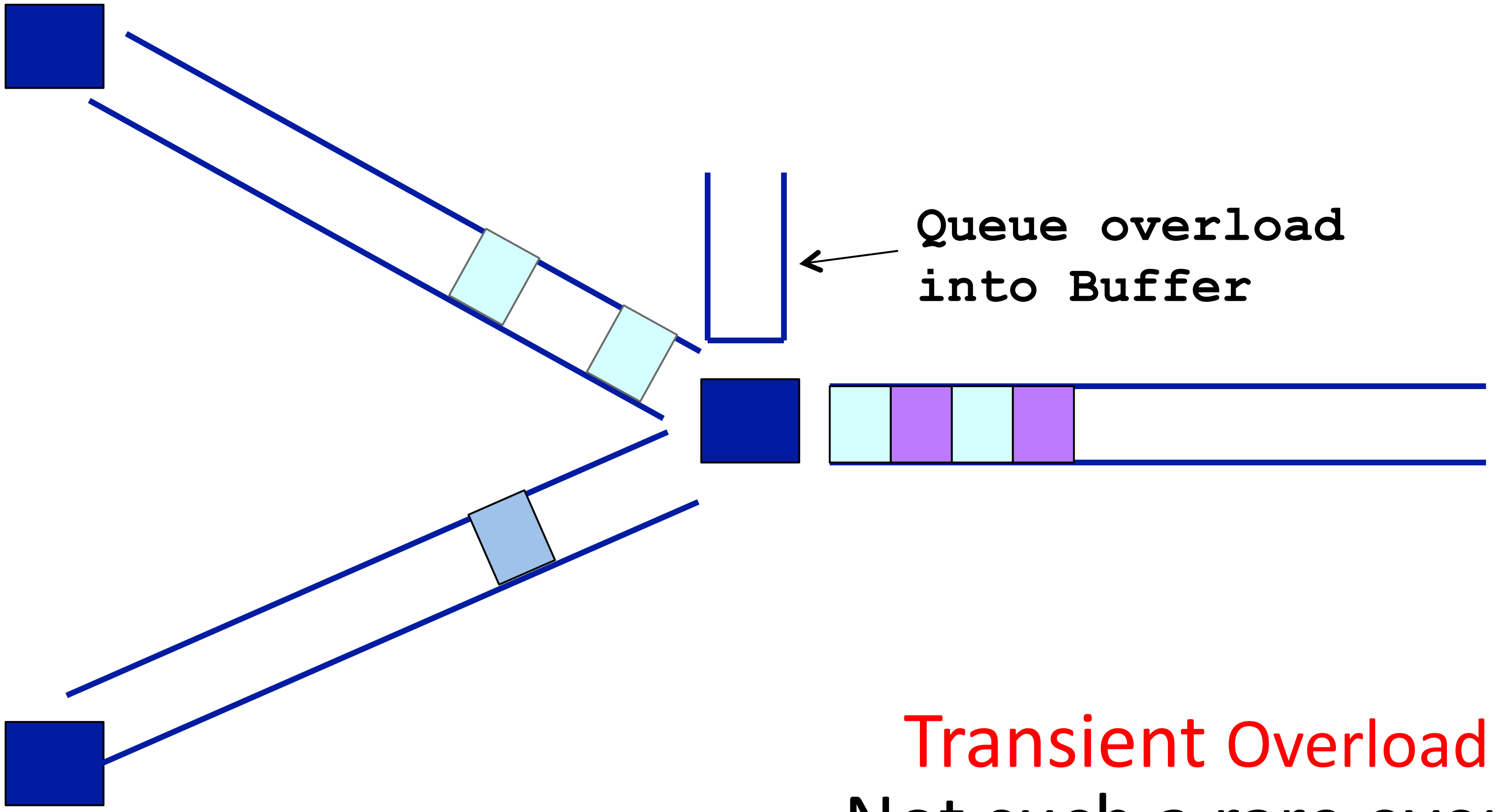
Statistical multiplexing: pipe view



Transient Overload

Not such a rare event

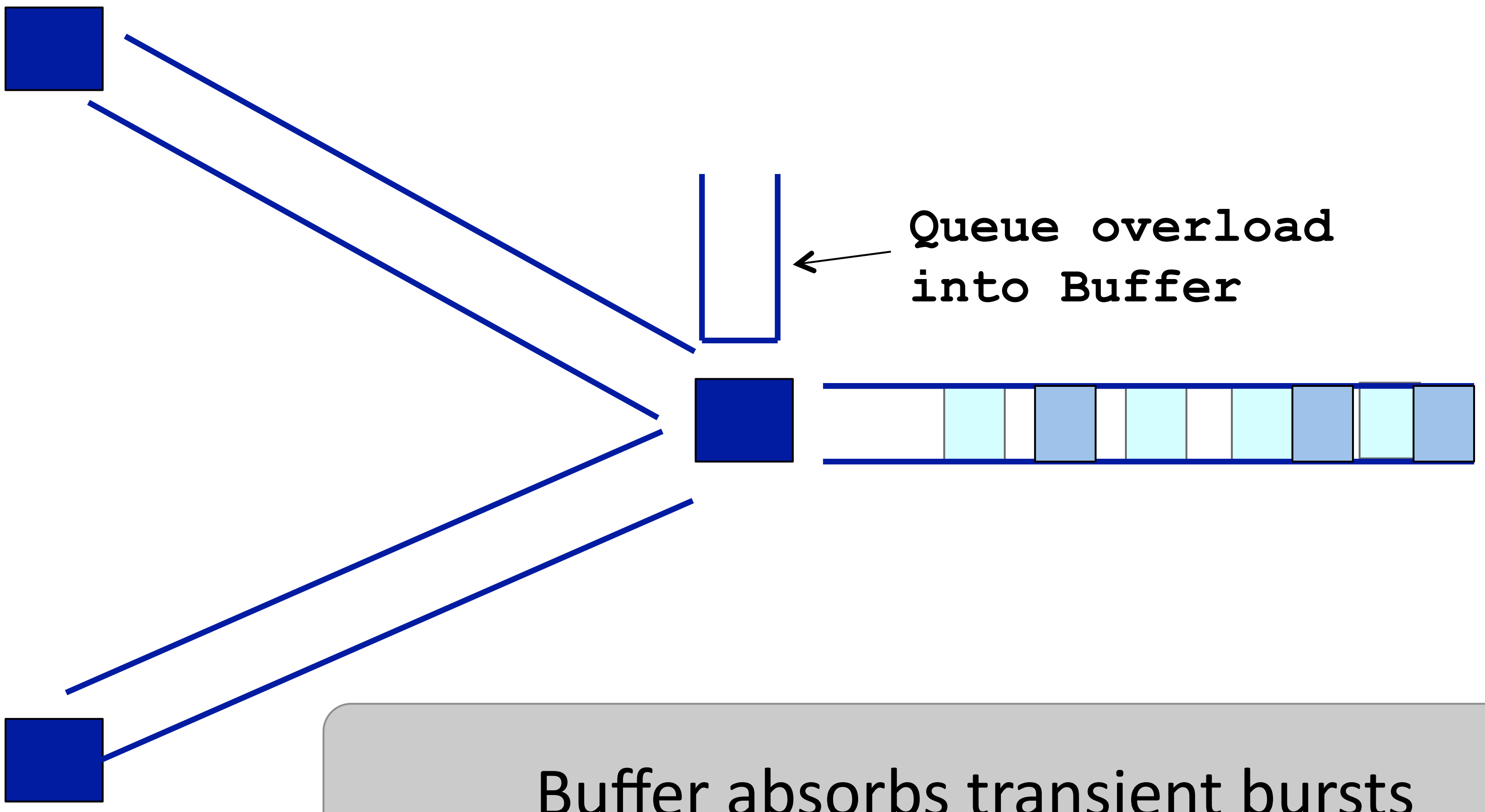
Statistical multiplexing: pipe view



Transient Overload

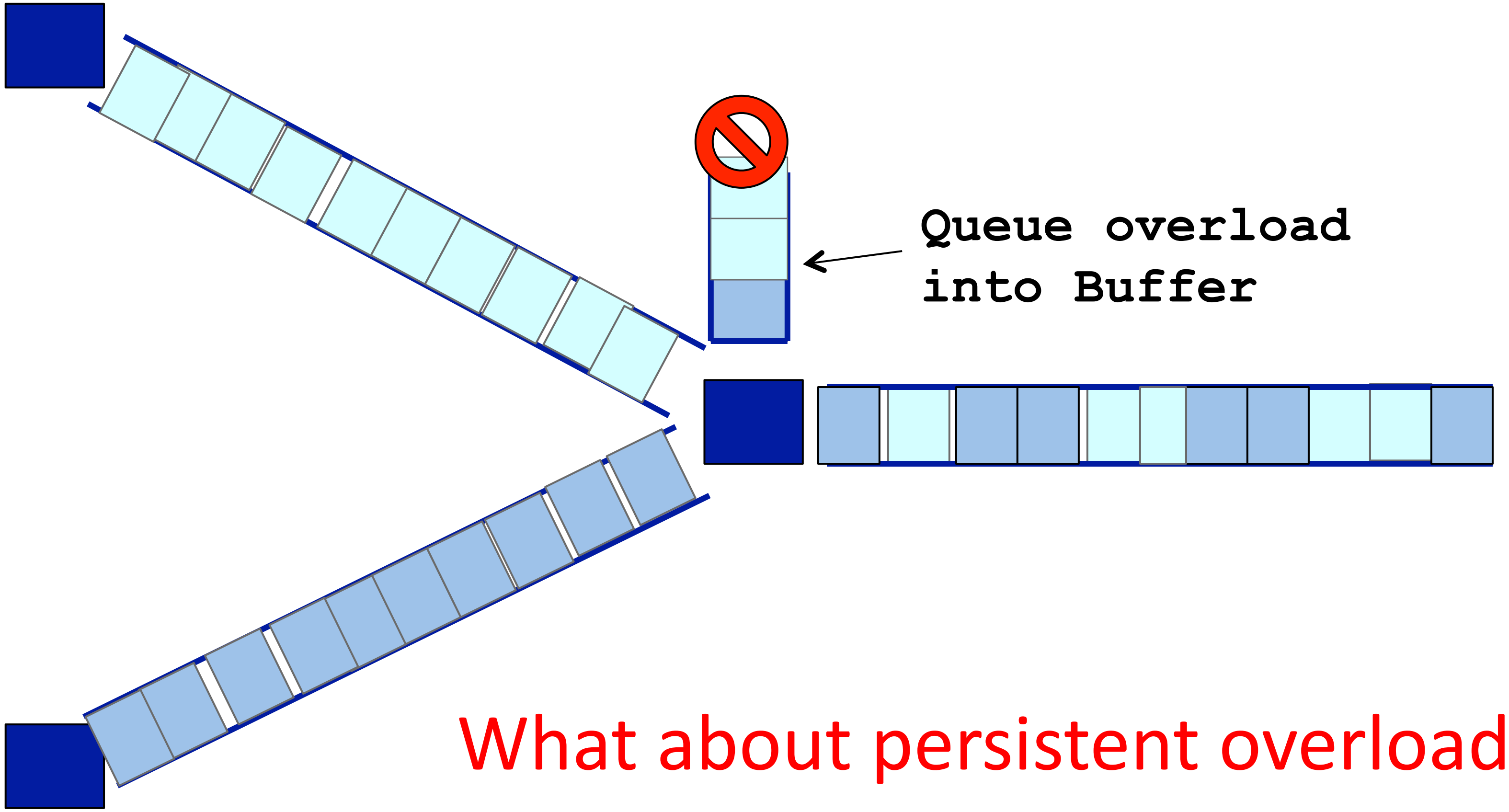
Not such a rare event

Statistical multiplexing: pipe view



Buffer absorbs transient bursts

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 muxing)

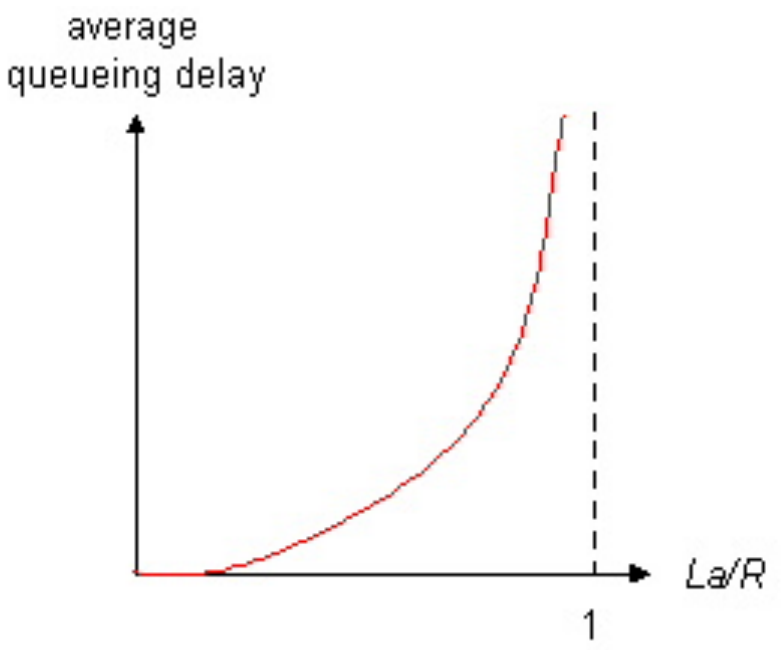
  packet delay = transmission delay + propagation delay + queuing delay

- 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

Queuing delay

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

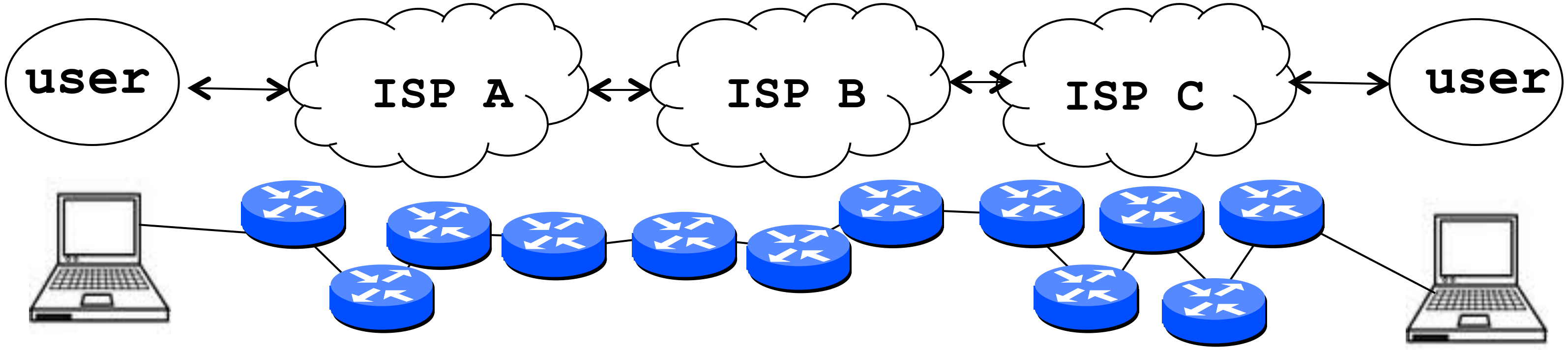


traffic intensity = La/R

- ❑ 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
  - >18,000 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 munnari.oz.au

(tracepath on pwf is similar)

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

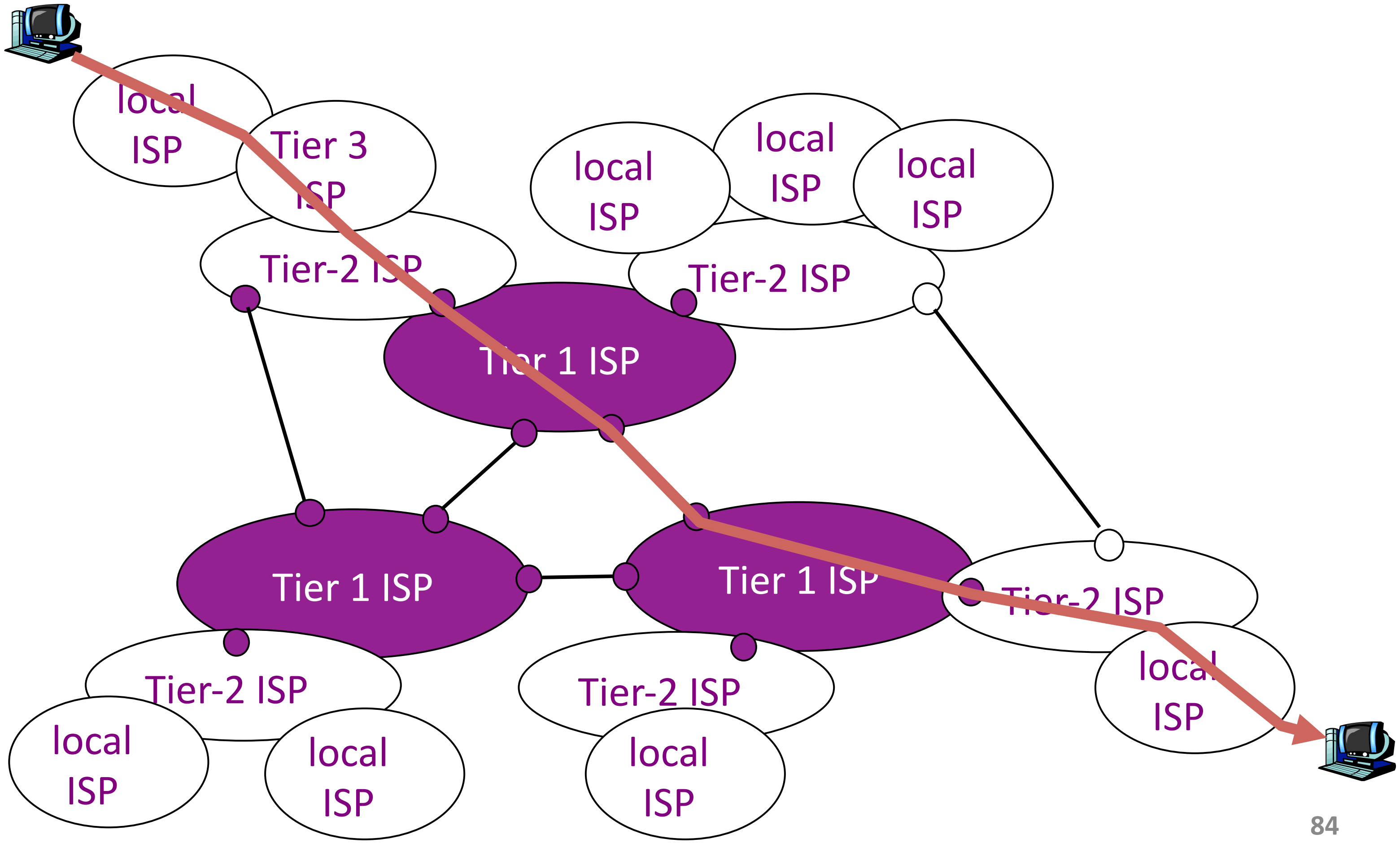
```
traceroute munnari.oz.au
traceroute to munnari.oz.au (202.29.151.3), 30 hops max, 60 byte packets
 1  gatwick.net.cl.cam.ac.uk (128.232.32.2)  0.416 ms  0.384 ms  0.427 ms
 2  cl-sby.route-nwest.net.cam.ac.uk (193.60.89.9)  0.393 ms  0.440 ms  0.494 ms
 3  route-nwest.route-mill.net.cam.ac.uk (192.84.5.137)  0.407 ms  0.448 ms  0.501 ms
 4  route-mill.route-enet.net.cam.ac.uk (192.84.5.94)  1.006 ms  1.091 ms  1.163 ms
 5  xe-11-3-0.camb-rbr1.eastern.ja.net (146.97.130.1)  0.300 ms  0.313 ms  0.350 ms
 6  ae24.lowdss-sbr1.ja.net (146.97.37.185)  2.679 ms  2.664 ms  2.712 ms
 7  ae28.londhx-sbr1.ja.net (146.97.33.17)  5.955 ms  5.953 ms  5.901 ms
 8  janet.mx1.lon.uk.geant.net (62.40.124.197)  6.059 ms  6.066 ms  6.052 ms
 9  ae0.mx1.par.fr.geant.net (62.40.98.77)  11.742 ms  11.779 ms  11.724 ms
10  ae1.mx1.mad.es.geant.net (62.40.98.64)  27.751 ms  27.734 ms  27.704 ms
11  mb-so-02-v4.bb.tein3.net (202.179.249.117)  138.296 ms  138.314 ms  138.282 ms
12  sg-so-04-v4.bb.tein3.net (202.179.249.53)  196.303 ms  196.293 ms  196.264 ms
13  th-pr-v4.bb.tein3.net (202.179.249.66)  225.153 ms  225.178 ms  225.196 ms
14  pyt-thairen-to-02-bdr-pyt.uni.net.th (202.29.12.10)  225.163 ms  223.343 ms  223.363 ms
15  202.28.227.126 (202.28.227.126)  241.038 ms  240.941 ms  240.834 ms
16  202.28.221.46 (202.28.221.46)  287.252 ms  287.306 ms  287.282 ms
17  * * *
18  * * *
19  * * *
20  coe-gw.psu.ac.th (202.29.149.70)  241.681 ms  241.715 ms  241.680 ms
21  munnari.OZ.AU (202.29.151.3)  241.610 ms  241.636 ms  241.537 ms
```

trans-continent link

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

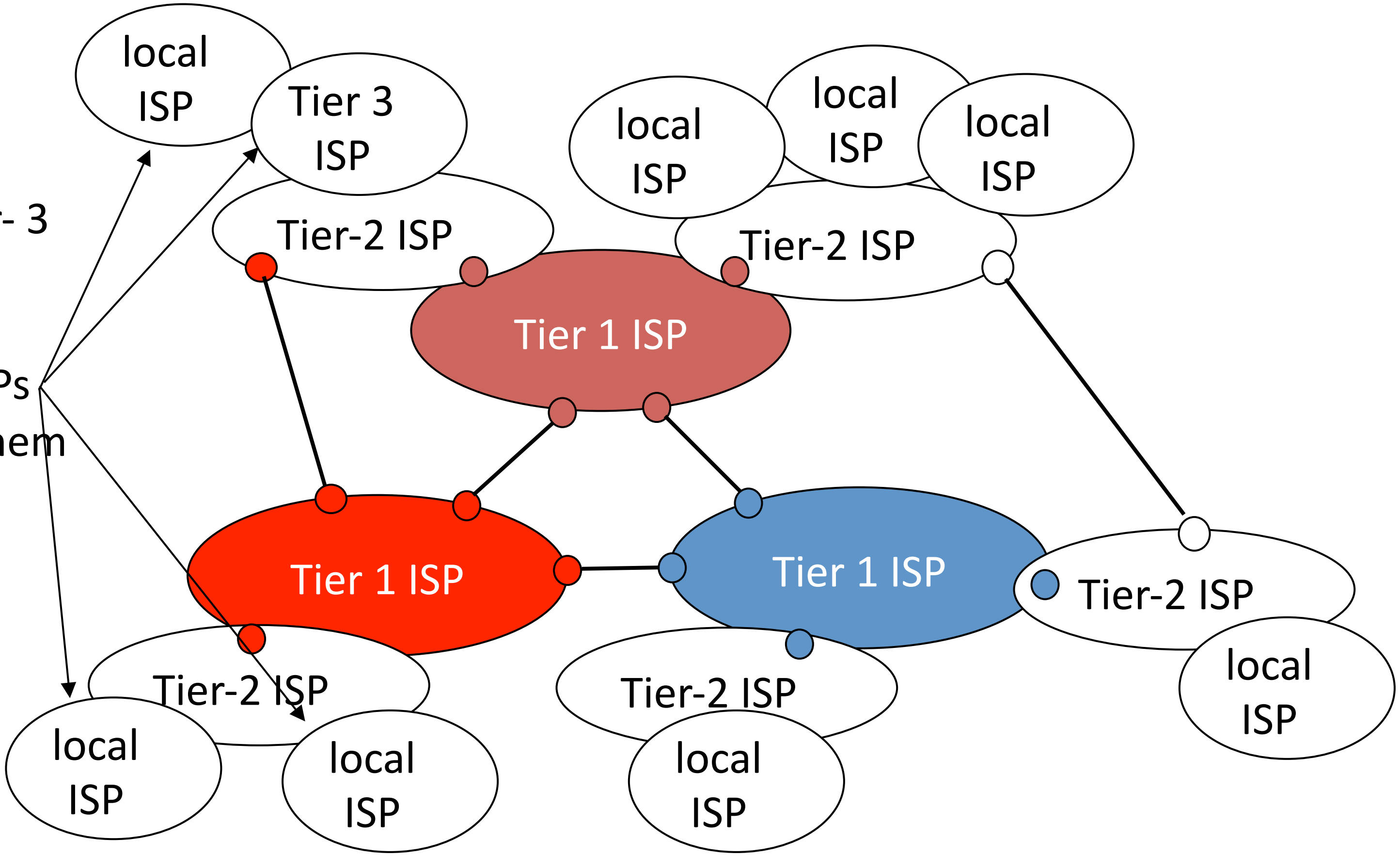
Internet structure: network of networks

- a packet passes through many networks!



Internet structure: network of networks

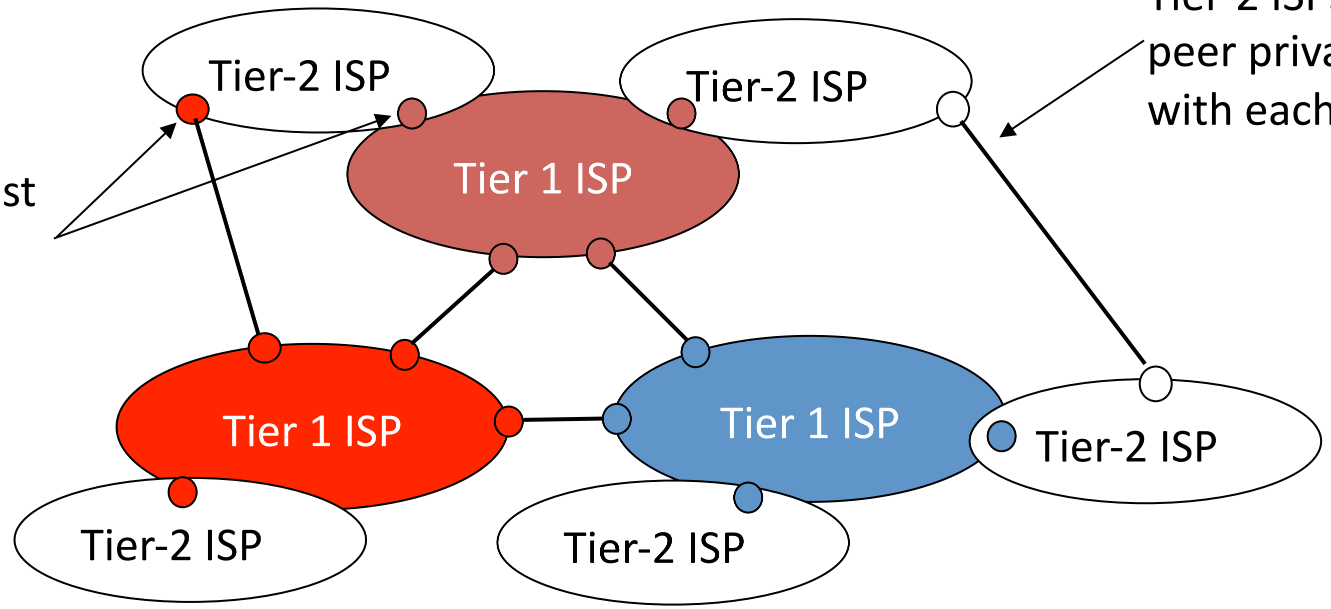
- “Tier-3” ISPs and local ISPs
  - last hop (“access”) network (closest to end systems)



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



Tier-2 ISP pays tier-1 ISP for connectivity to rest of Internet

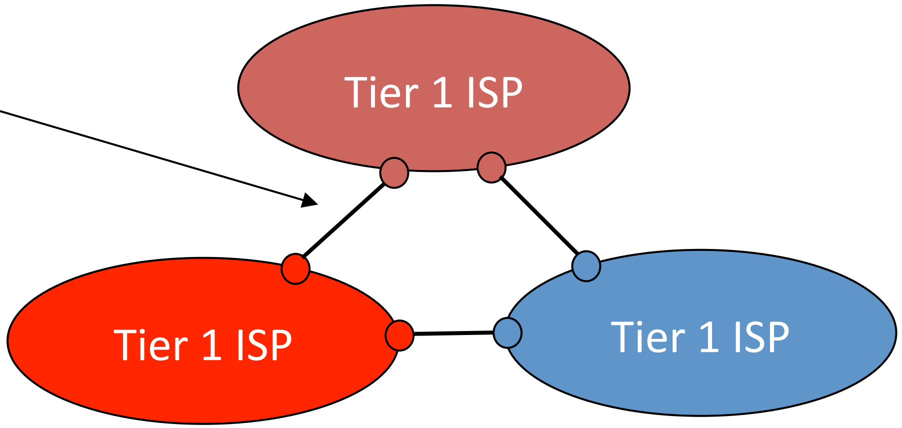
❑ tier-2 ISP is *customer* of tier-1 provider

Tier-2 ISPs also peer privately with each other.

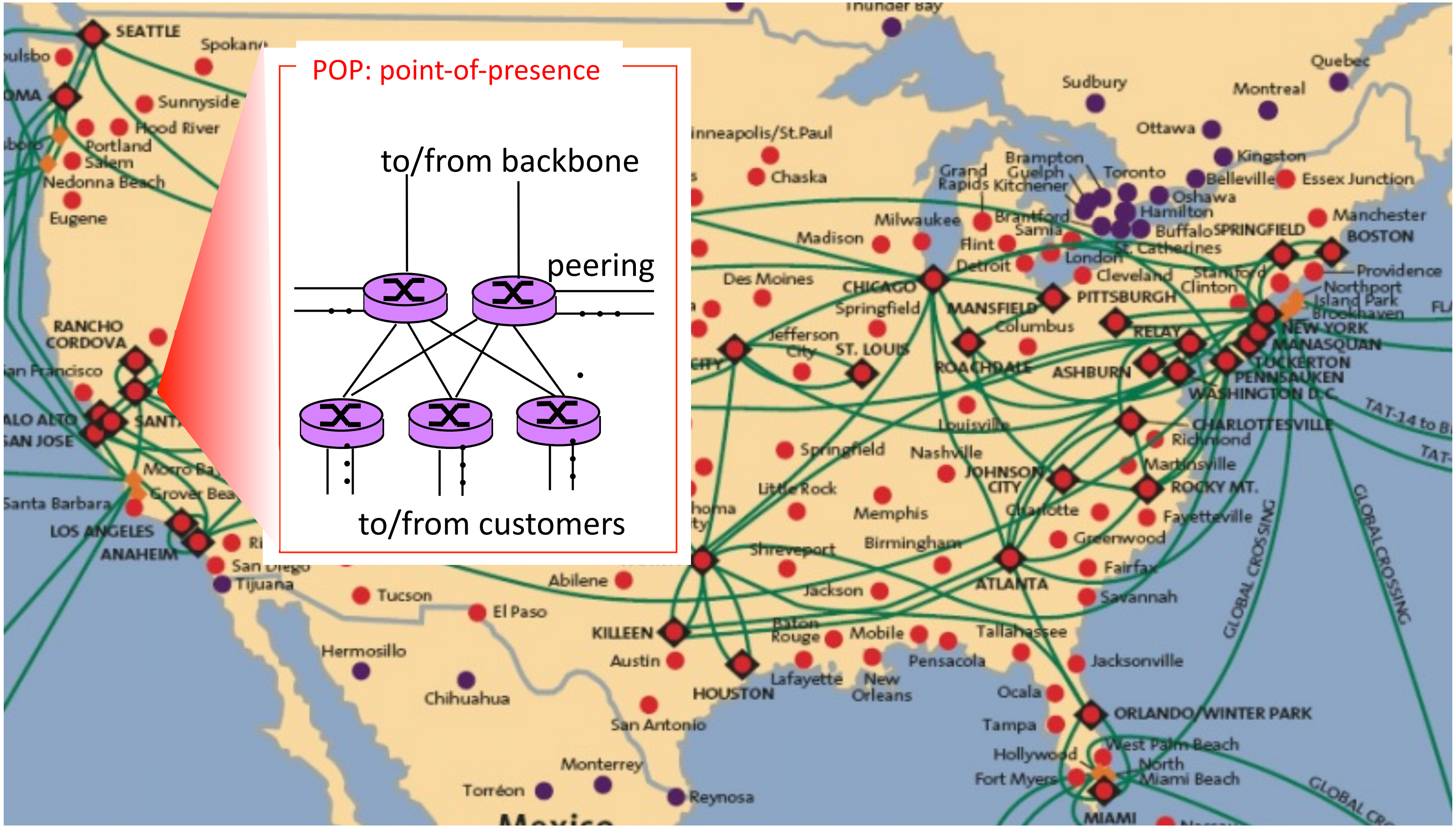
Internet structure: network of networks

- 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 providers interconnect (peer) privately



Tier-1 ISP: e.g., Sprint



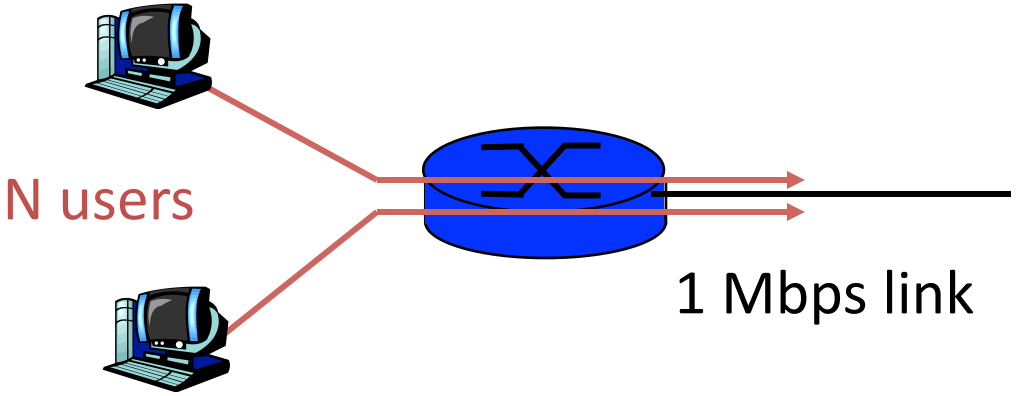
Packet Switching

- 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

HINT: Binomial Distribution

- *circuit-switching:*
  - 10 users
- *packet switching:*
  - with 35 users, probability > 10 active at same time is less than .0004

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

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’

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

Topic 2 – Foundations and Architecture

- 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

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 build on top of (“layered over”) 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

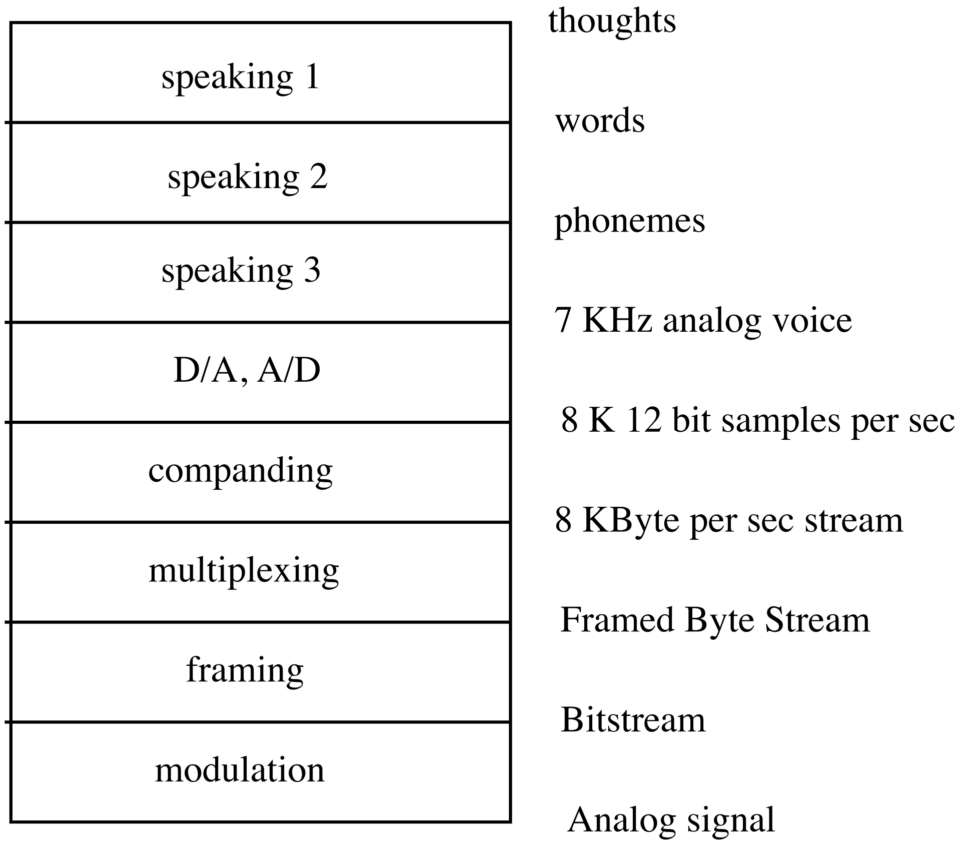
Network System Modularity

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!



thoughts

speaking 1

words

speaking 2

phonemes

speaking 3

7 KHz analog voice

D/A, A/D

8 K 12 bit samples per sec

companding

8 KByte per sec stream

multiplexing

Framed Byte Stream

framing

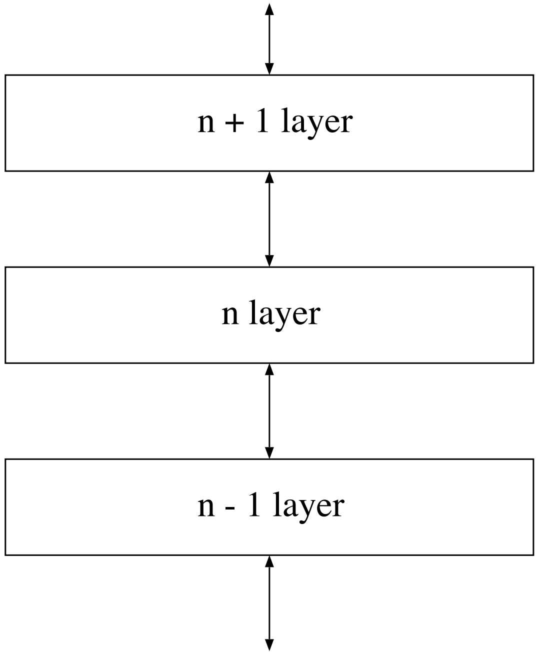
Bitstream

modulation

Analog signal

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

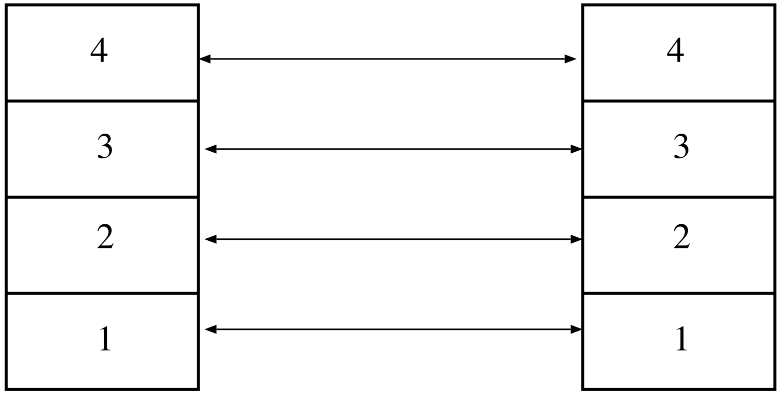
Entity – a *thing* (an independent existence)

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

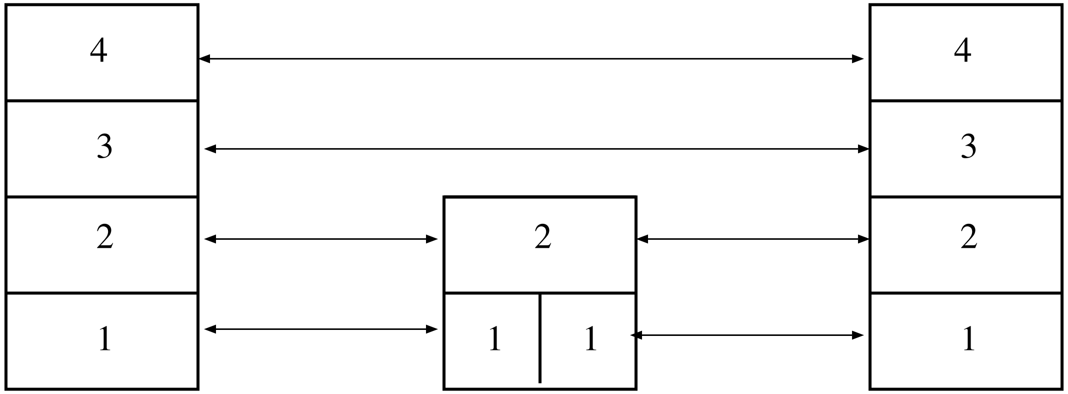
Entities usually do something useful

- 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

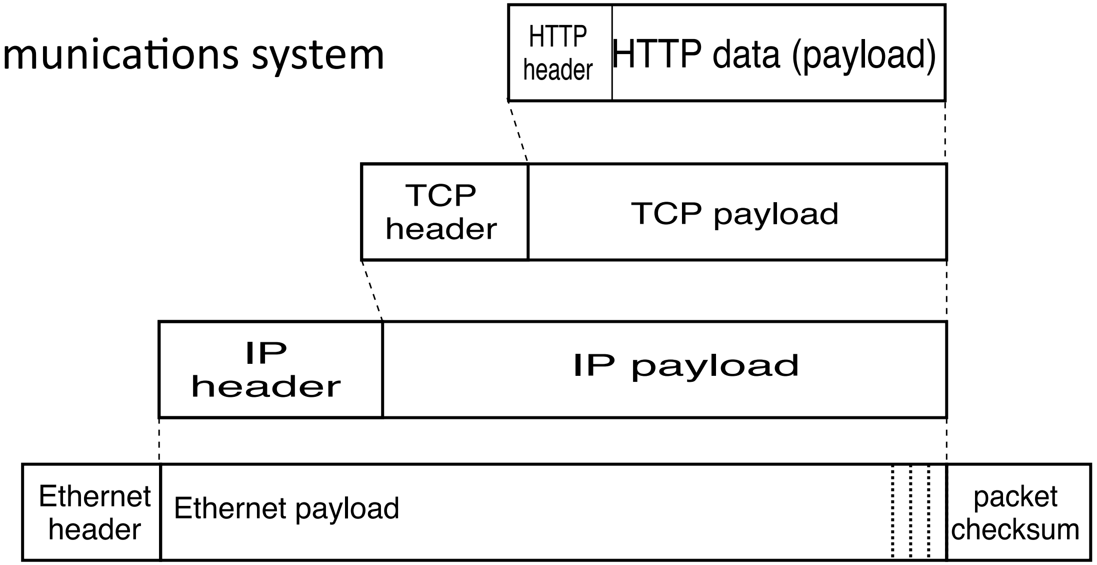
- Such embedding can be considered a form of layering
- Higher layer information is generated by stripping off headers and trailers of the current layer
- eg an IP entity only looks at the IP headers

BUT embedding is not the only form of layering

Layering is to help understand a communications system

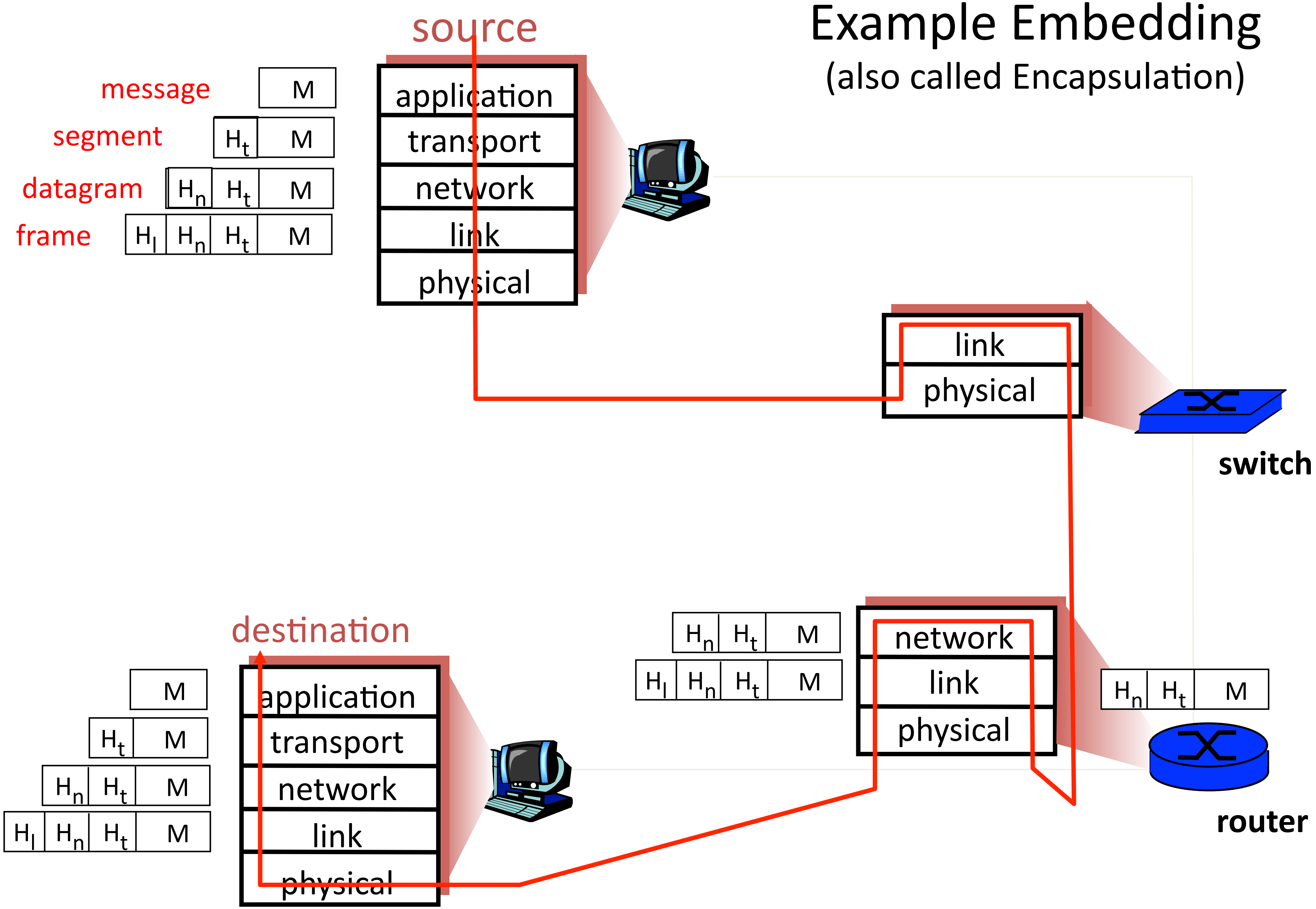
NOT

determine implementation strategy



Example Embedding

(also called Encapsulation)



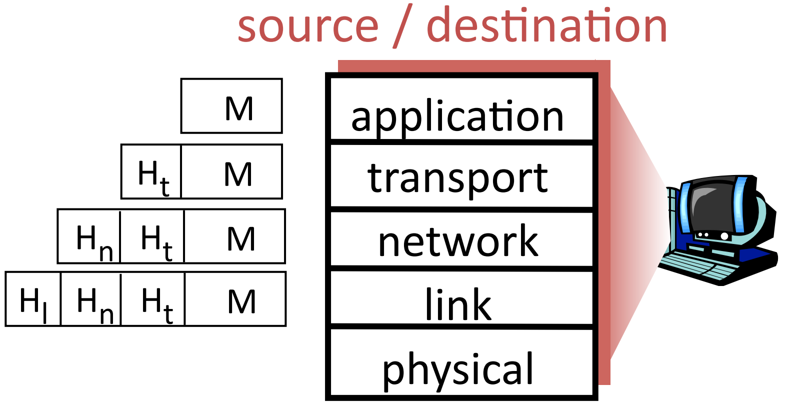
Distributing Layers Across Network

- 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?

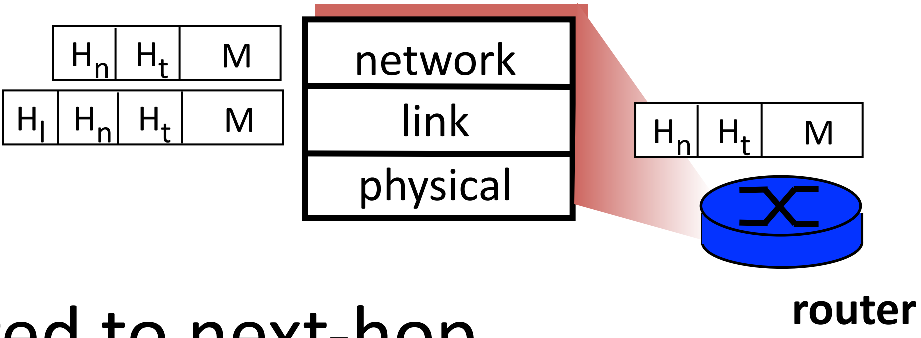
What Gets Implemented on Host?

- Bits arrive on wire, must make it up to application

- Therefore, all layers must exist at the host



What Gets Implemented on a Router?

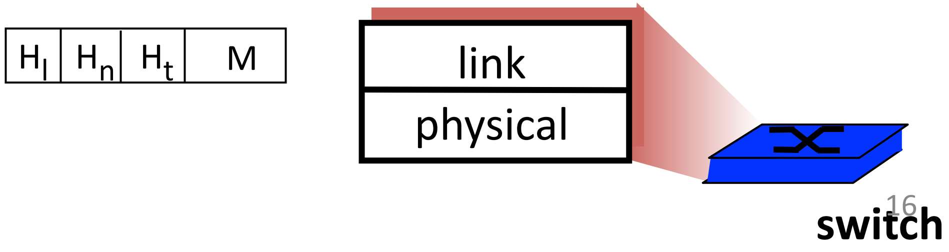


- Bits arrive on wire
  - Physical layer necessary
- Packets must be delivered to next-hop
  - Datalink layer necessary
- Routers participate in global delivery
  - Network layer necessary
- Routers don’t support reliable delivery
  - Transport layer (and above) ***not*** supported

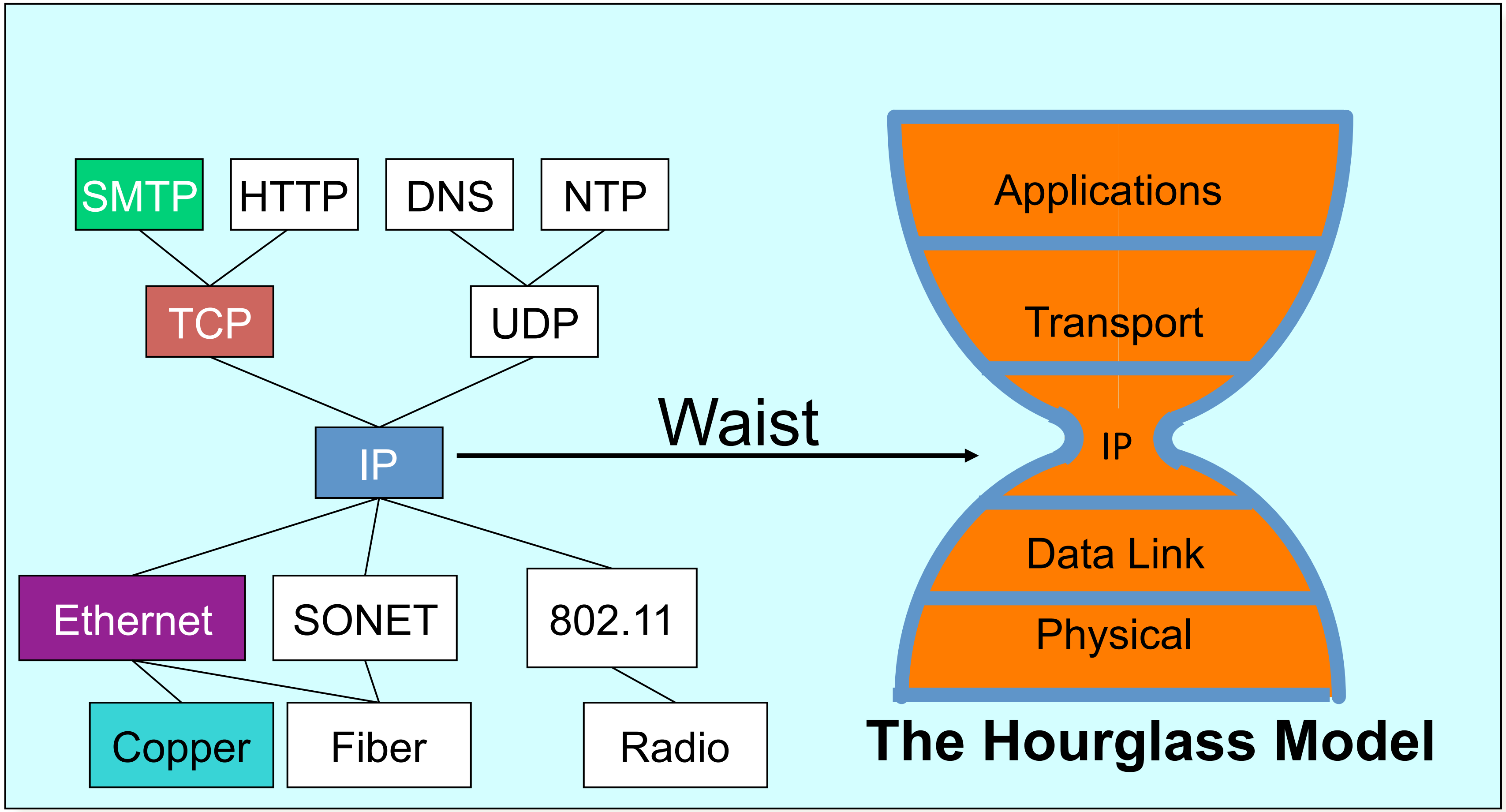
What Gets Implemented on Switches?

- 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
  - When I say switch, I almost always mean router
  - Almost all boxes support network layer these days

Routers have switches but switches do not have routers

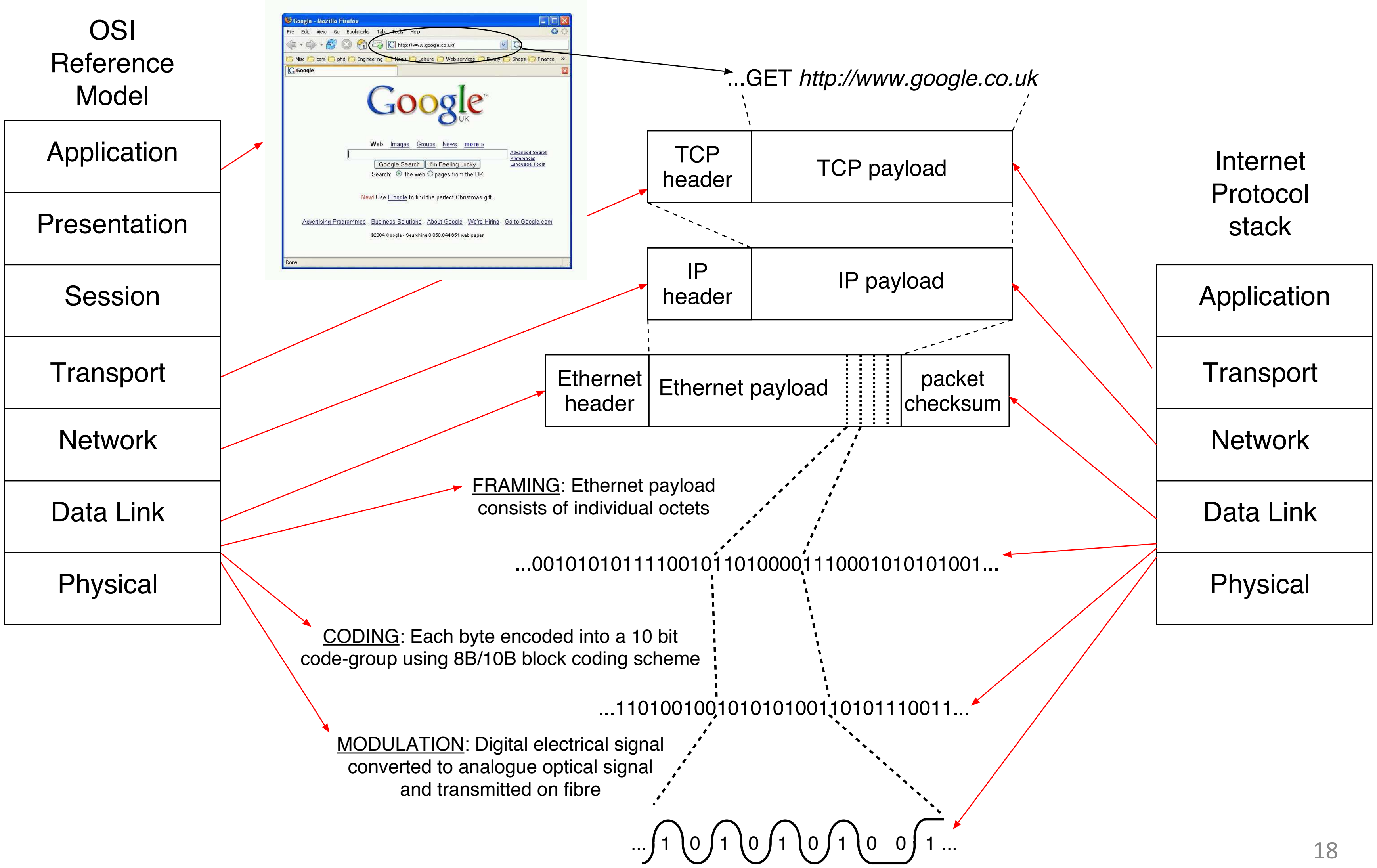


The Internet *Hourglass*



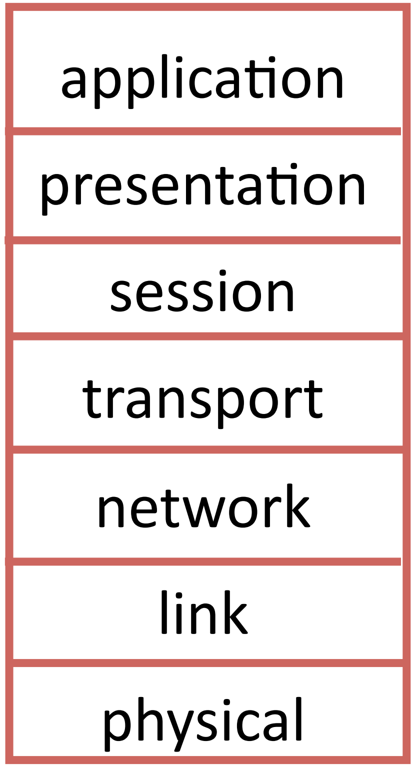
There is just one network-layer protocol, **IP**.
The “narrow waist” facilitates interoperability.

Internet protocol stack *versus* OSI Reference Model


OSI Reference Model
Application
Presentation
Session
Transport
Network
Data Link
Physical
Google - Mozilla Firefox
File Edit View Go Bookmarks Tab Tools Help
http://www.google.co.uk/
Misc cam phd Engineering News Leisure Web services Funny Shops Finance
Google
Google UK
Web Images Groups News more »
Google Search I'm Feeling Lucky
Advanced Search Preferences Language Tools
Search: the web pages from the UK
New! Use Froogle to find the perfect Christmas gift.
Advertising Programmes - Business Solutions - About Google - We're Hiring - Go to Google.com
©2004 Google - Searching 8,058,044,651 web pages
Done
...GET *http://www.google.co.uk*
TCP header
TCP payload
IP header
IP payload
Ethernet header
Ethernet payload
packet checksum
FRAMING: Ethernet payload consists of individual octets
...0010101011110010110100001110001010101001...
CODING: Each byte encoded into a 10 bit code-group using 8B/10B block coding scheme
...1101001001010101001110101110011...
MODULATION: Digital electrical signal converted to analogue optical signal and transmitted on fibre
... 1 0 1 0 1 0 1 0 0 1 ...
Internet Protocol stack
Application
Transport
Network
Data Link
Physical

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
  - needed?



application

presentation

session

transport

network

link

physical

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

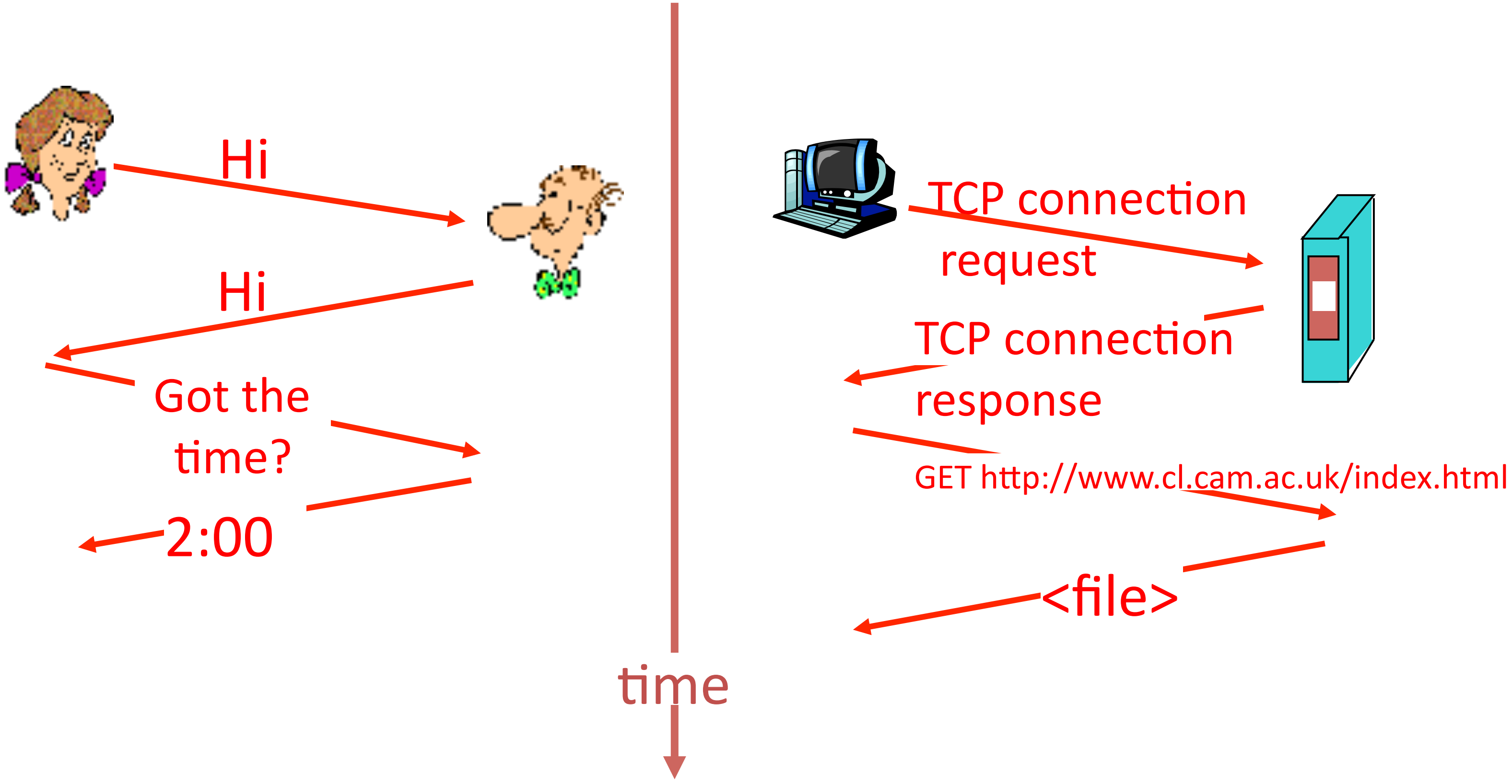
network protocols:

- 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:



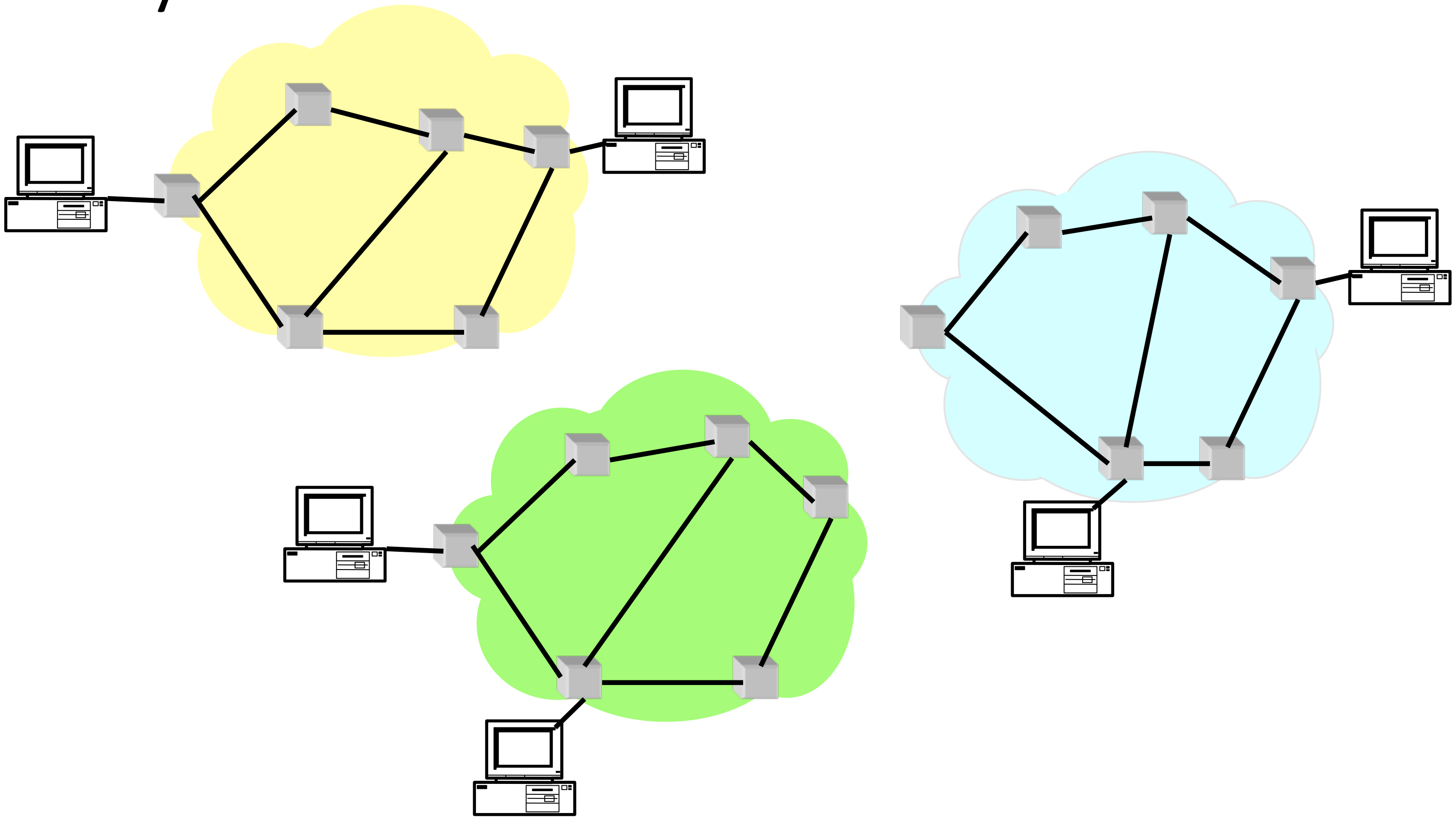
Q: Other human protocols?

Protocol Standardization

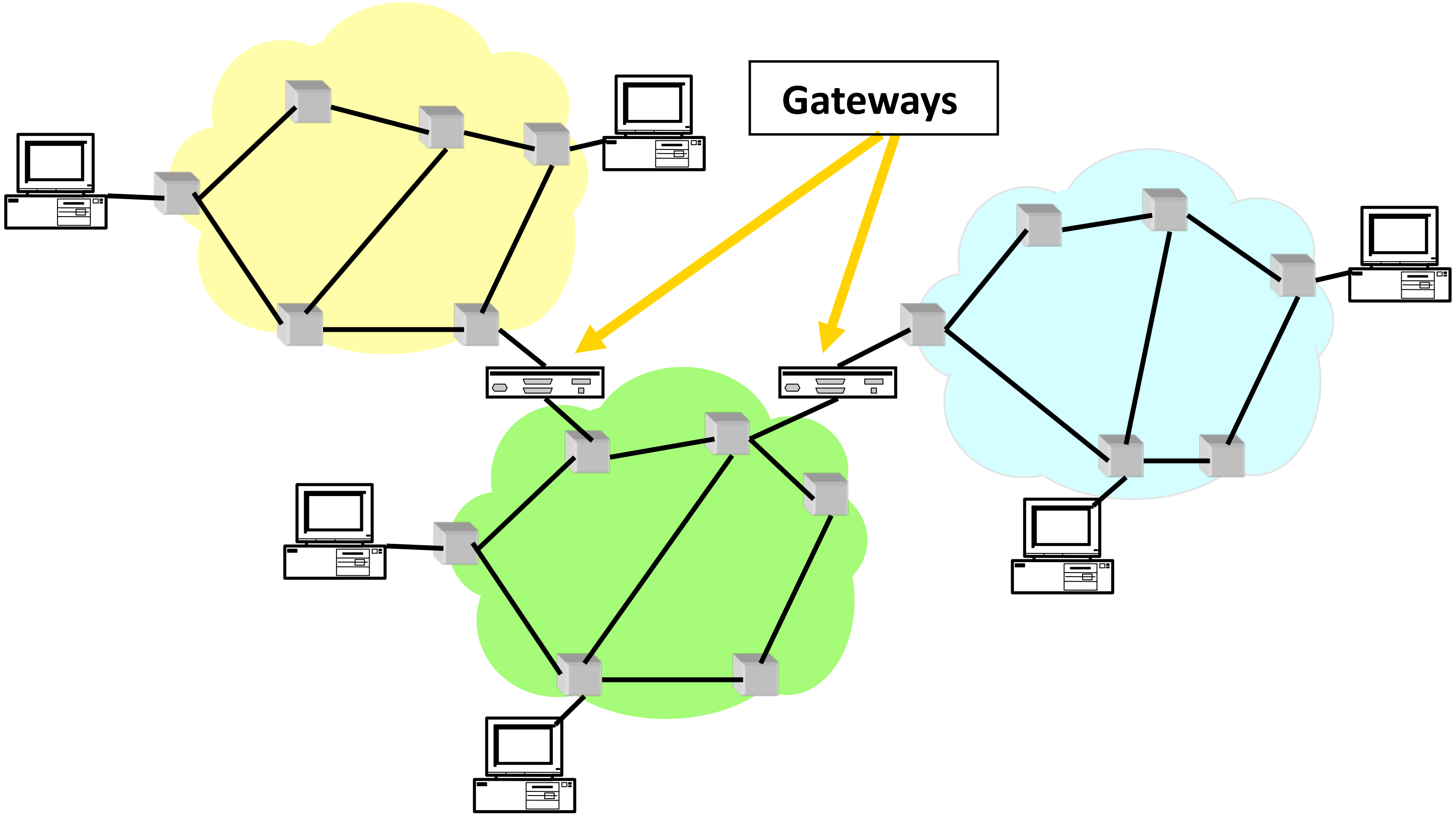
- All hosts must follow same protocol
  - Very small modifications can make a big difference
  - Or prevent it from working altogether
  - Cisco bug compatible!
- This is why we have standards
  - Can have multiple implementations of protocol
- Internet Engineering Task Force
  - Based on working groups 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



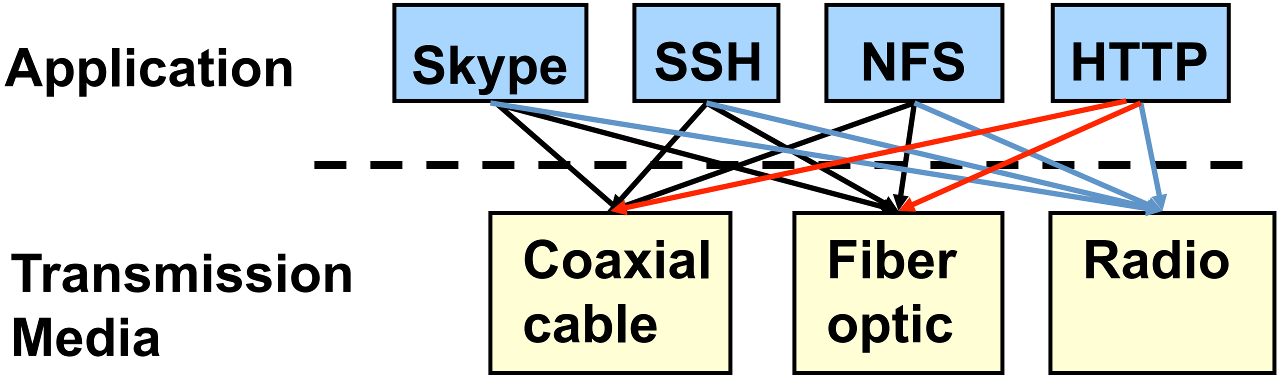
INTERnet Solution



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, many P2P implementations, etc.

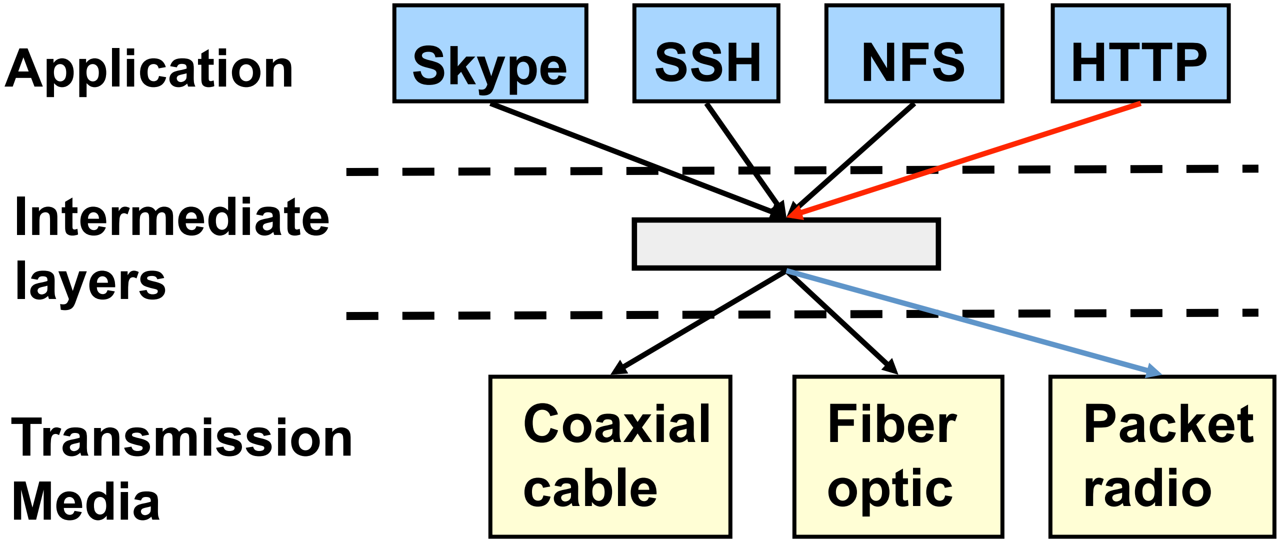
A Multitude of Apps Problem



- 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"



Remember that slide!

- The relationship between architectural principles and architectural decisions is crucial to understand

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~~

In the context of the Internet

Applications

…built on…

Reliable (or unreliable) transport

…built on…

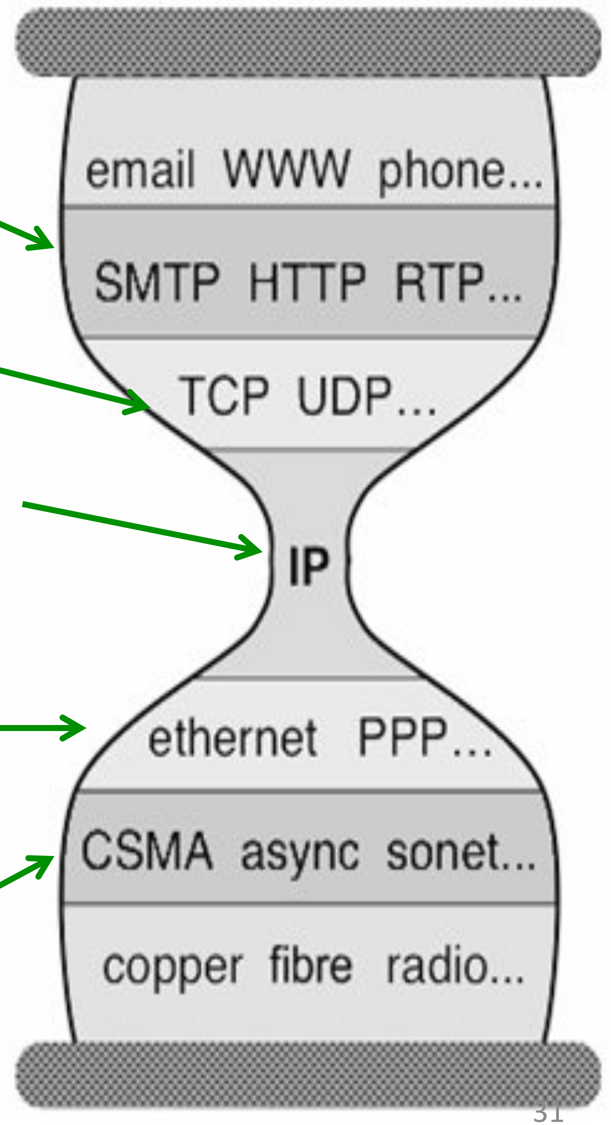
Best-effort global packet delivery

…built on…

Best-effort local packet delivery

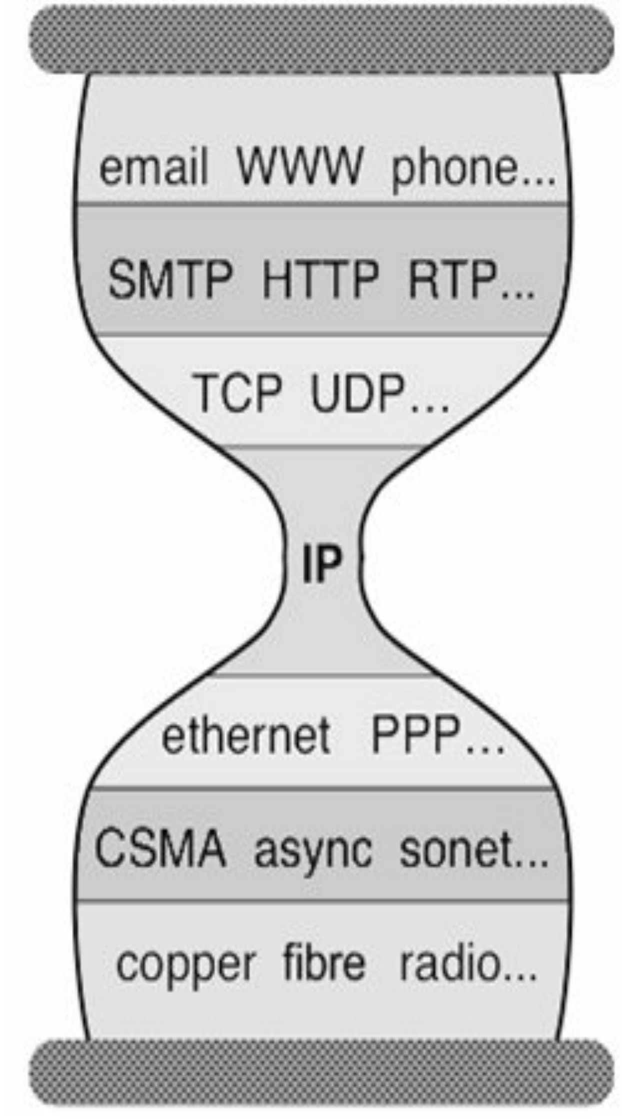
…built on…

Physical transfer of bits



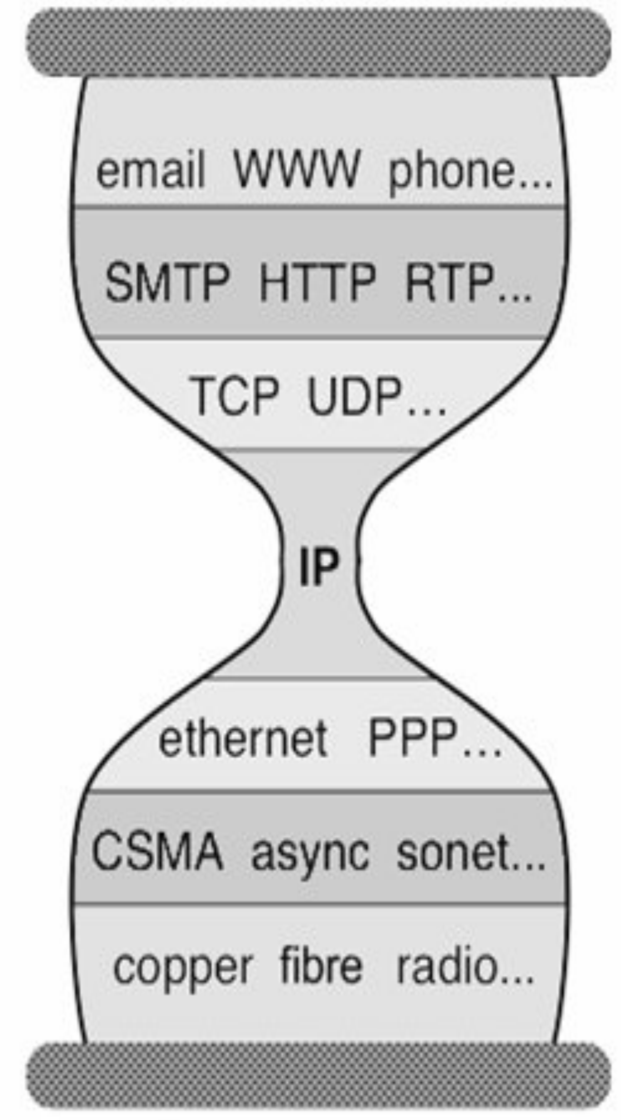
Three Observations

- 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!)

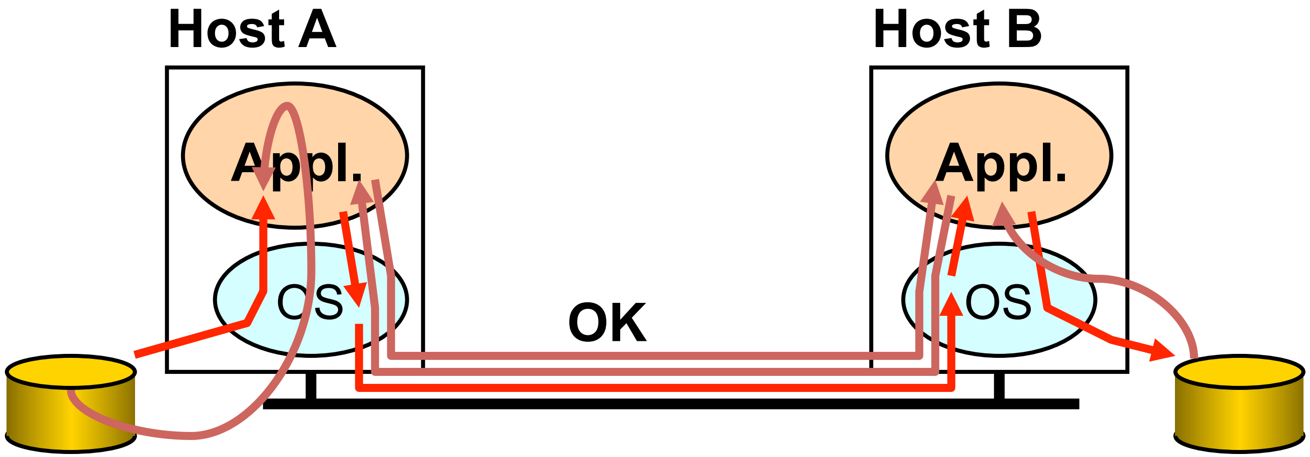
Basic Observation

- 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
- 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