Computer Networking

Michaelmas/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

2014-2015

Topic 1 Foundation

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

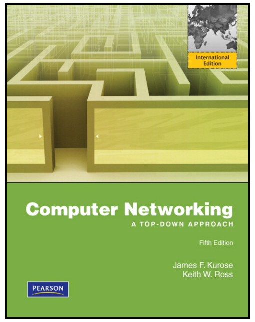
Course Administration

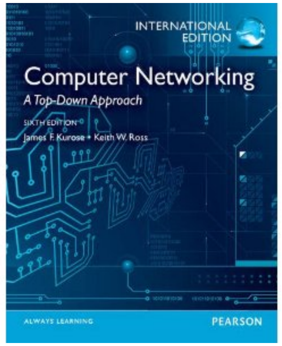
Commonly Available Texts

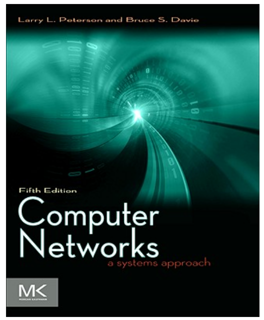
- ❑ Computer Networking: A Top-Down Approach
  Kurose and Ross, 6th edition 2013, Addison-Wesley
  (5th edition is also commonly available)
- ❑ Computer Networks: A Systems Approach
  Peterson and Davie, 5th edition 2011, Morgan-Kaufman

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





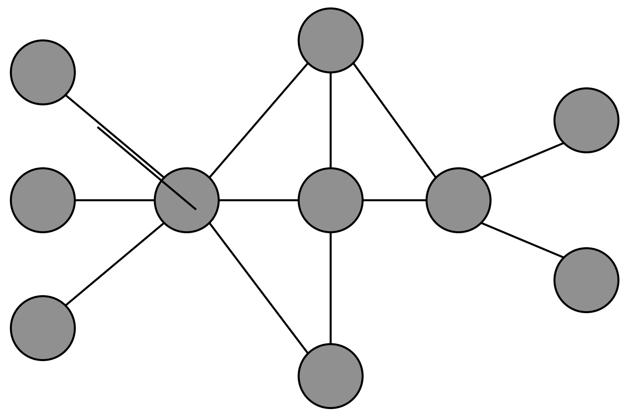


Thanks

- 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, and the fantastic TA teams of 144 and 168
- Practical material will become available through this year
  But would be impossible without Georgina Kalogeridou,
  Nick McKeown, Bob Lantz, Te-Yuan Huang and Vimal Jeyakumar
- 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



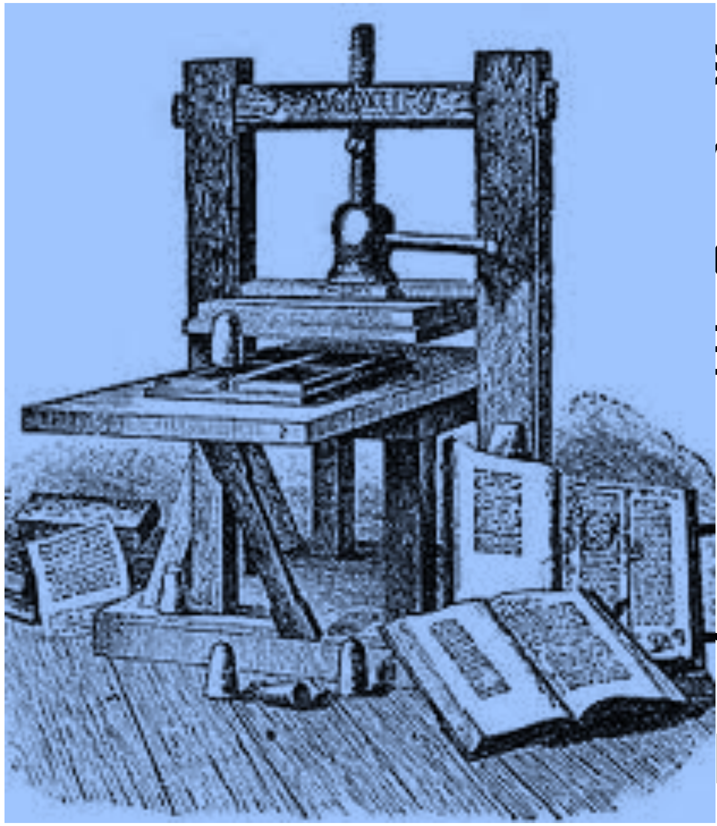
- Yes, this is very vague

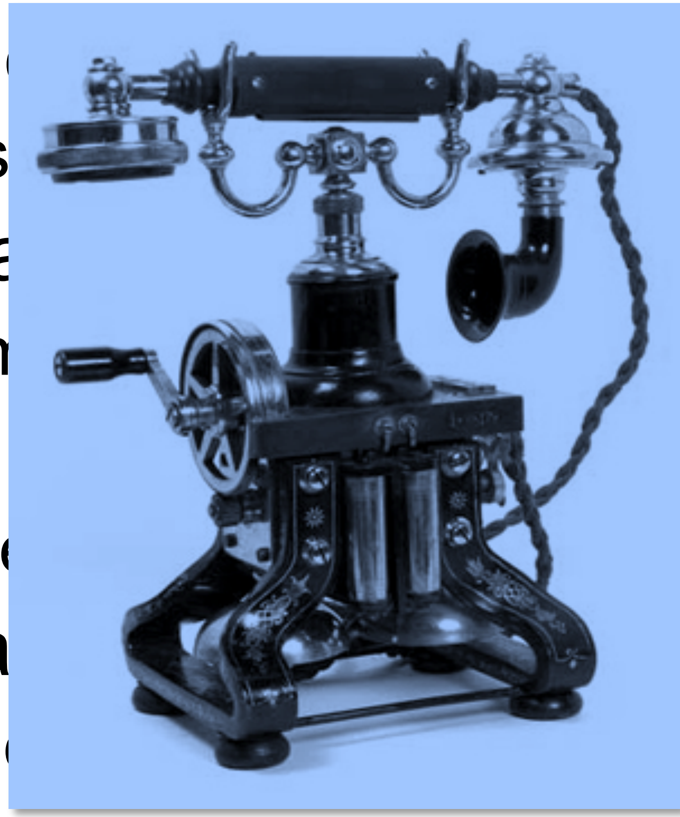
There are *many* different types of networks

- 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*

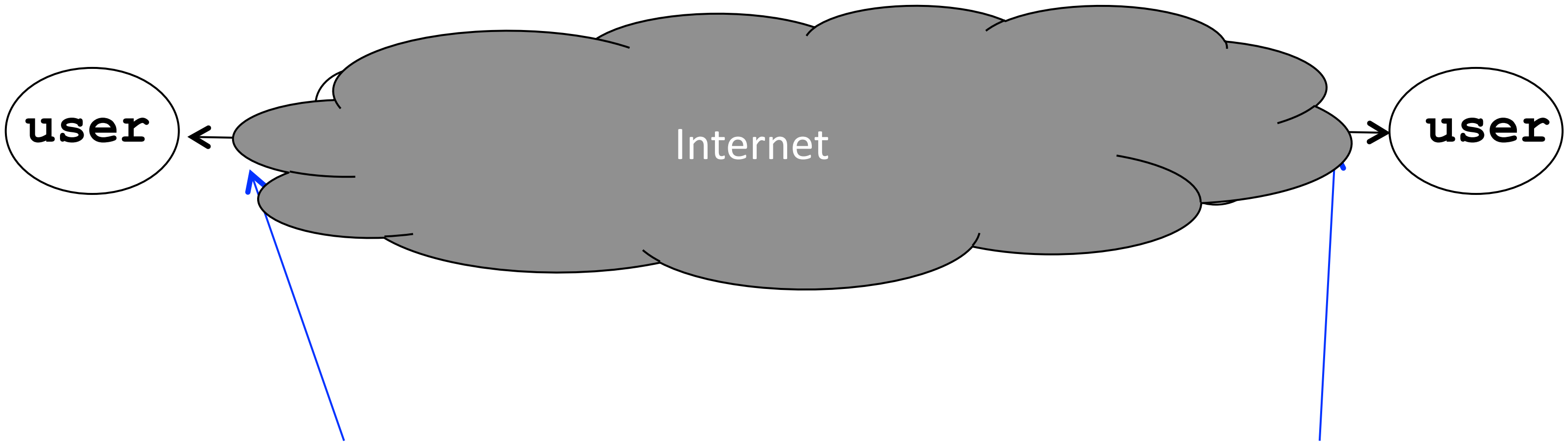
"What's with all these TLA protocols?" – *all*

"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

- 3 Billion users (43% of world population)
- 1+ Trillion unique URLs
- 194 Billion emails sent per day
- 1.75 Billion smartphones
- 1.23 Billion Facebook users
- 50 Billion WhatsApp messages per day
- 2 Billion YouTube videos watched per day
- Routers that switch 92Terabits/second
- Links that carry 400Gigabits/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

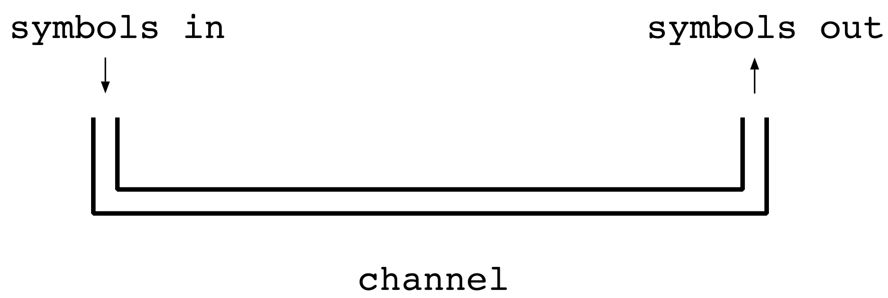
- Others?

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

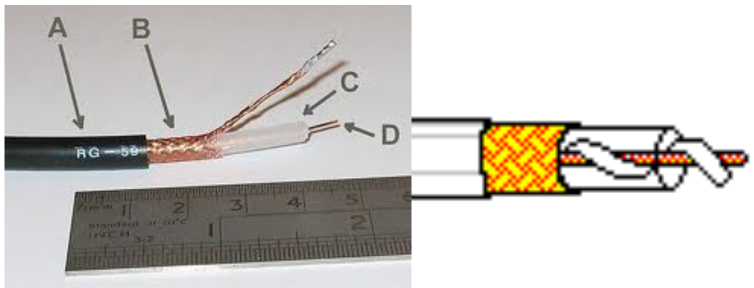
these example physical channels are also known as *Physical Media*

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:

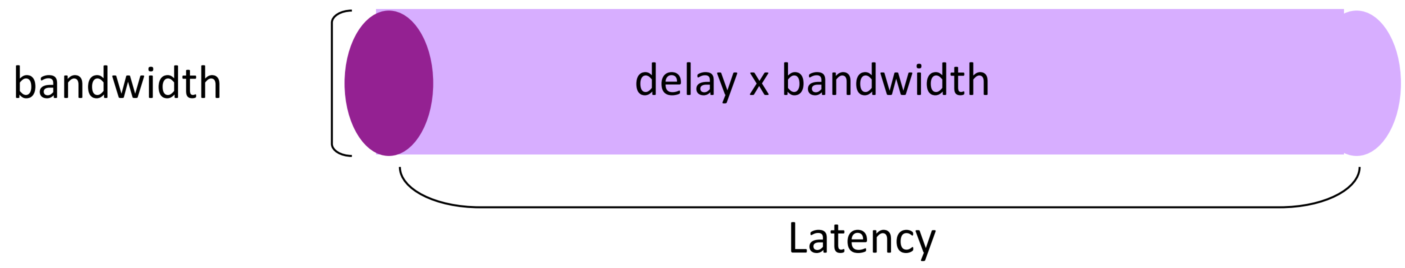
- terrestrial microwave
  - e.g. 45 Mbps channels
- LAN (e.g., Wifi)
  - 11Mbps, 54 Mbps, 200 Mbps
- wide-area (e.g., cellular)
  - 4G cellular: ~ 4 Mbps
- satellite
  - Kbps to 45Mbps channel (or multiple smaller channels)
  - 270 msec end-end delay
  - geosynchronous versus low altitude

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

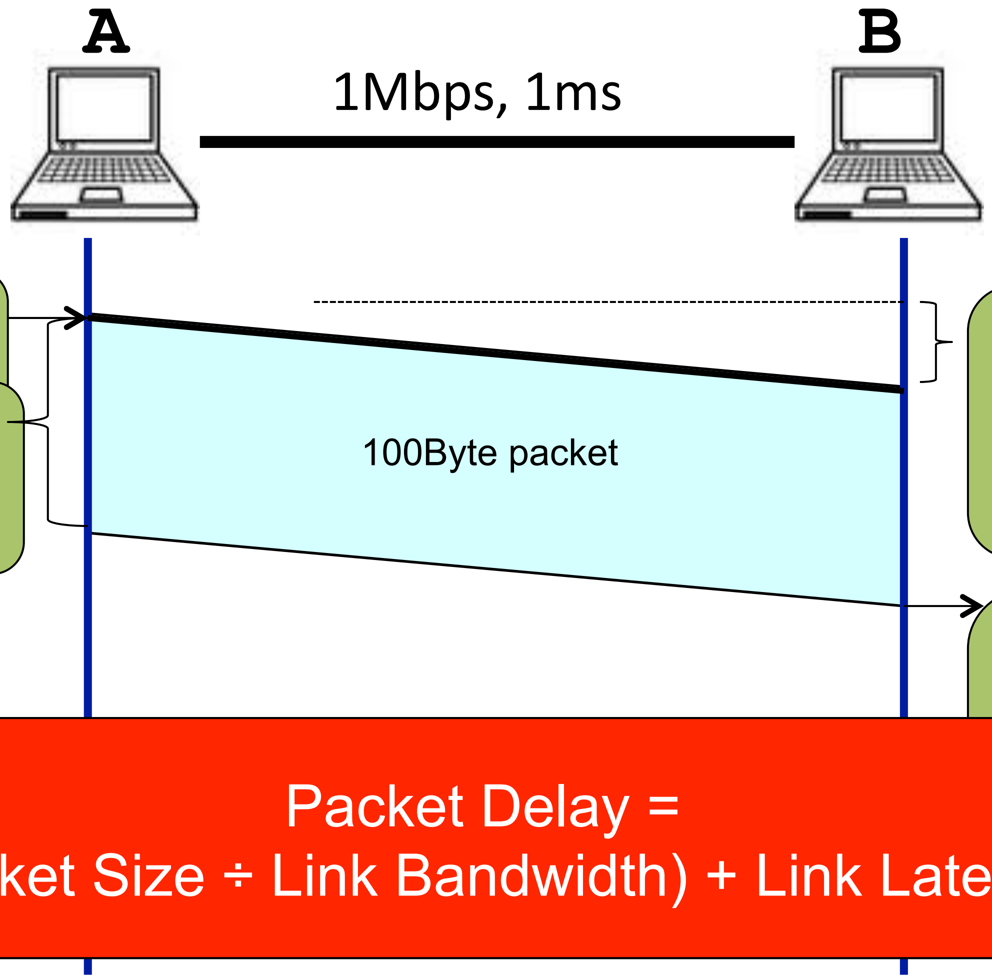
Examples of Bandwidth-Delay

- 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+1/10^3s

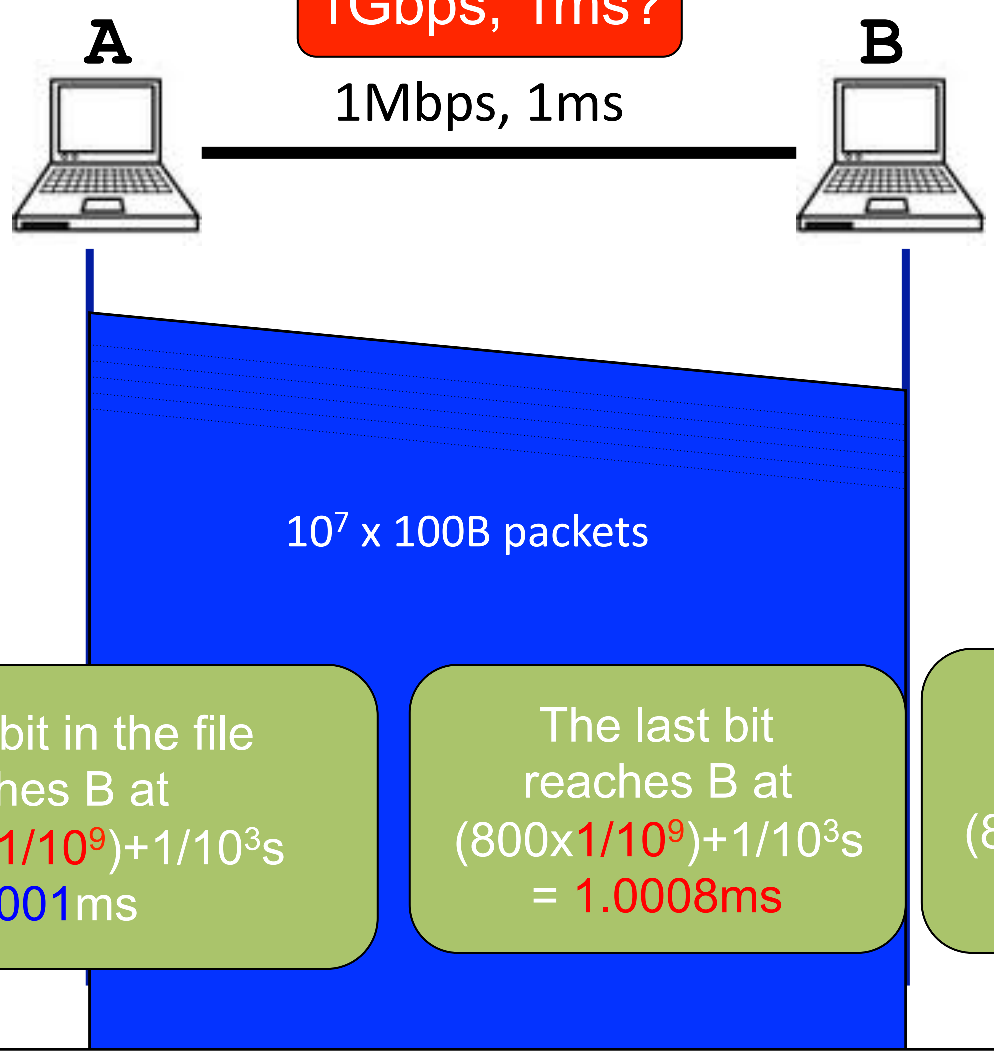
The last bit
at
/10^3s

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

1GB file in 100B packets

ay

Sending a 100B packet from A to B?



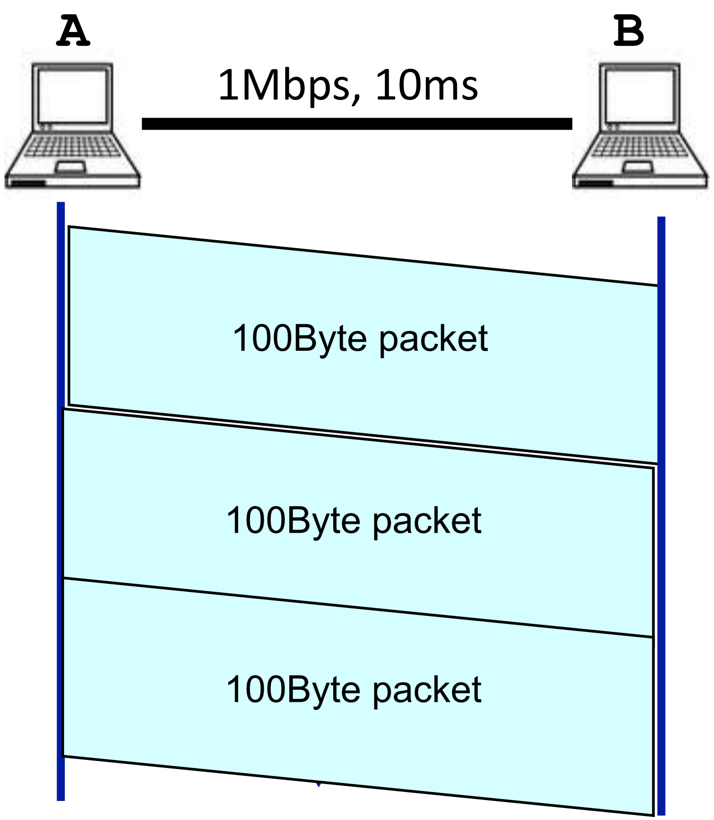
The last bit in the file reaches B at $(10^7 \times 800 \times 1/10^9) + 1/10^3$ s = 8001ms

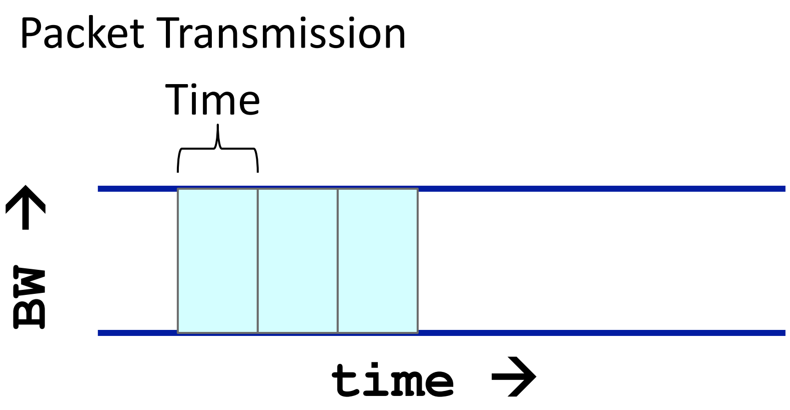
The last bit reaches B at $(800 \times 1/10^9) + 1/10^3$ s = 1.0008ms

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

Packet Delay: The "pipe" view

Sending 100B packets from A to B?

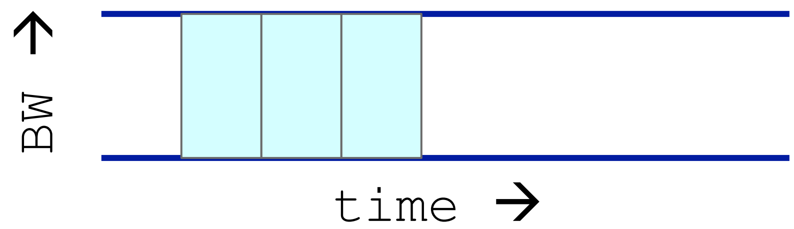




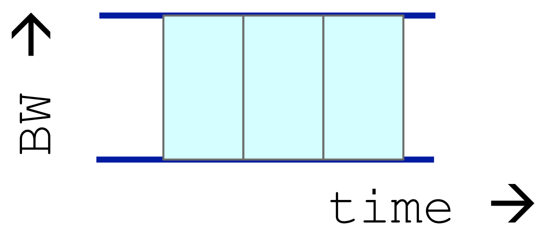
Packet Delay: The "pipe" view

Sending 100B packets from A to B?

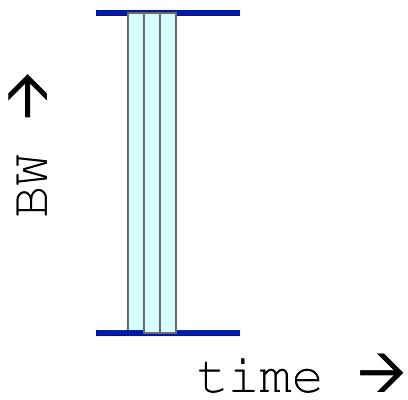
1Mbps, 10ms (BDP=10,000)



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

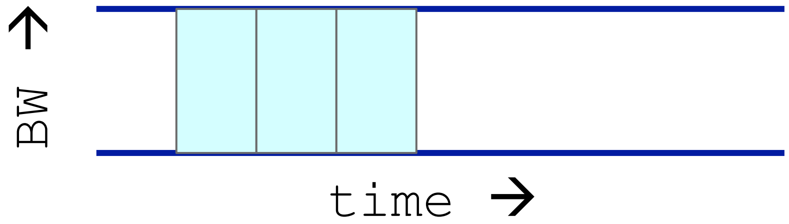


10Mbps, 1ms (BDP=10,000)



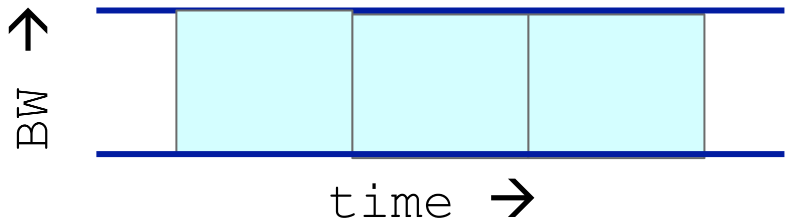
Packet Delay: The "pipe" view
Sending 100B packets from A to B?

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



What if we used *200Byte packets??*

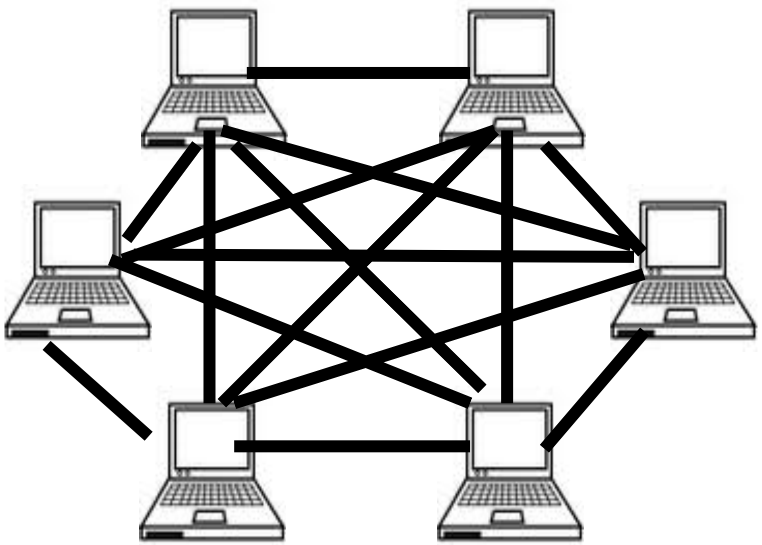
1Mbps, 10ms (BDP=10,000)



Recall Nodes and Links

What if we have more nodes?

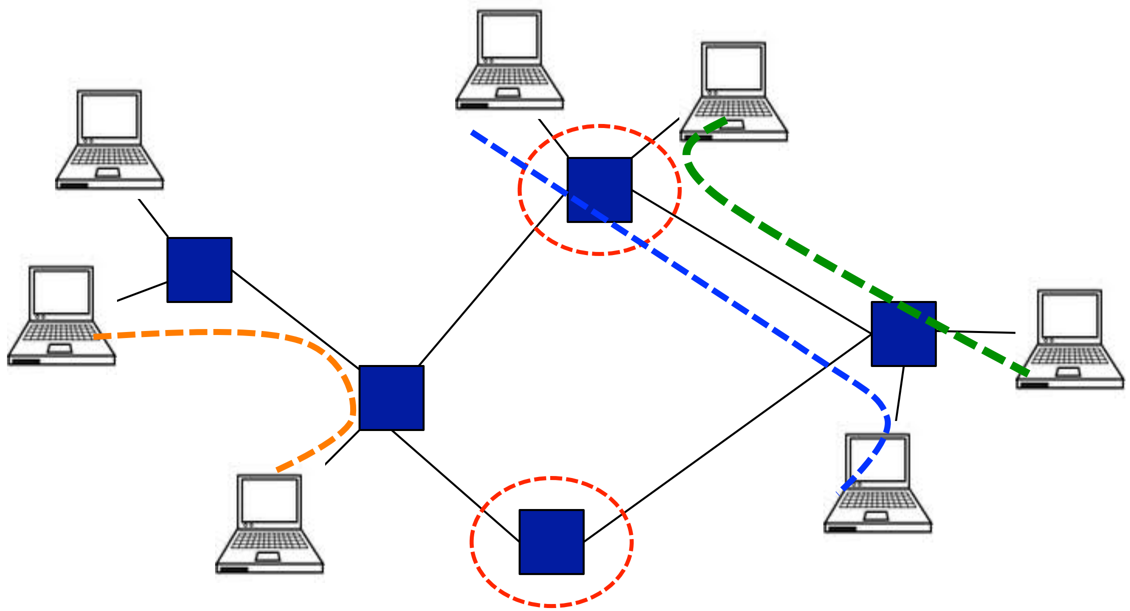
One link for every node?



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

Solution: A switched network

Nodes share network link resources



How is this sharing implemented?

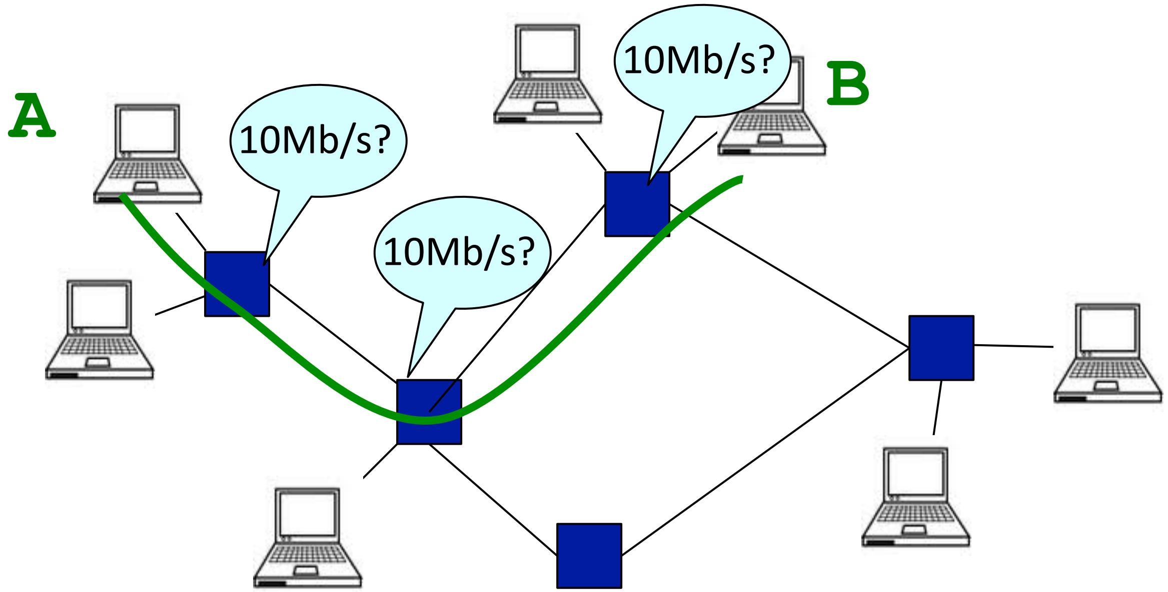
Two forms of switched networks

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

- Packet switching (used in the Internet)

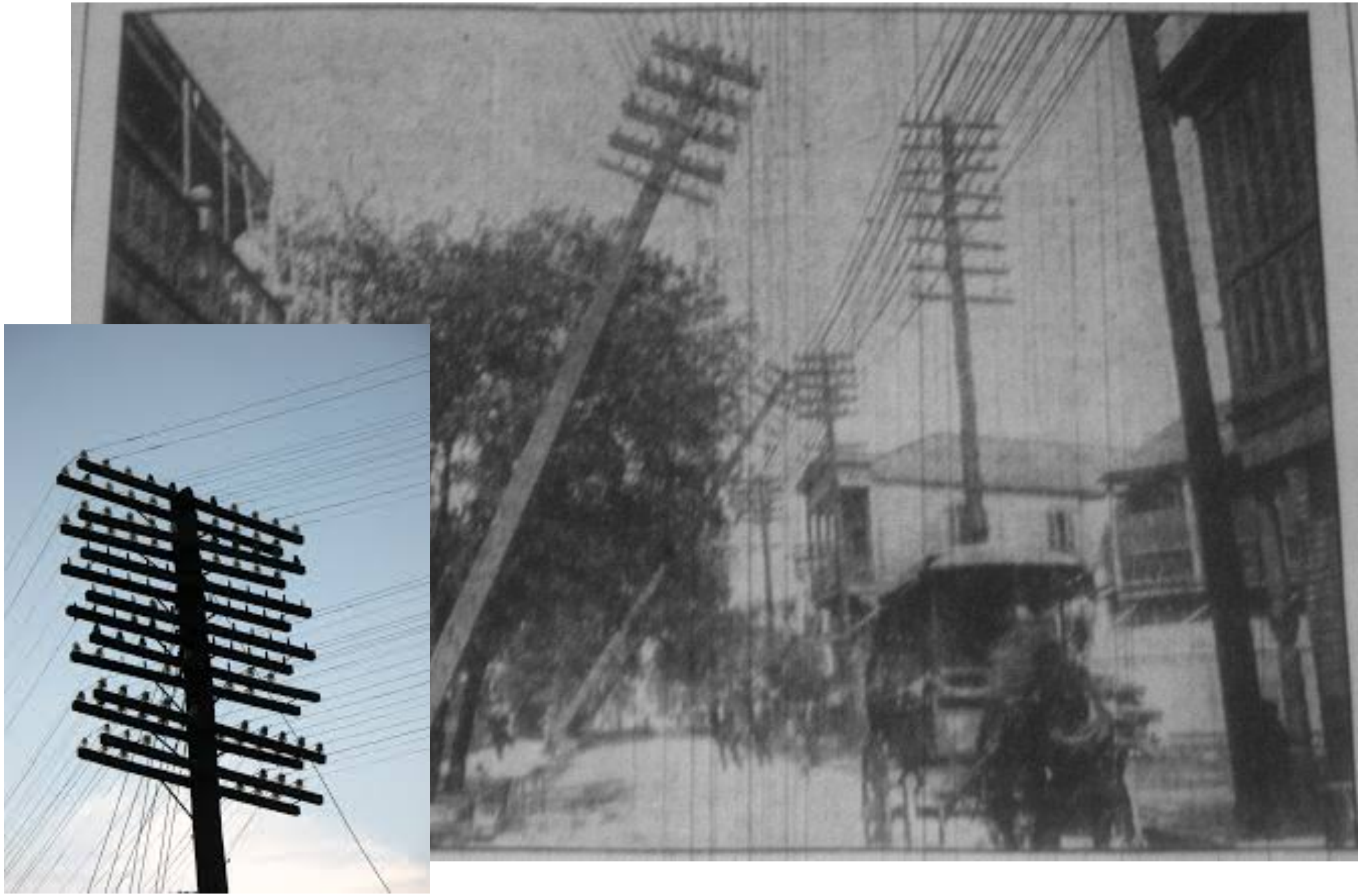
Circuit switching

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

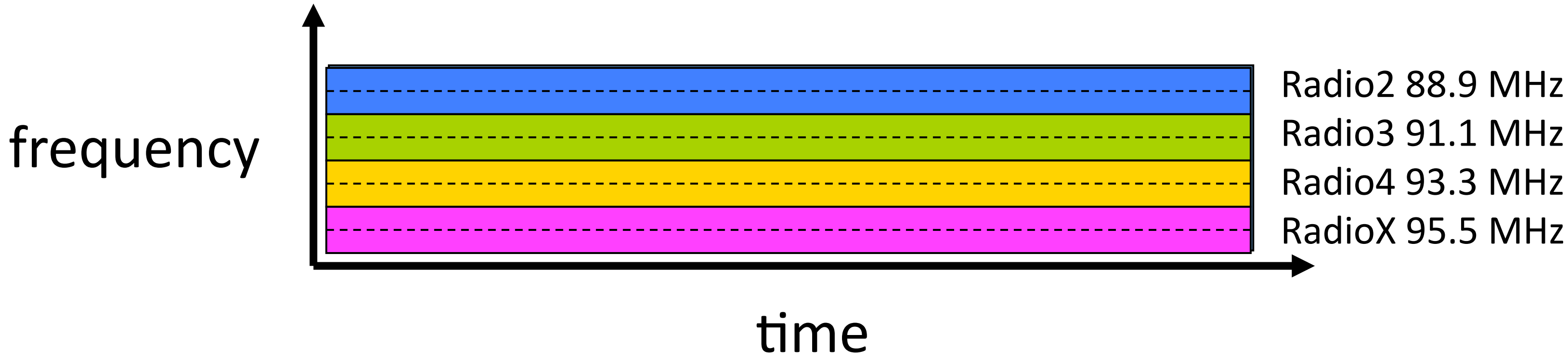


Circuit Switching: FDM and TDM

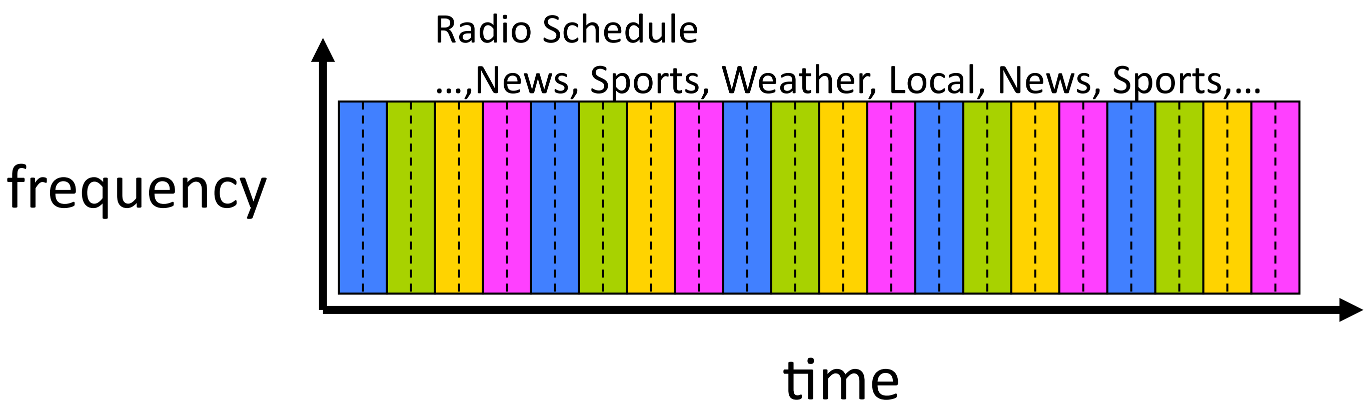
Example:

4 users

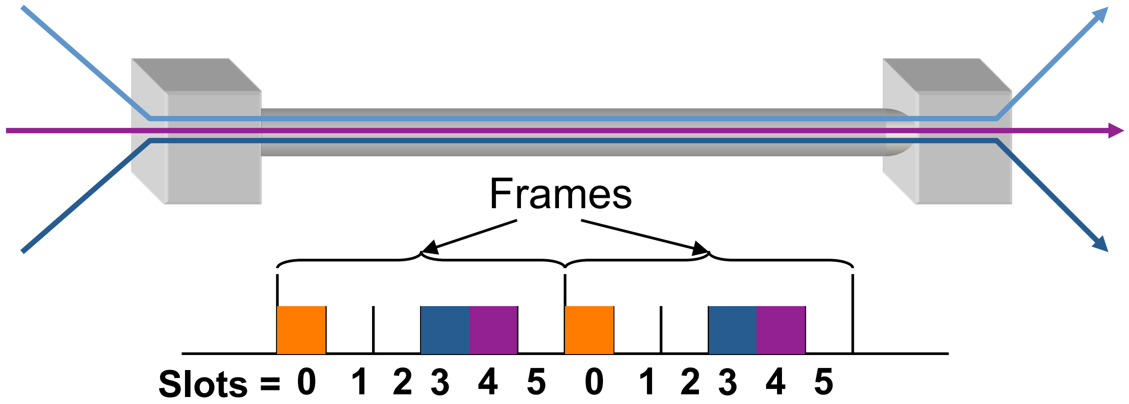
Frequency Division Multiplexing



Time Division Multiplexing

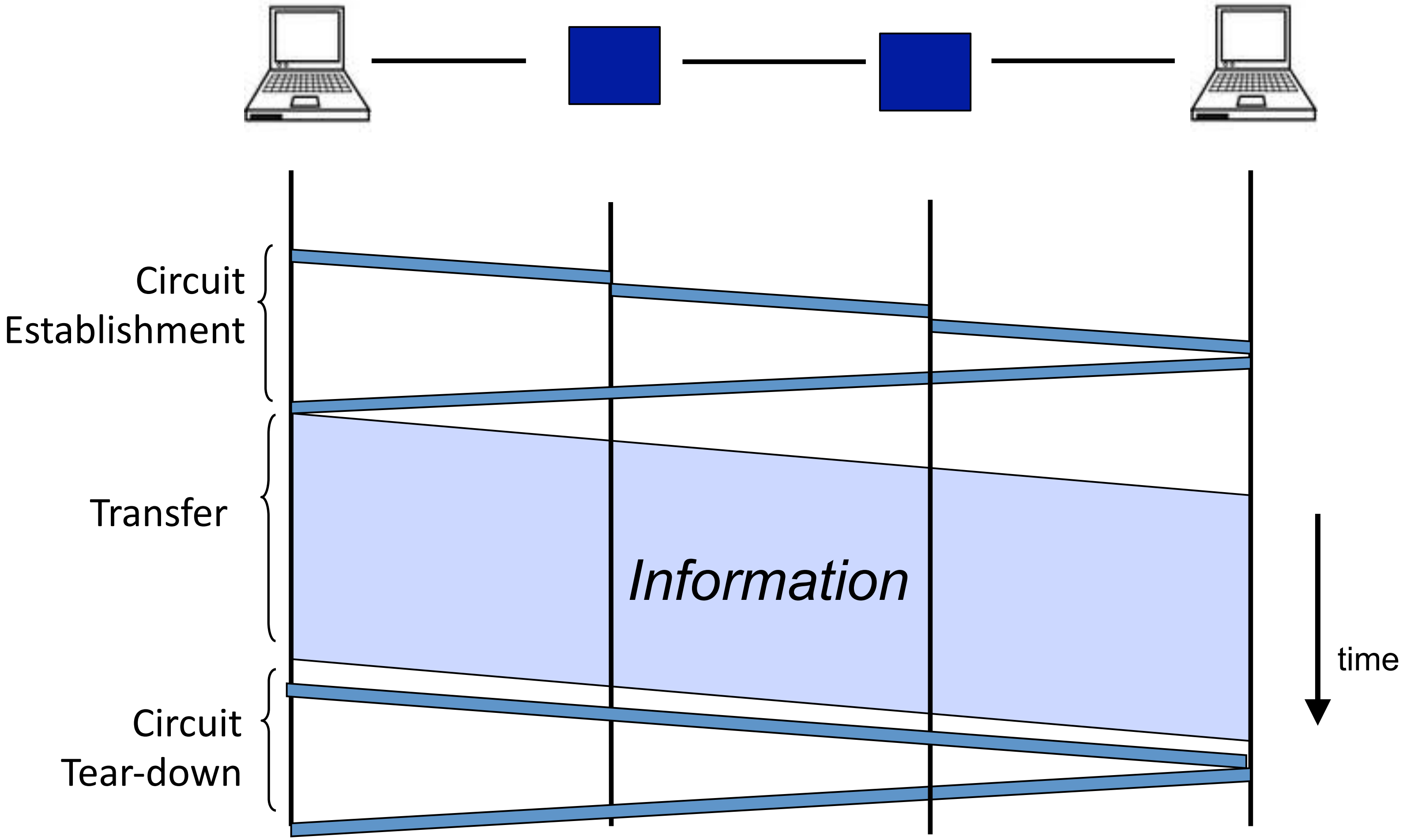


Time-Division Multiplexing/Demultiplexing



- 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!**

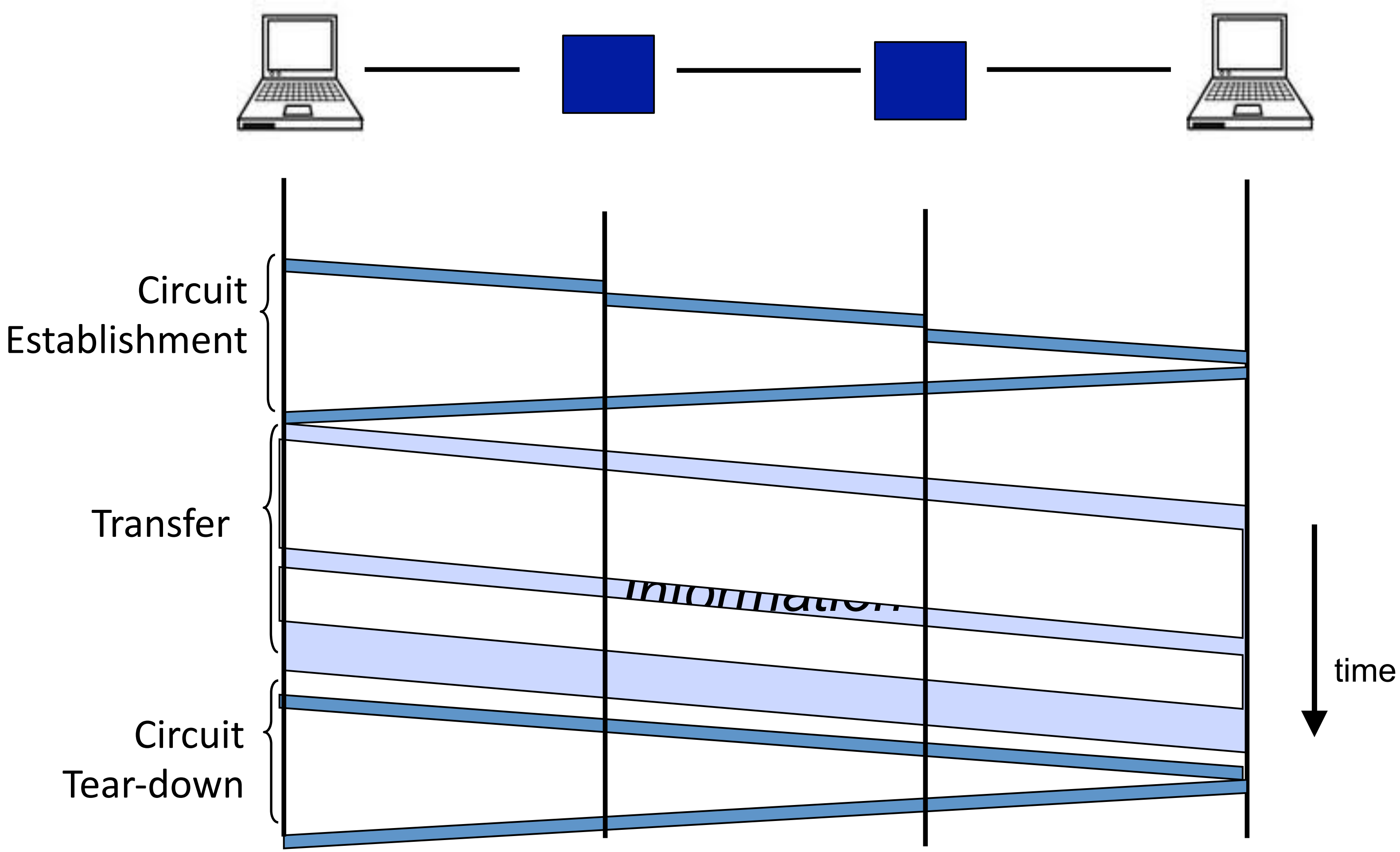
Timing in Circuit Switching



Circuit switching: pros and cons

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

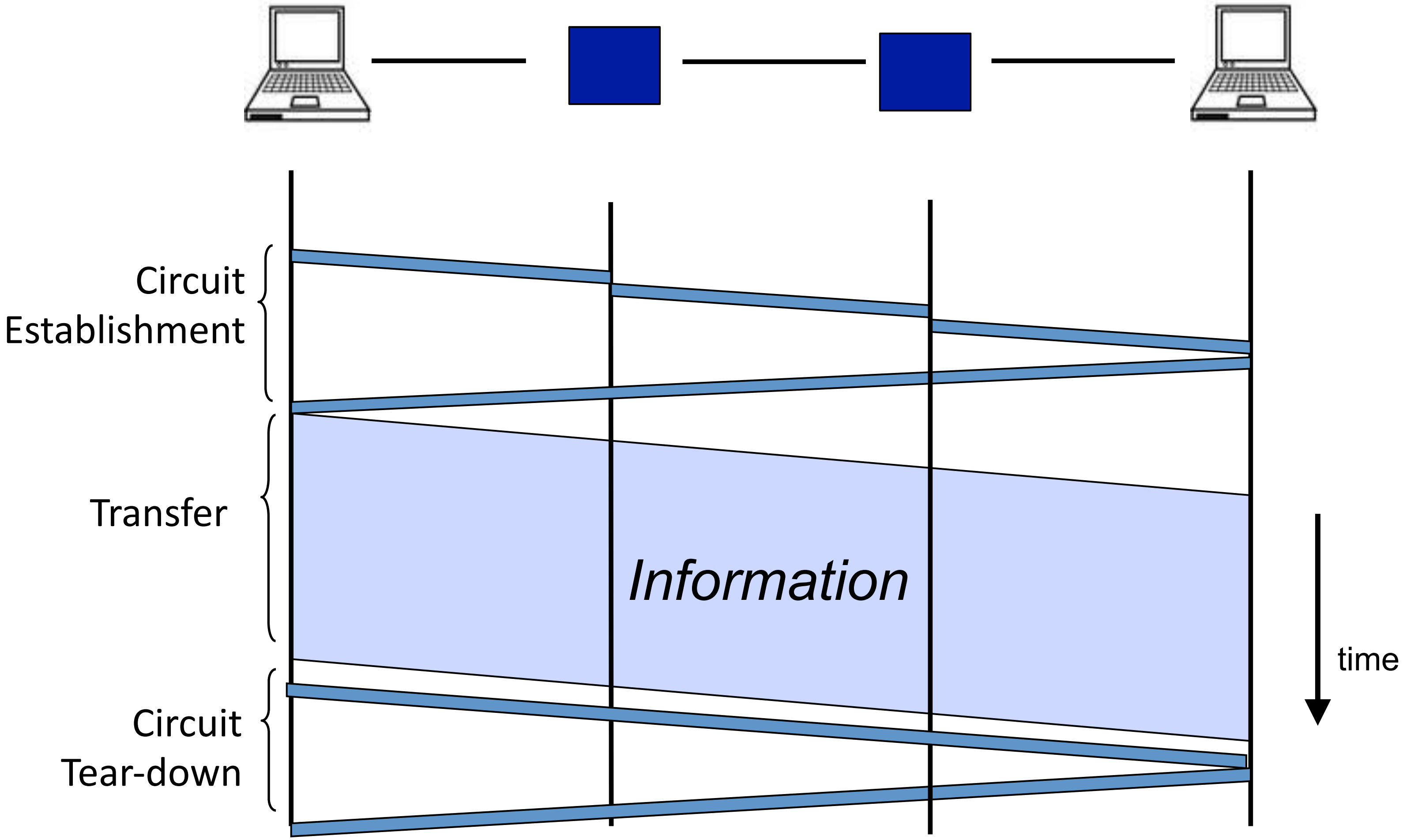
Timing in Circuit Switching



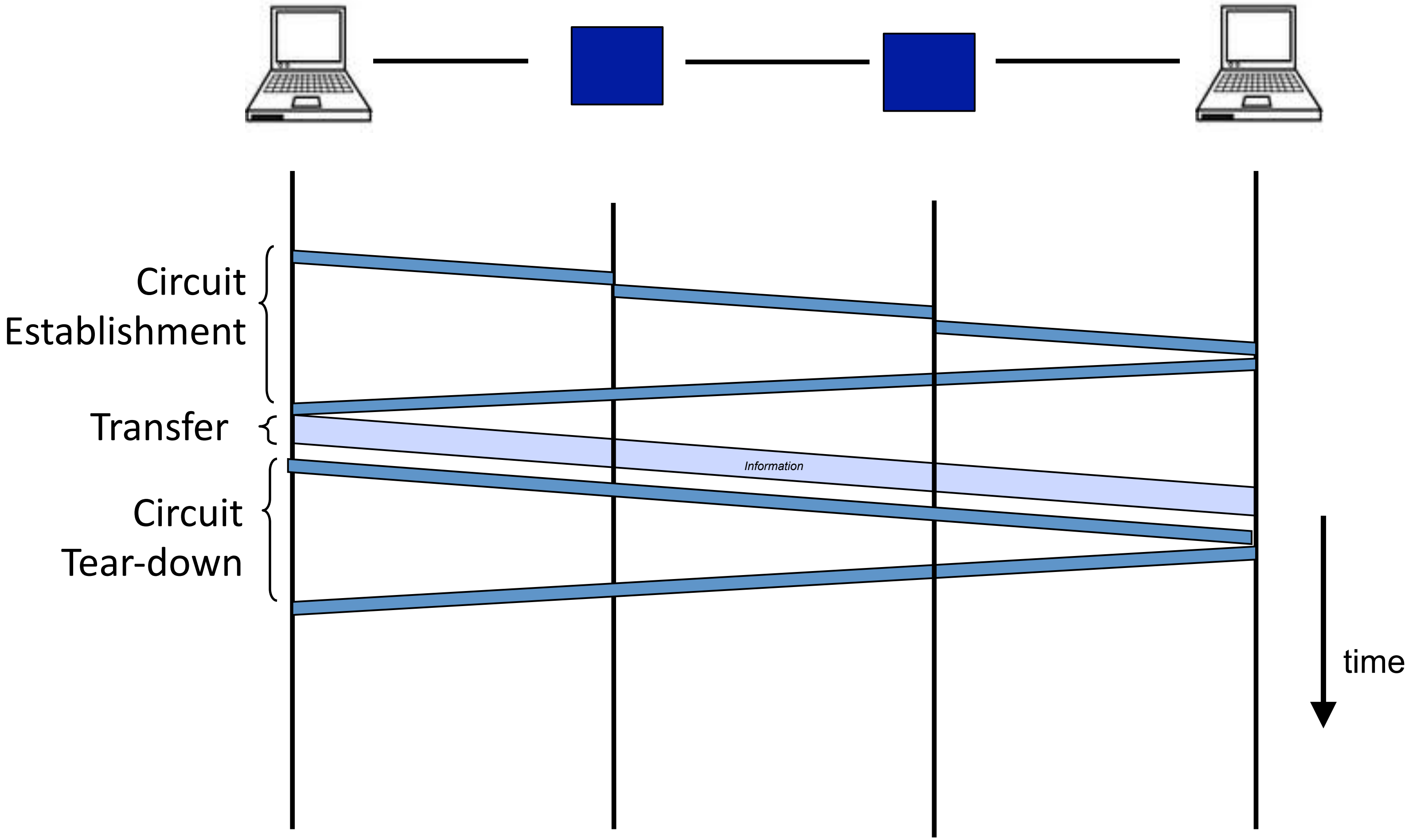
Circuit switching: pros and cons

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

Timing in Circuit Switching



Timing in Circuit Switching

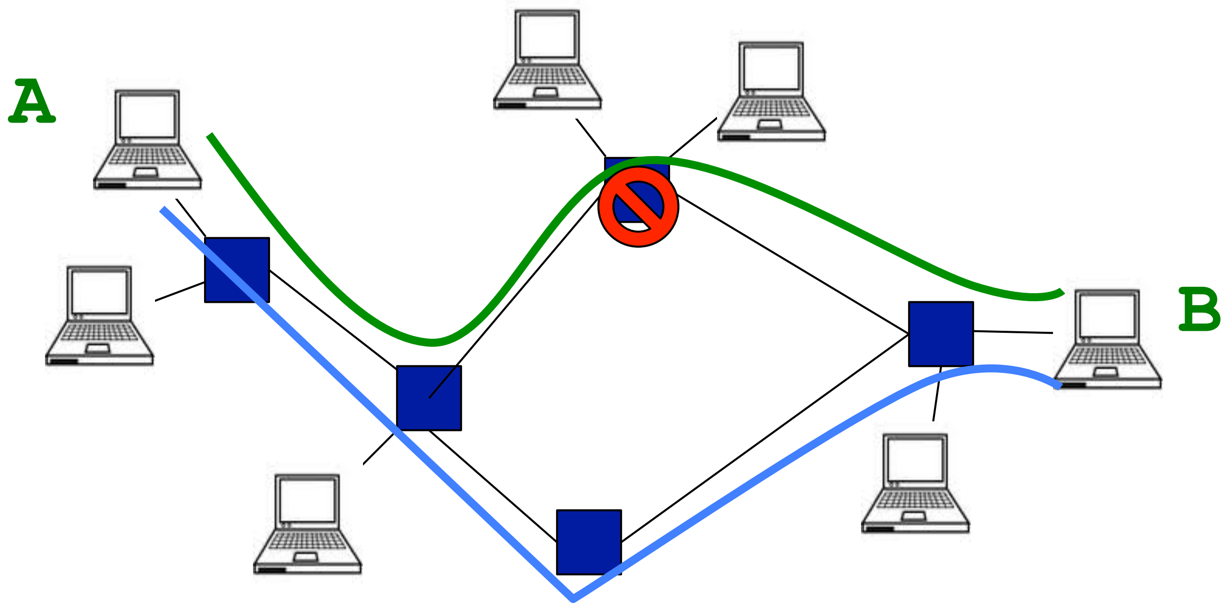


Circuit switching: pros and cons

- 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!

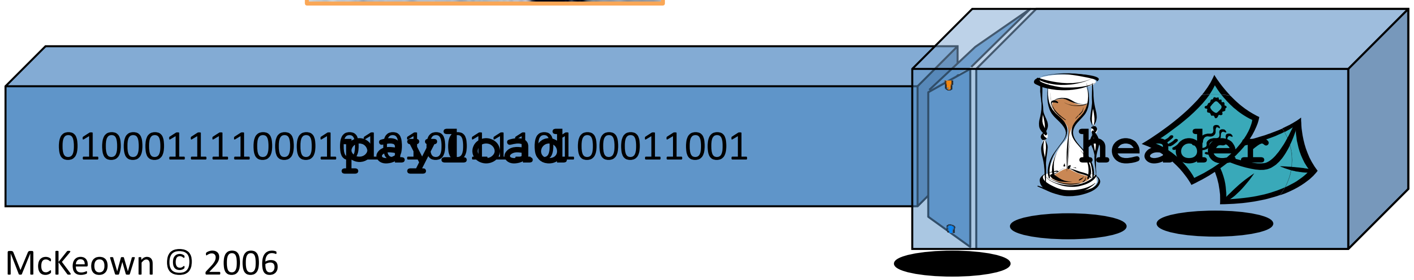
Two forms of switched networks

- 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



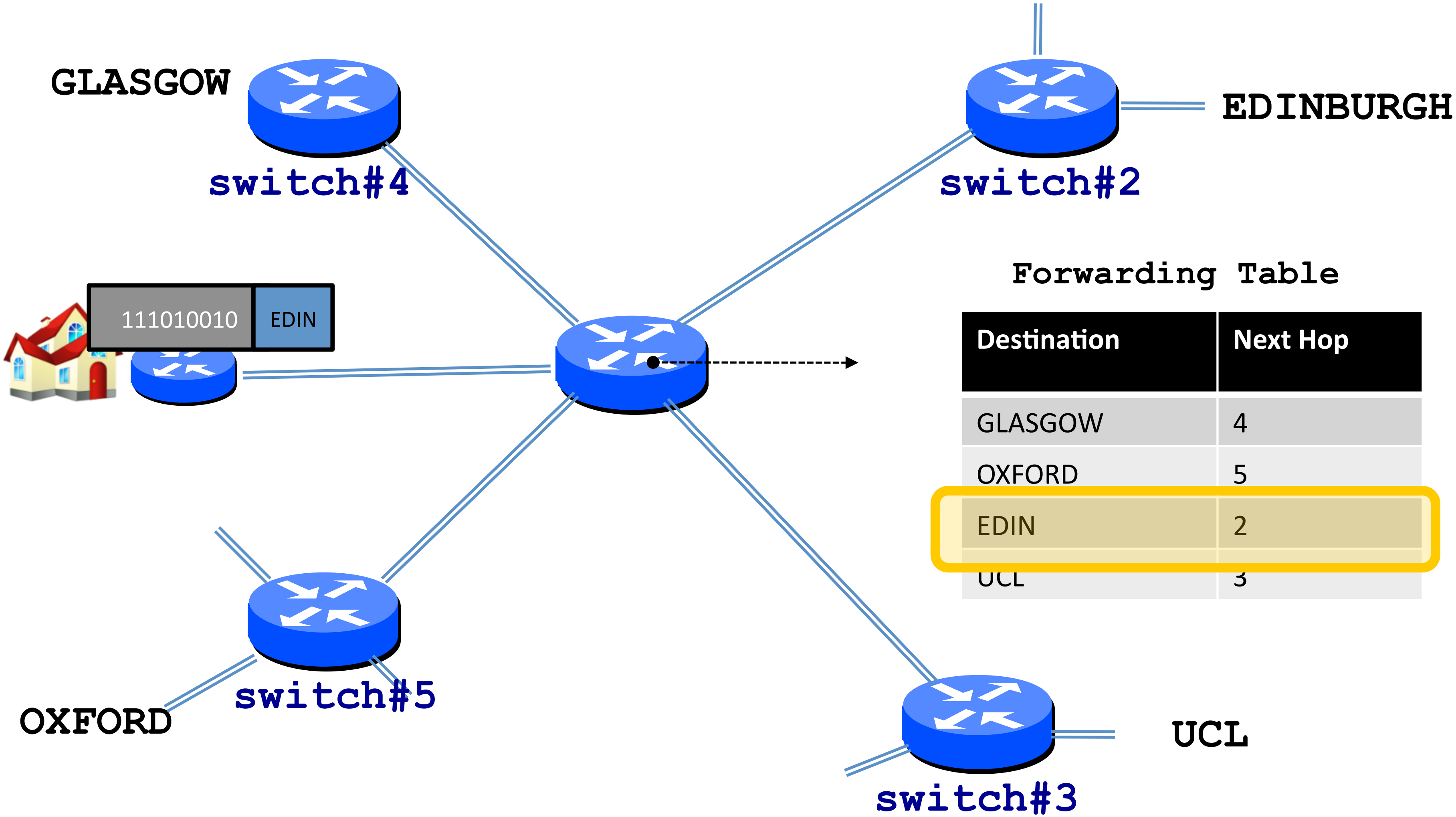
Packet Switching

- 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

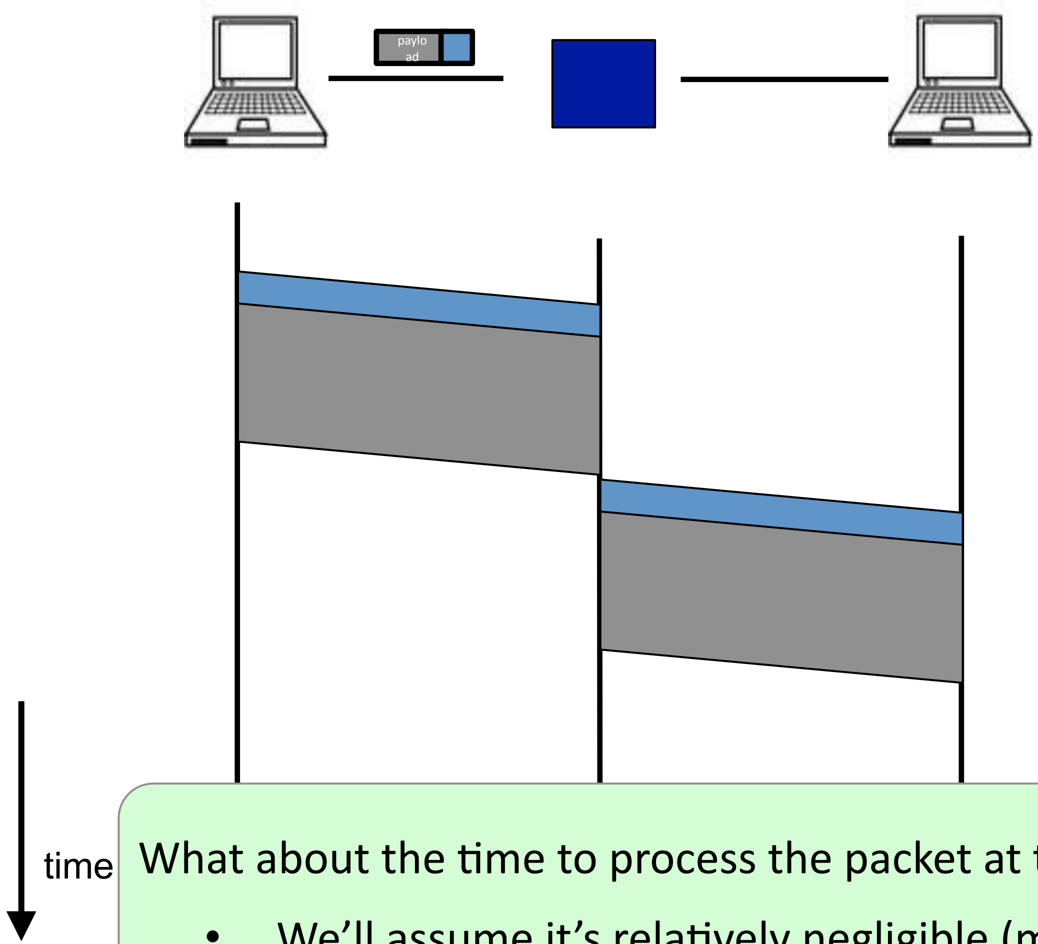
Switches forward packets



Forwarding Table

Destination	Next Hop
GLASGOW	4
OXFORD	5
EDIN	2
UCL	3

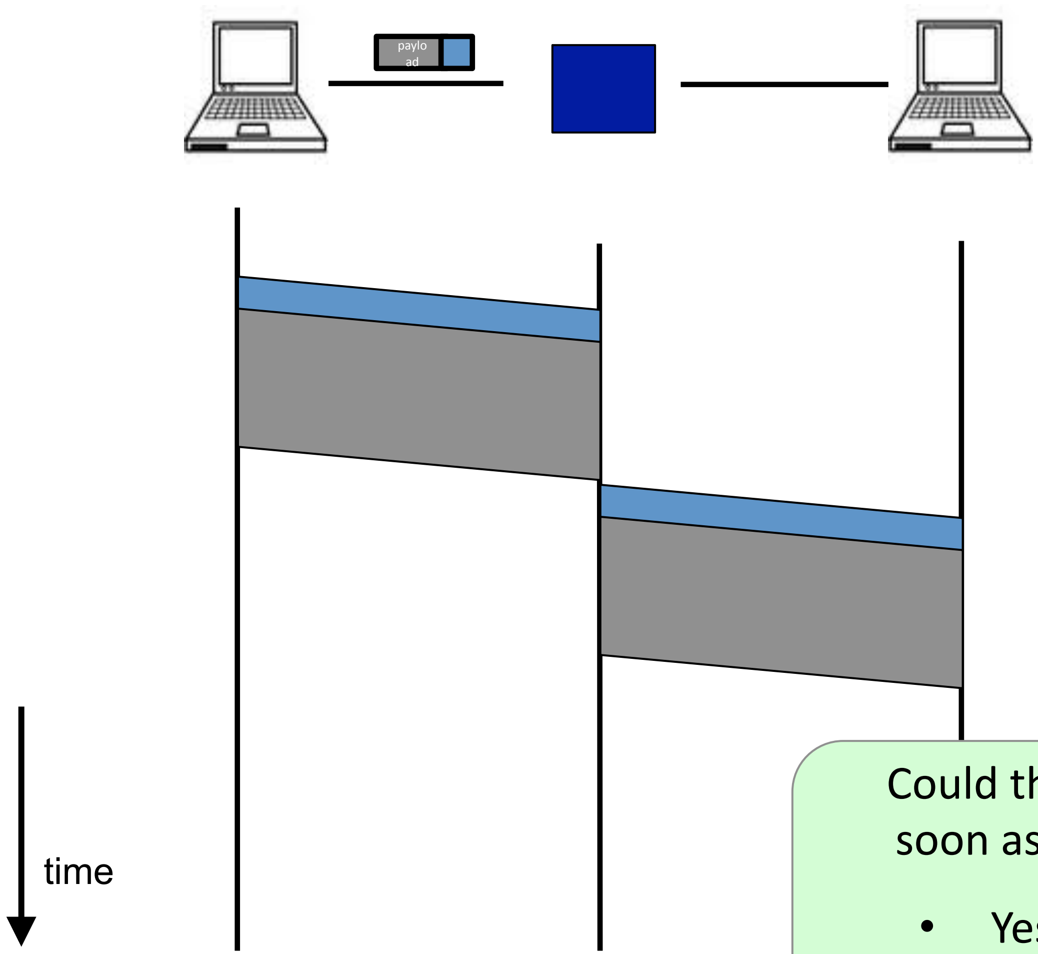
Timing in Packet Switching



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

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

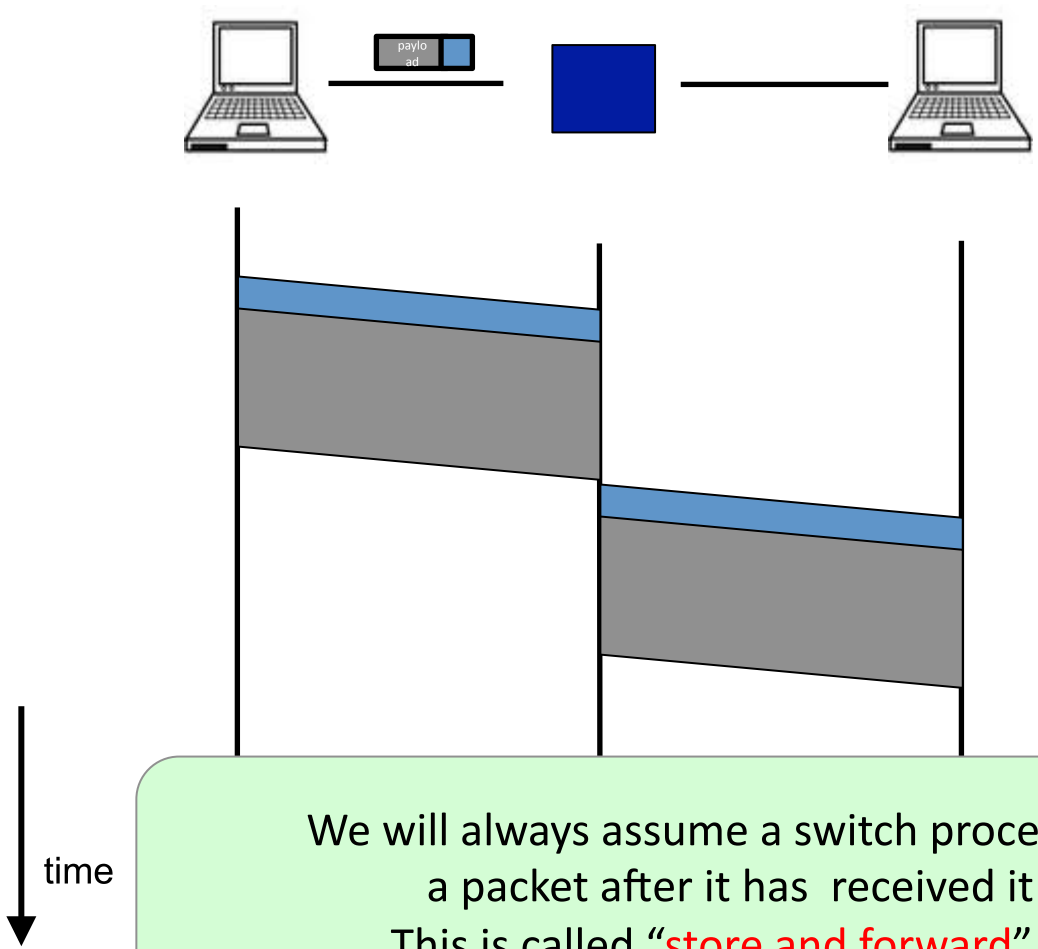
Timing in Packet Switching



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

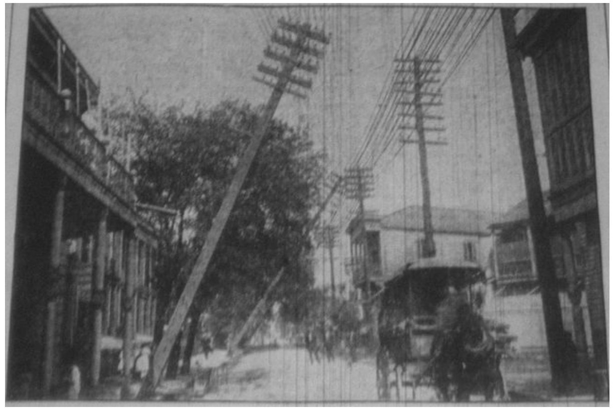
Packet Switching

- 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)

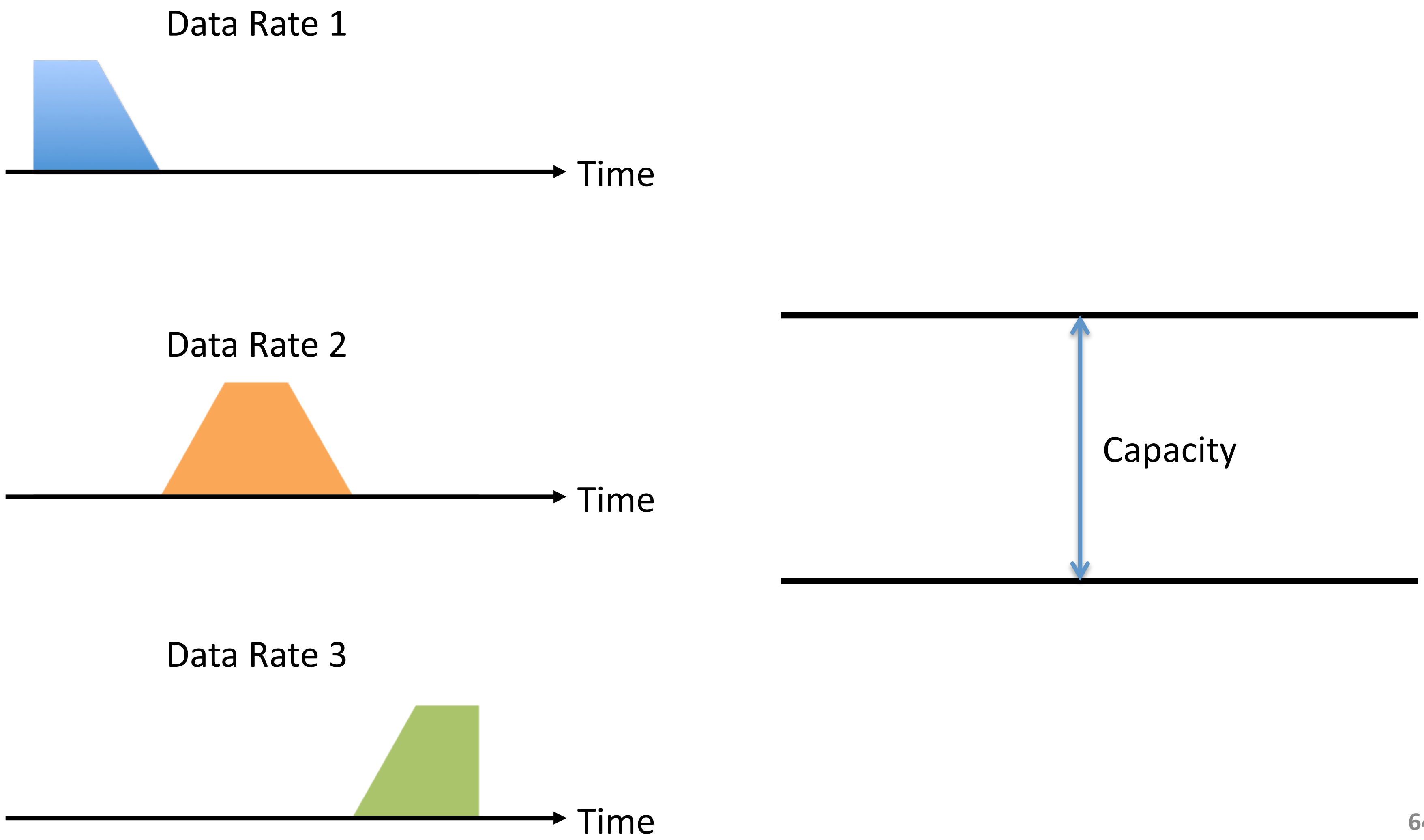
Multiplexing



Sharing makes things efficient (cost less)

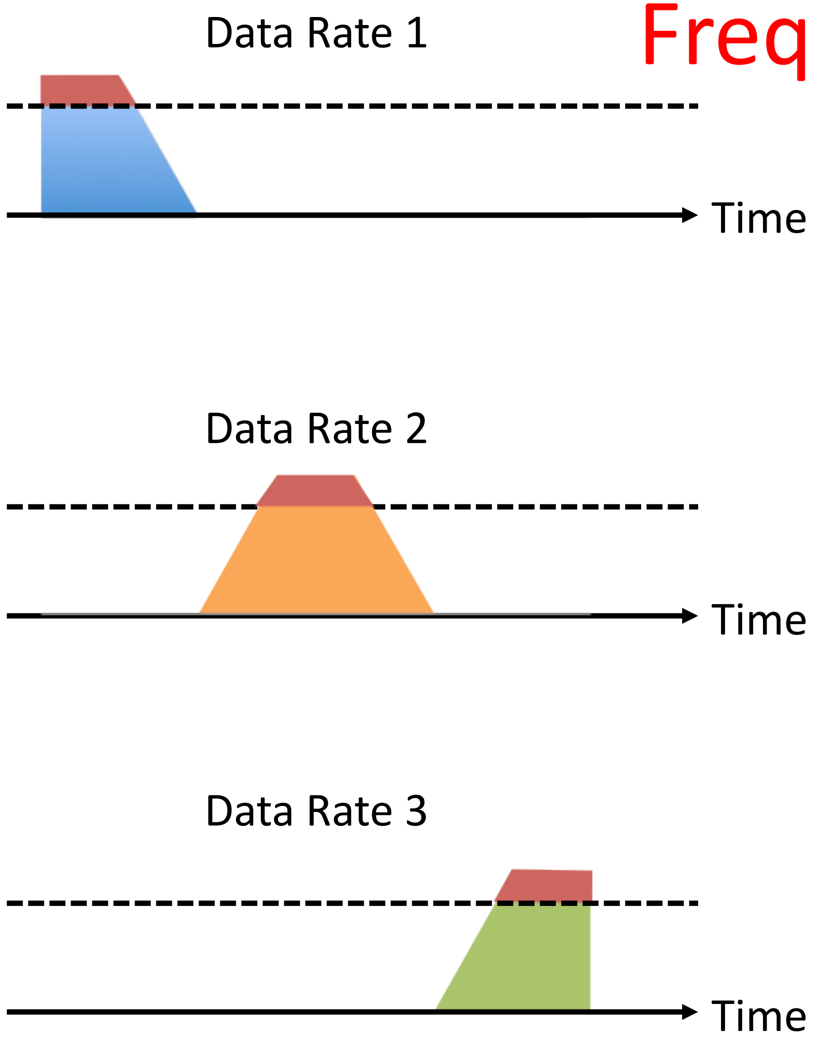
- One airplane/train for 100 people
- One telephone for many calls
- One lecture theatre for many classes
- One computer for many tasks
- One network for many computers
- One datacenter many applications

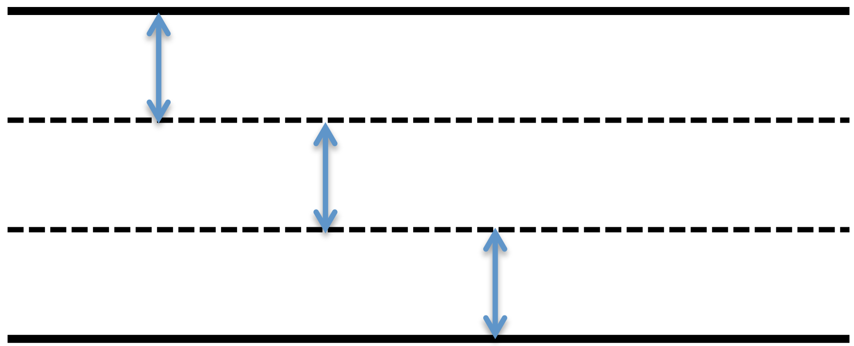
Three Flows with Bursty Traffic



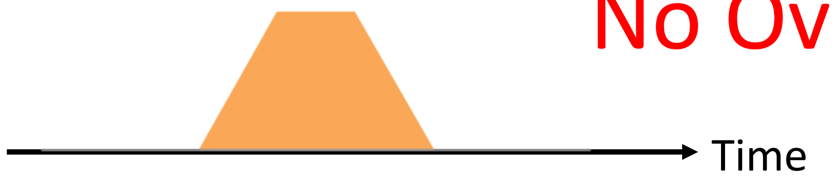
When Each Flow Gets 1/3rd of Capacity

Frequent Overloading





When Flows Share Total Capacity

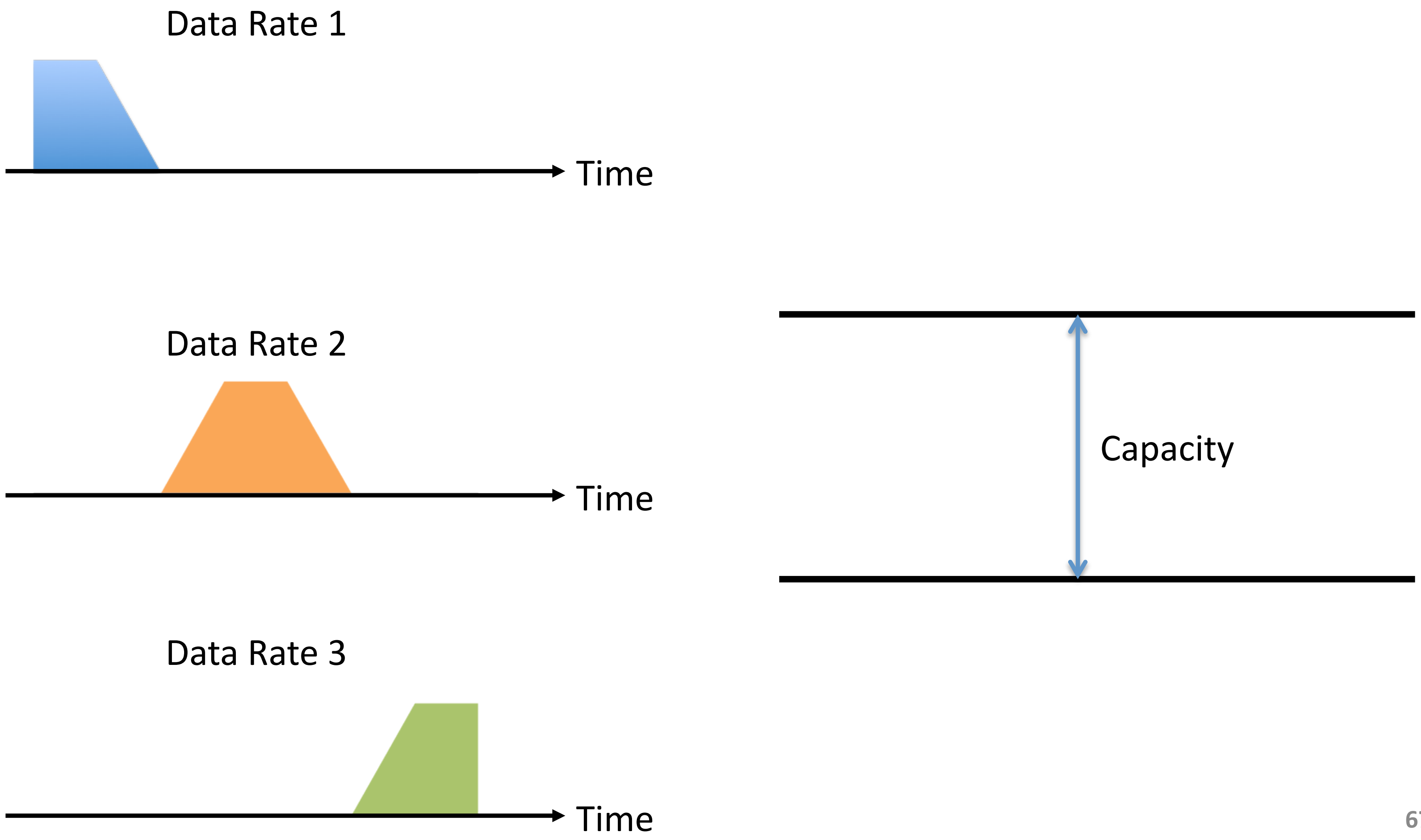


No Overloading

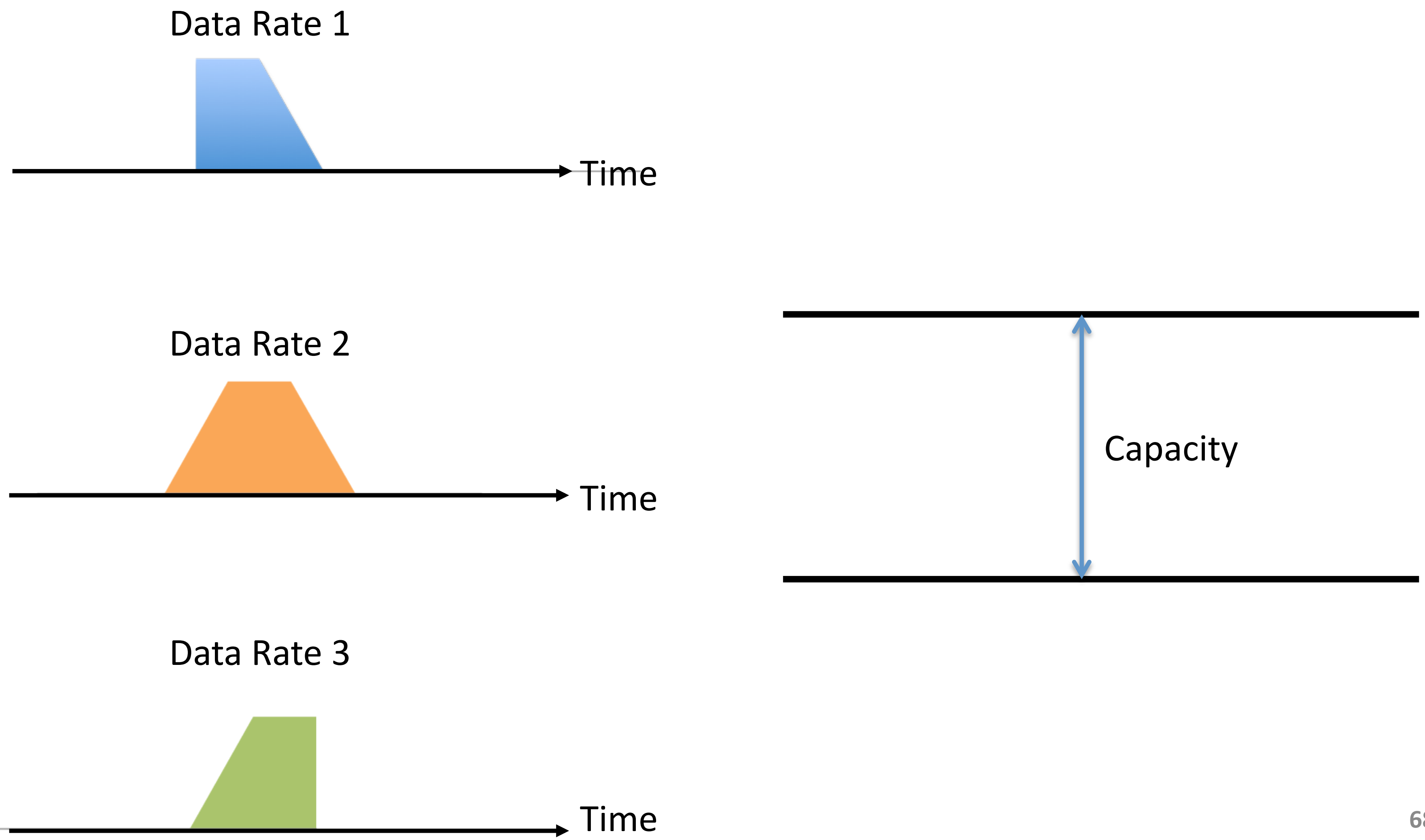
Statistical multiplexing relies on the assumption that not all flows burst at the same time.

Very similar to insurance, and has same failure case

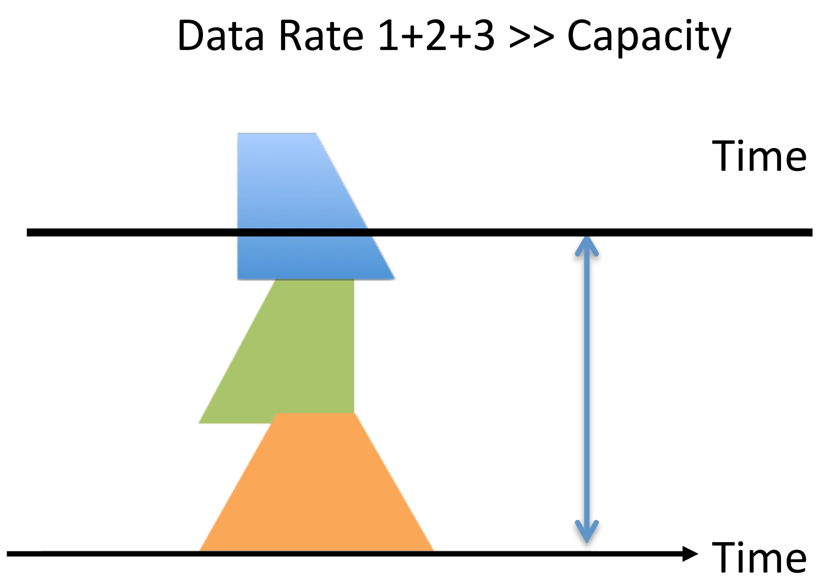
Three Flows with Bursty Traffic

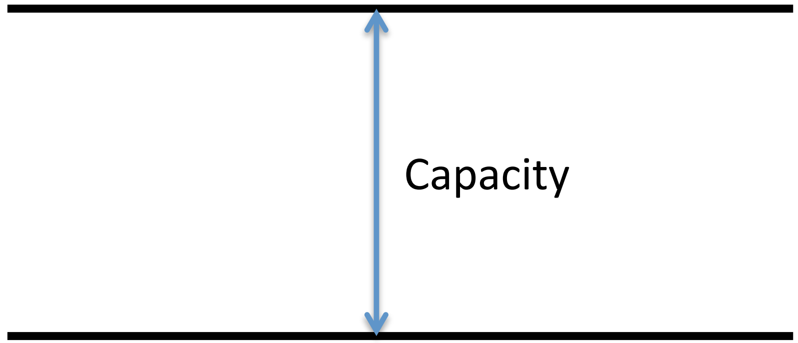


Three Flows with Bursty Traffic



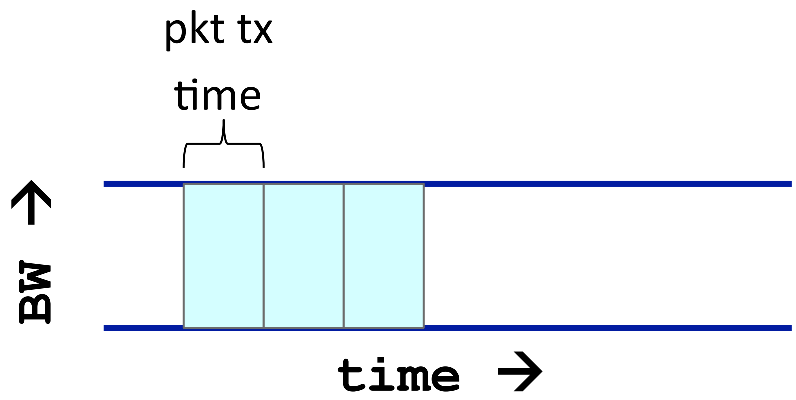
Three Flows with Bursty Traffic



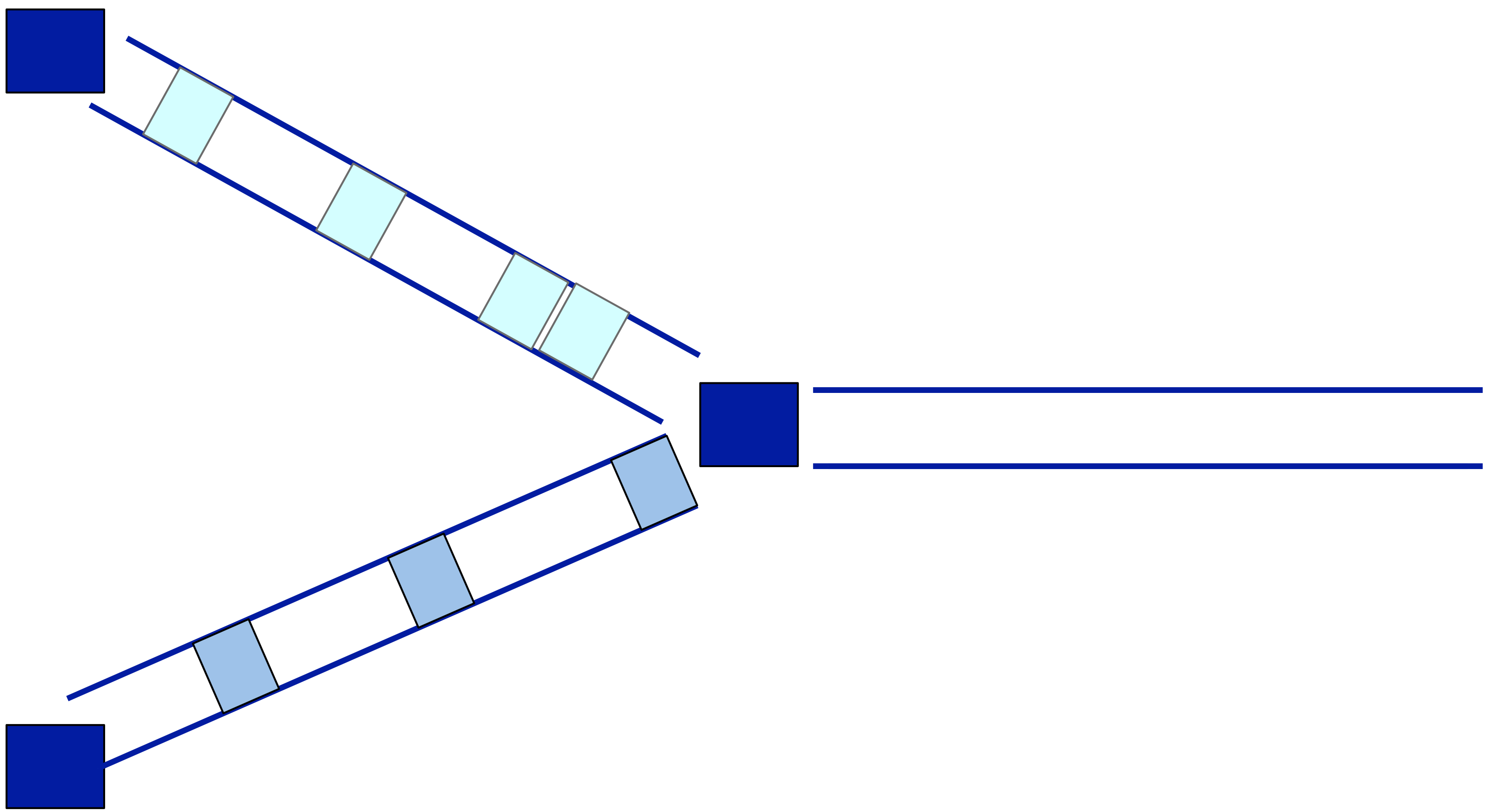


What do we do under overload?

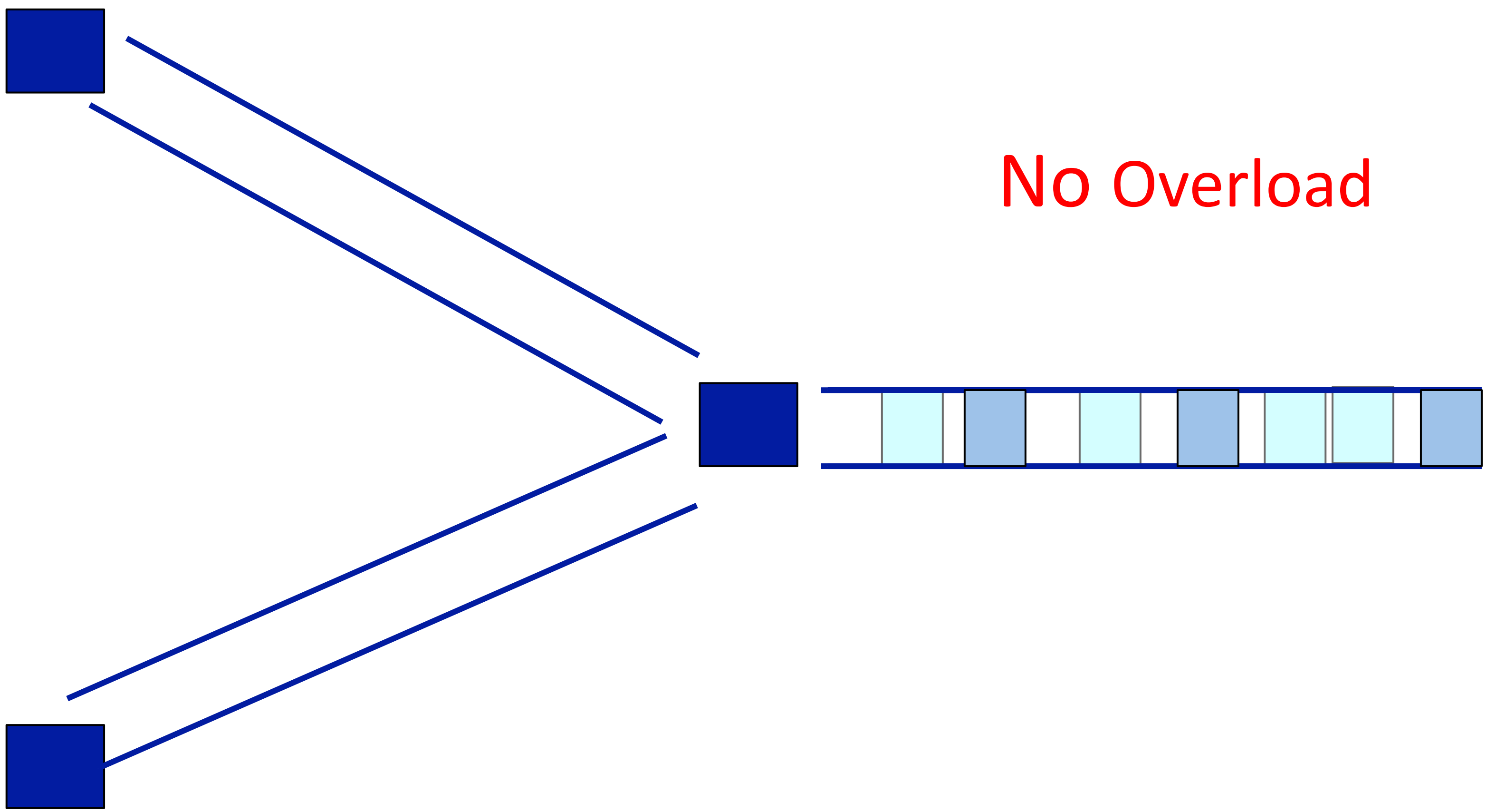
Statistical multiplexing: pipe view



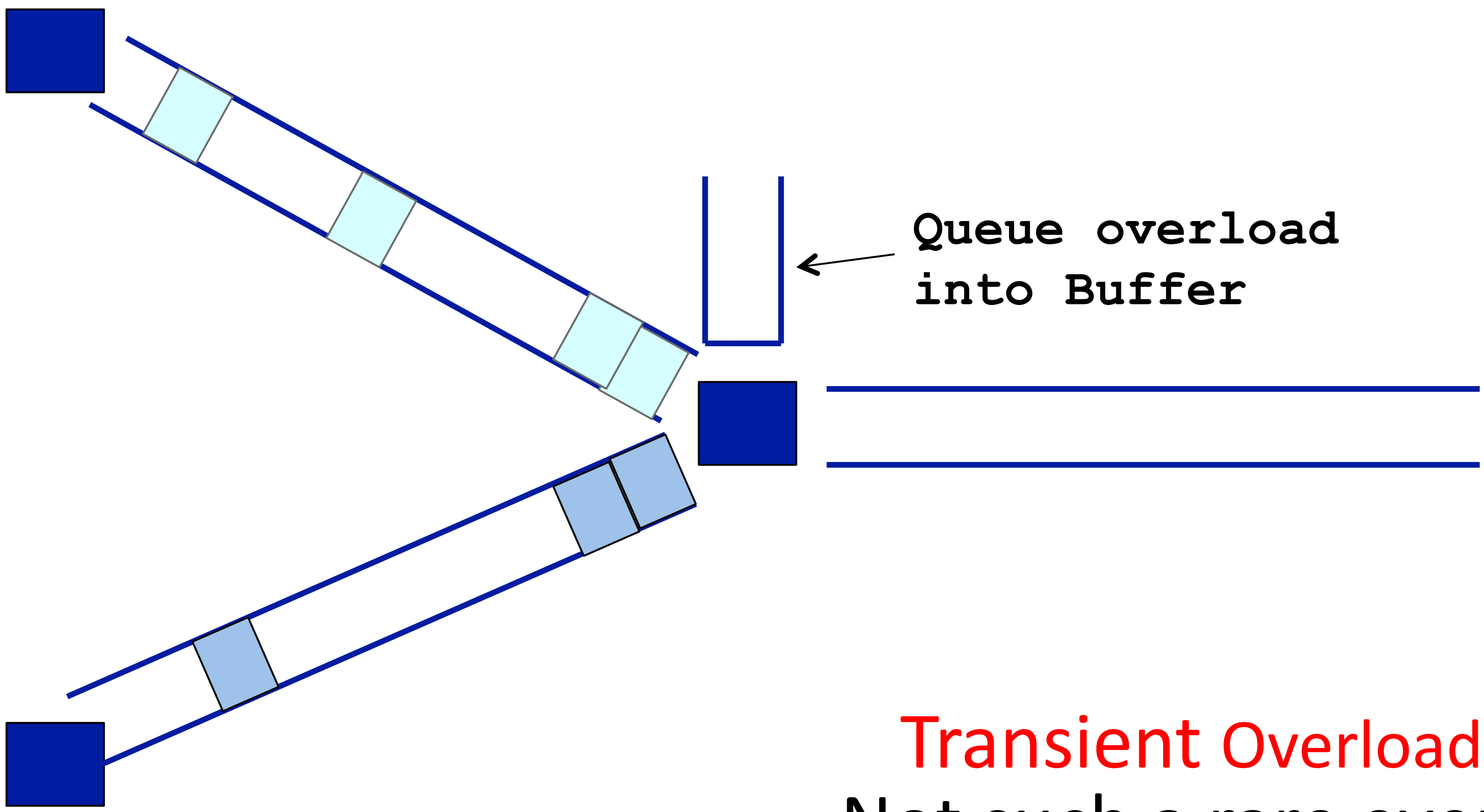
Statistical multiplexing: pipe view



Statistical multiplexing: pipe view



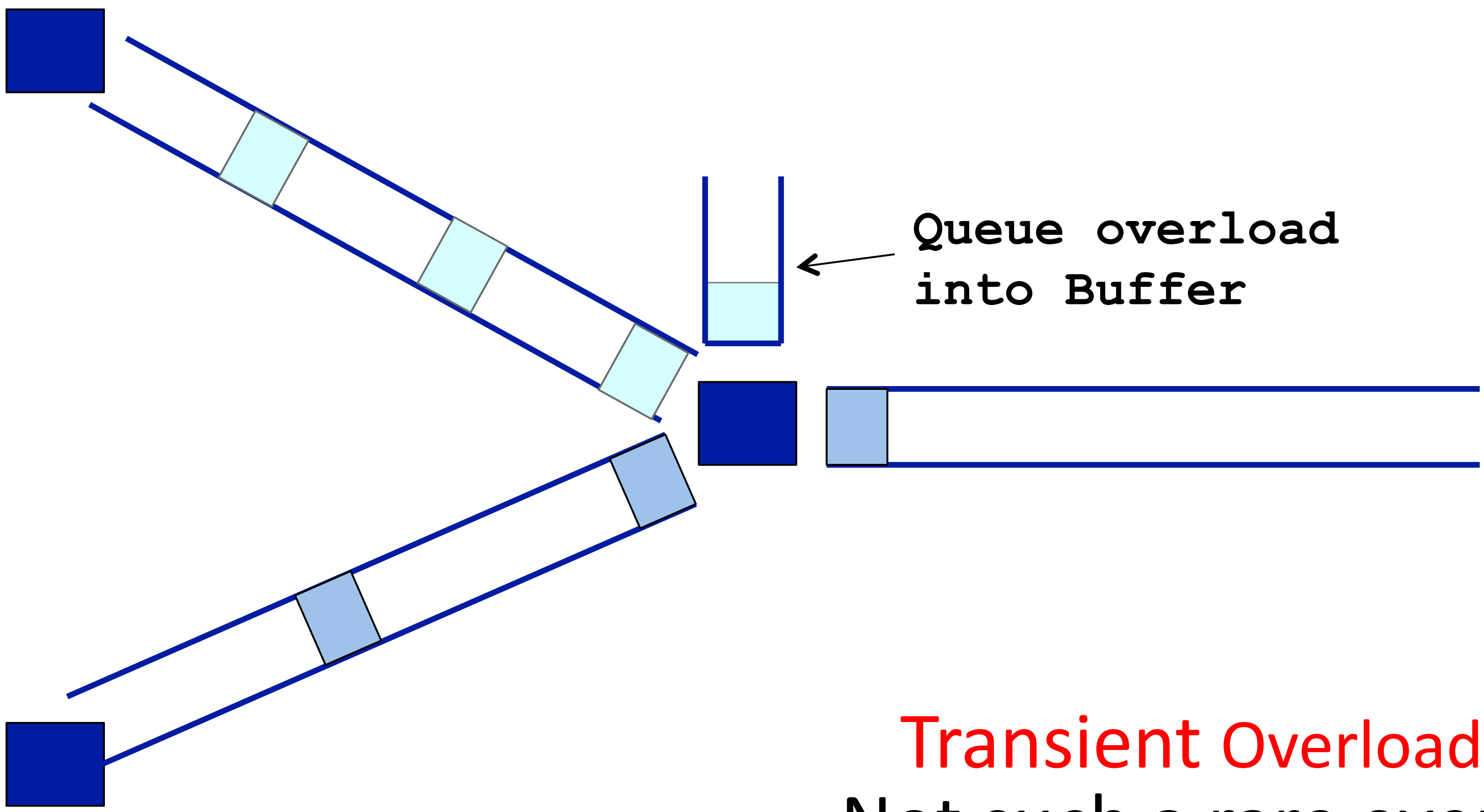
Statistical multiplexing: pipe view



Transient Overload

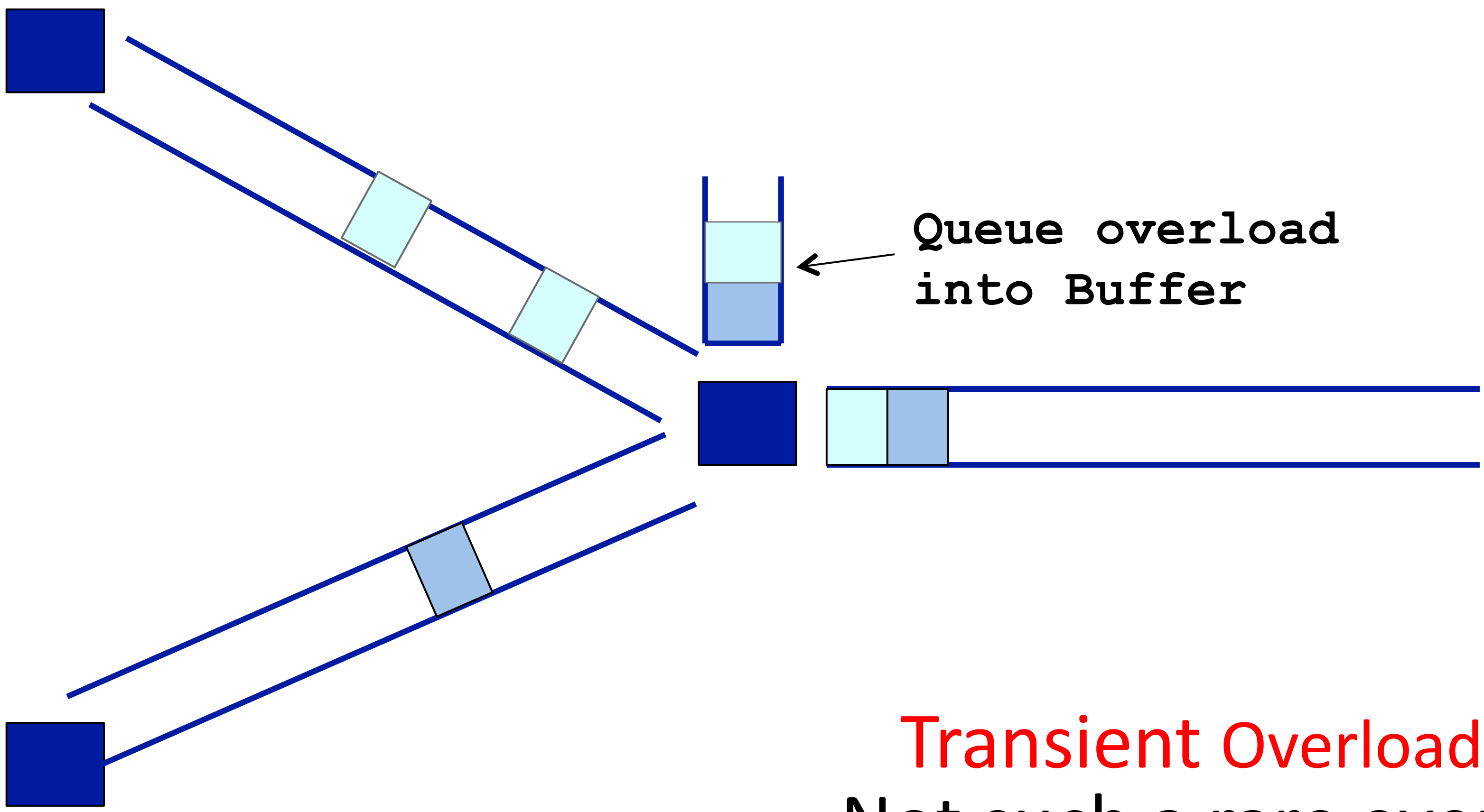
Not such a rare event

Statistical multiplexing: pipe view



Transient Overload
Not such a rare event

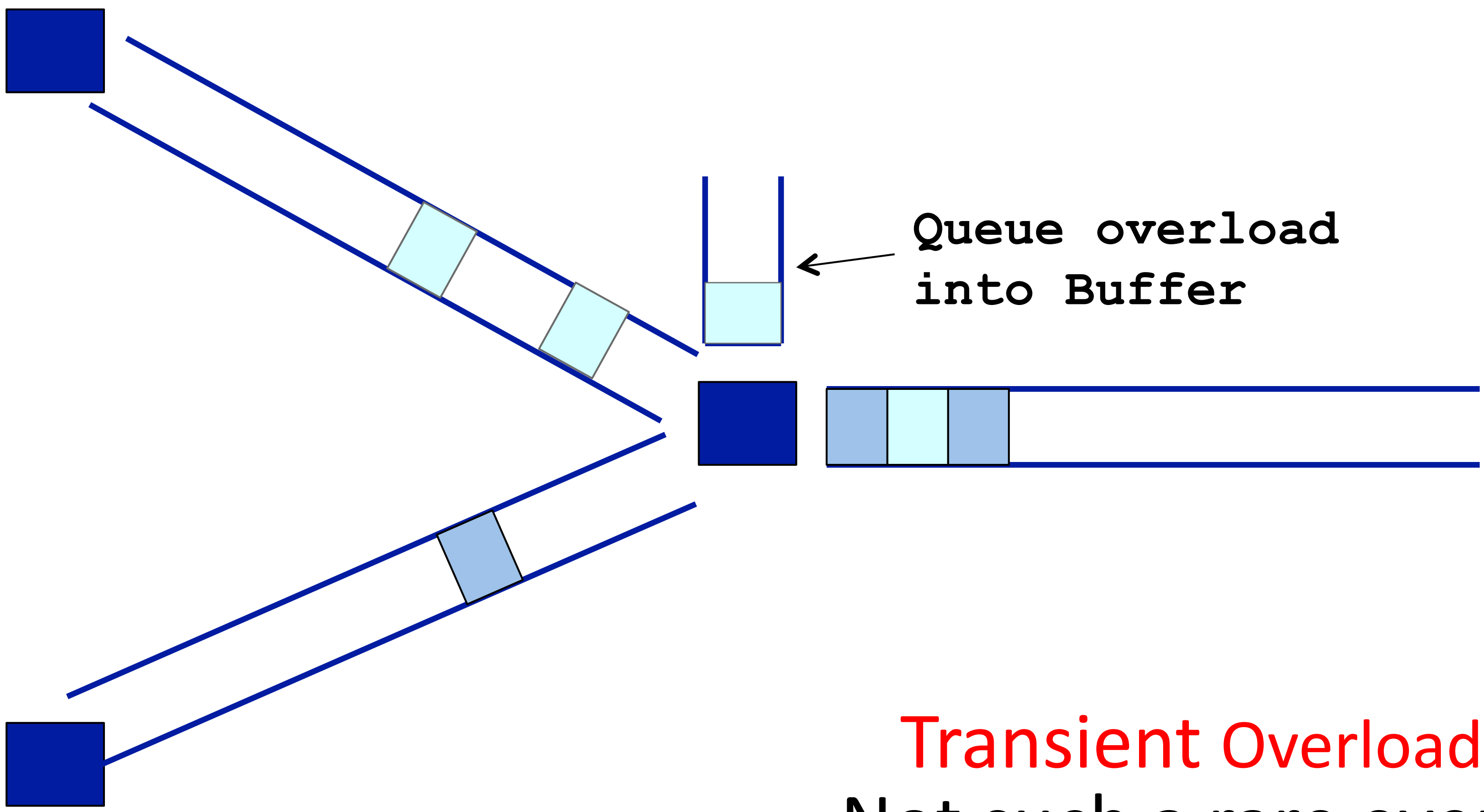
Statistical multiplexing: pipe view



Transient Overload

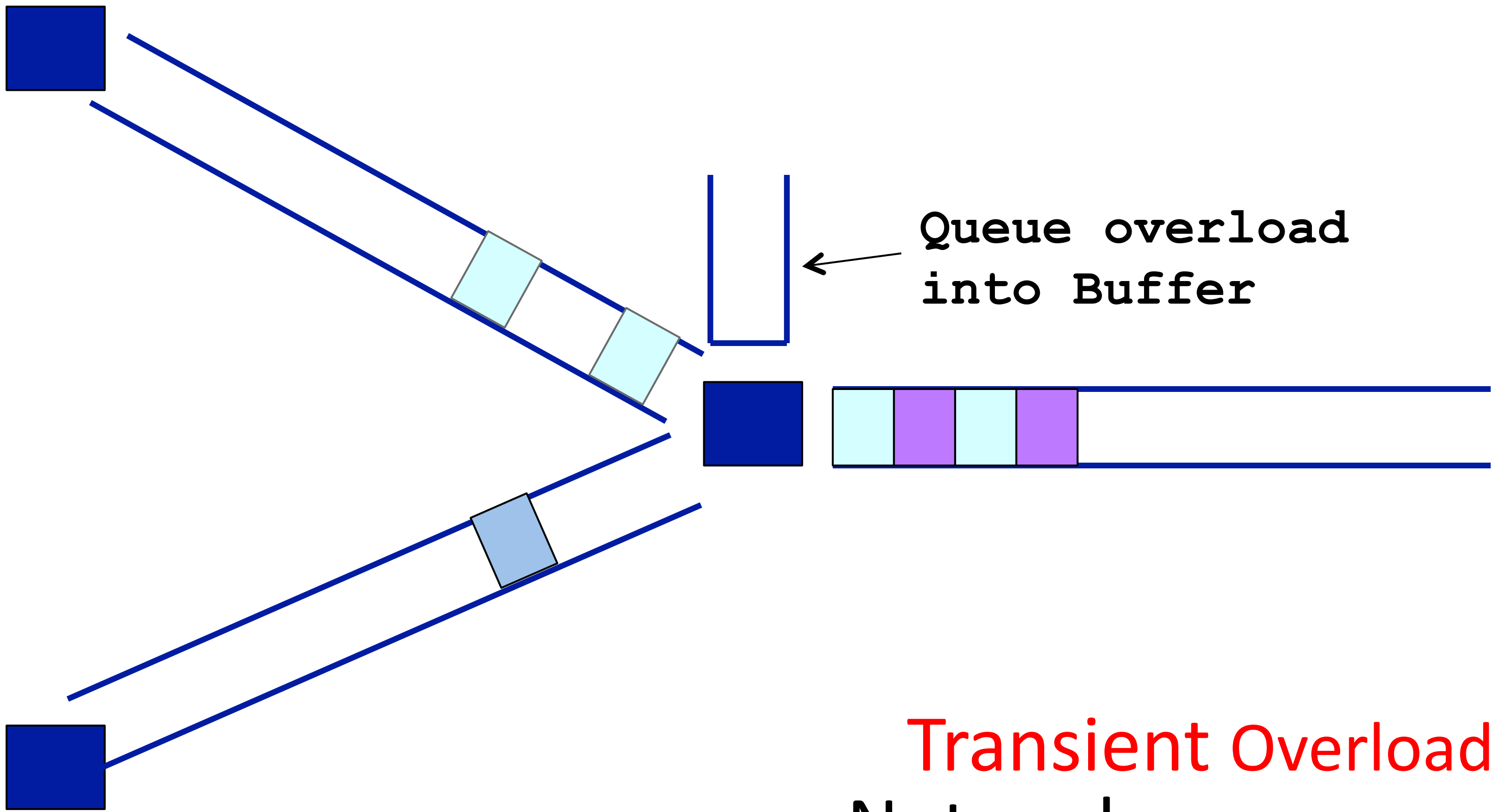
Not such a rare event

Statistical multiplexing: pipe view



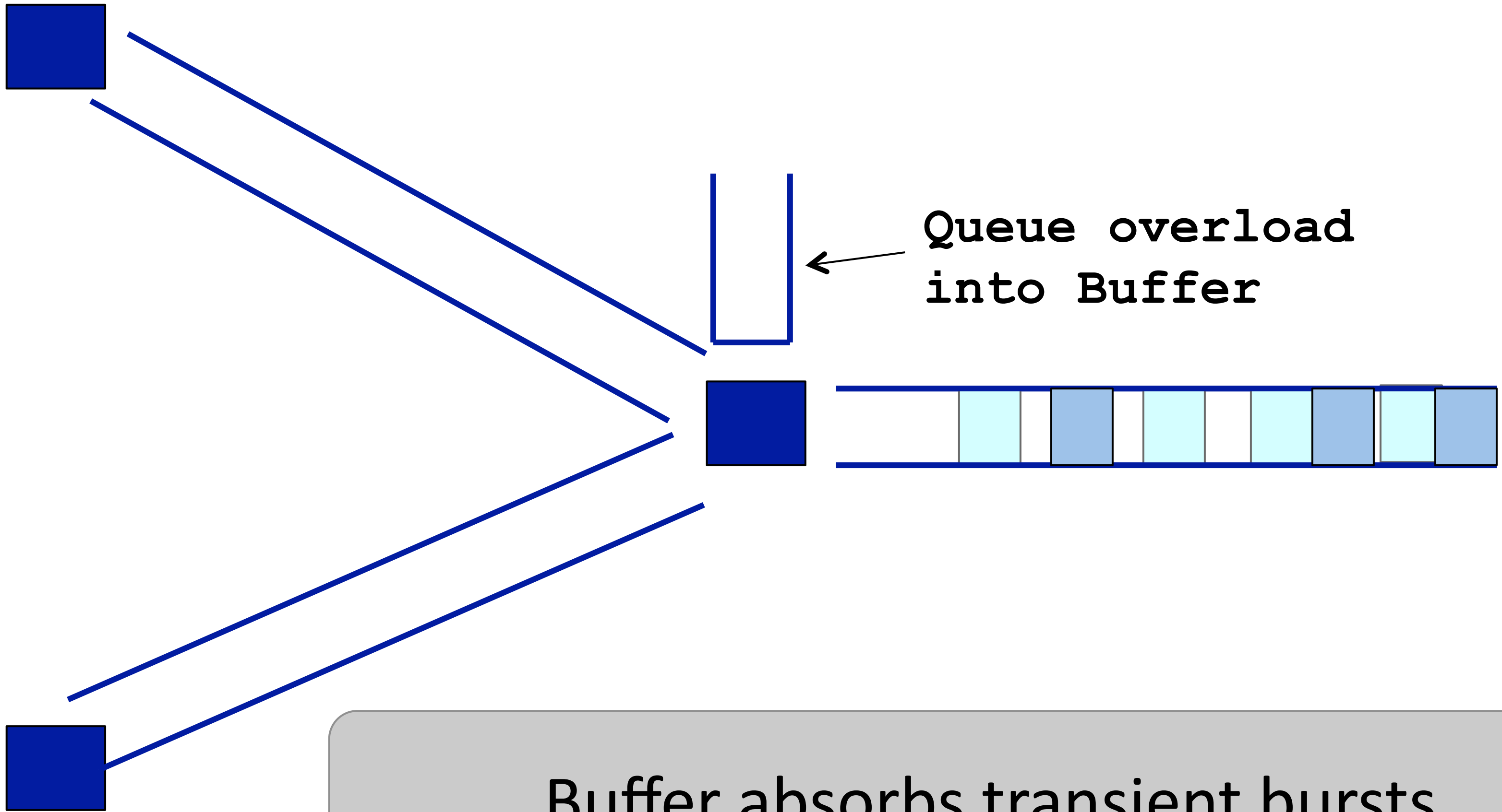
Transient Overload
Not such a rare event

Statistical multiplexing: pipe view



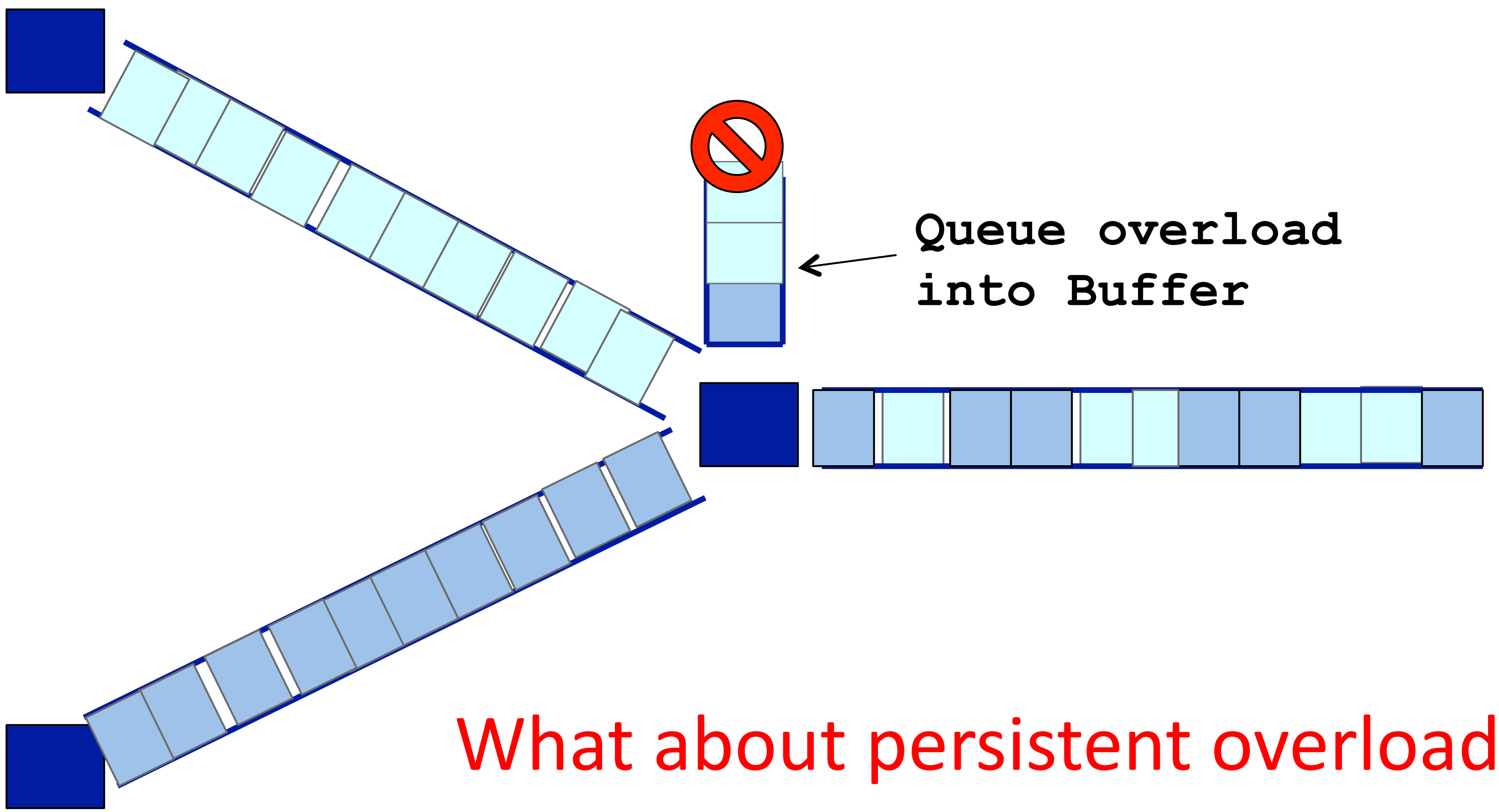
Transient Overload
Not such a rare event

Statistical multiplexing: pipe view



Buffer absorbs transient bursts

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 (*)

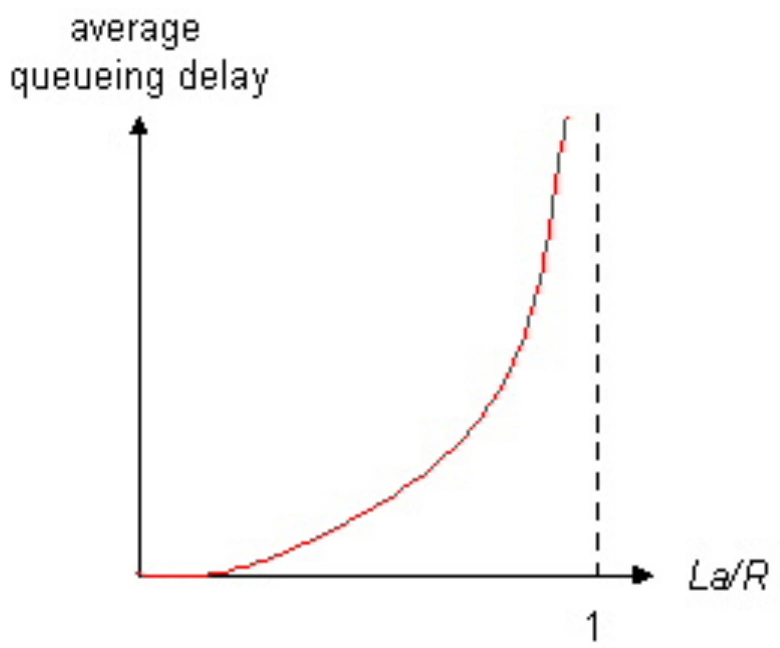
- Queuing delay caused by “packet interference”

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

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

Queuing delay

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

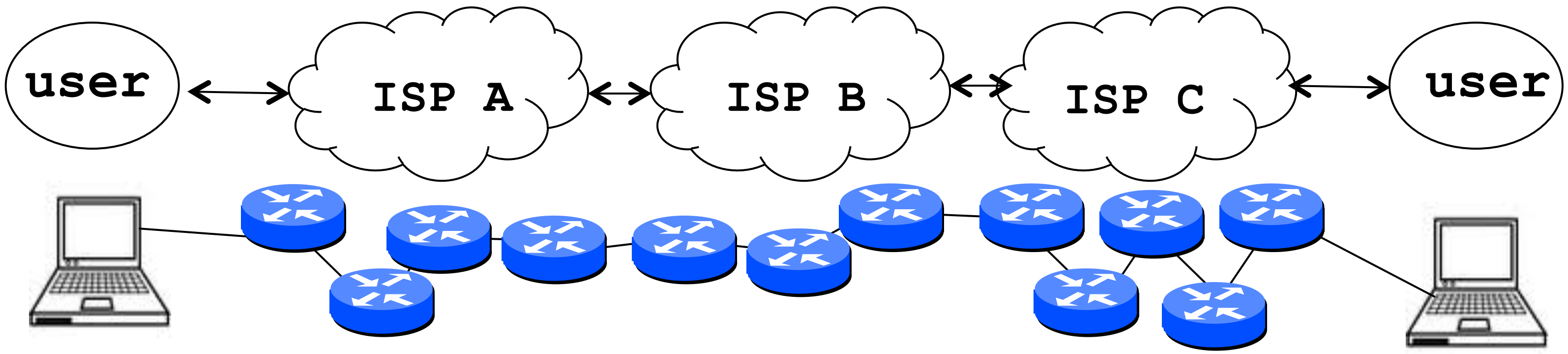


traffic intensity = La/R

- ❑ 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

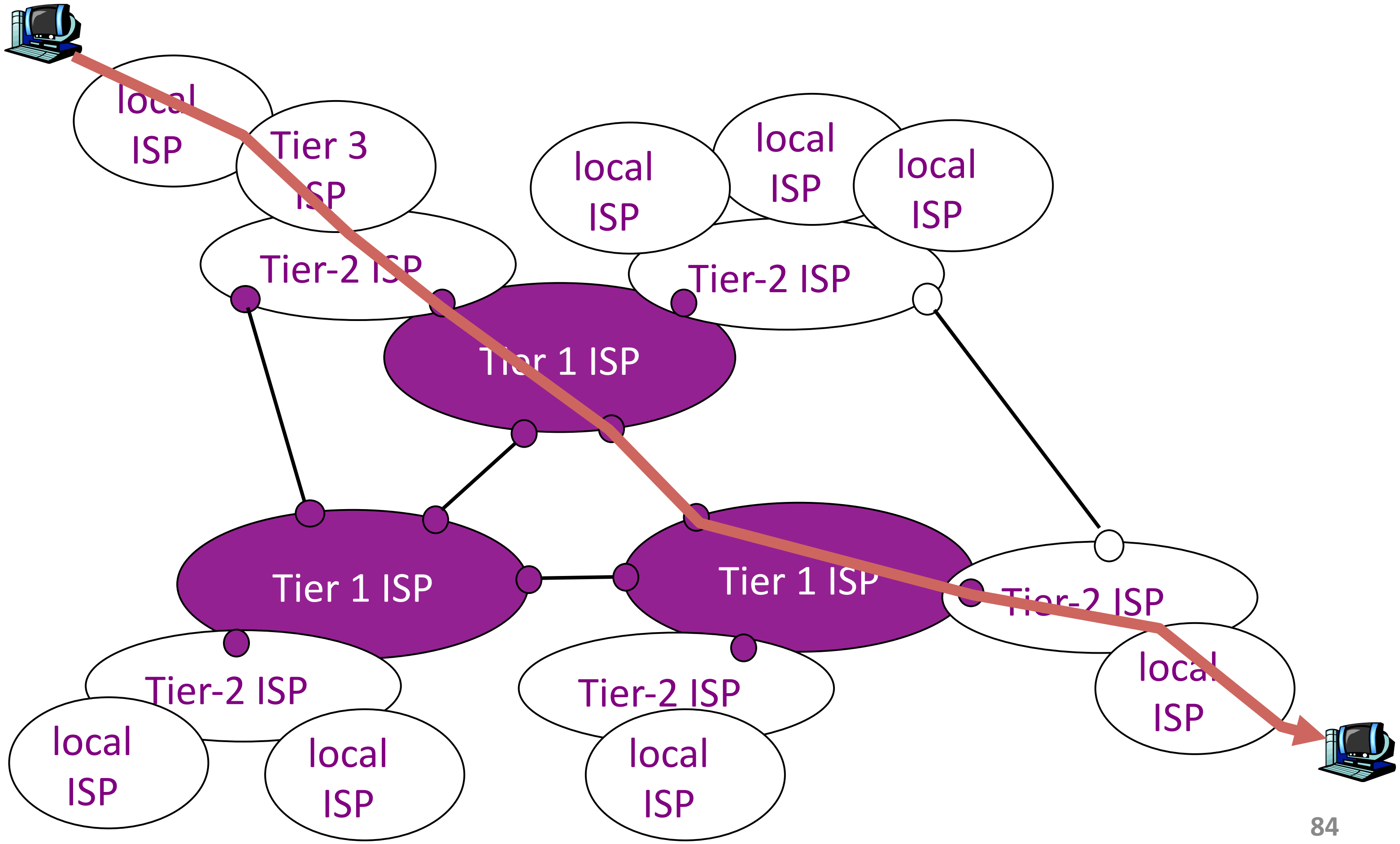
```
traceroute munnari.oz.au
traceroute to munnari.oz.au (202.29.151.3), 30 hops max, 60 byte packets
 1  gatwick.net.cl.cam.ac.uk (128.232.32.2)  0.416 ms  0.384 ms  0.427 ms
 2  cl-sby.route-nwest.net.cam.ac.uk (193.60.89.9)  0.393 ms  0.440 ms  0.494 ms
 3  route-nwest.route-mill.net.cam.ac.uk (192.84.5.137)  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 (between hops 10 and 11)

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

Internet structure: network of networks

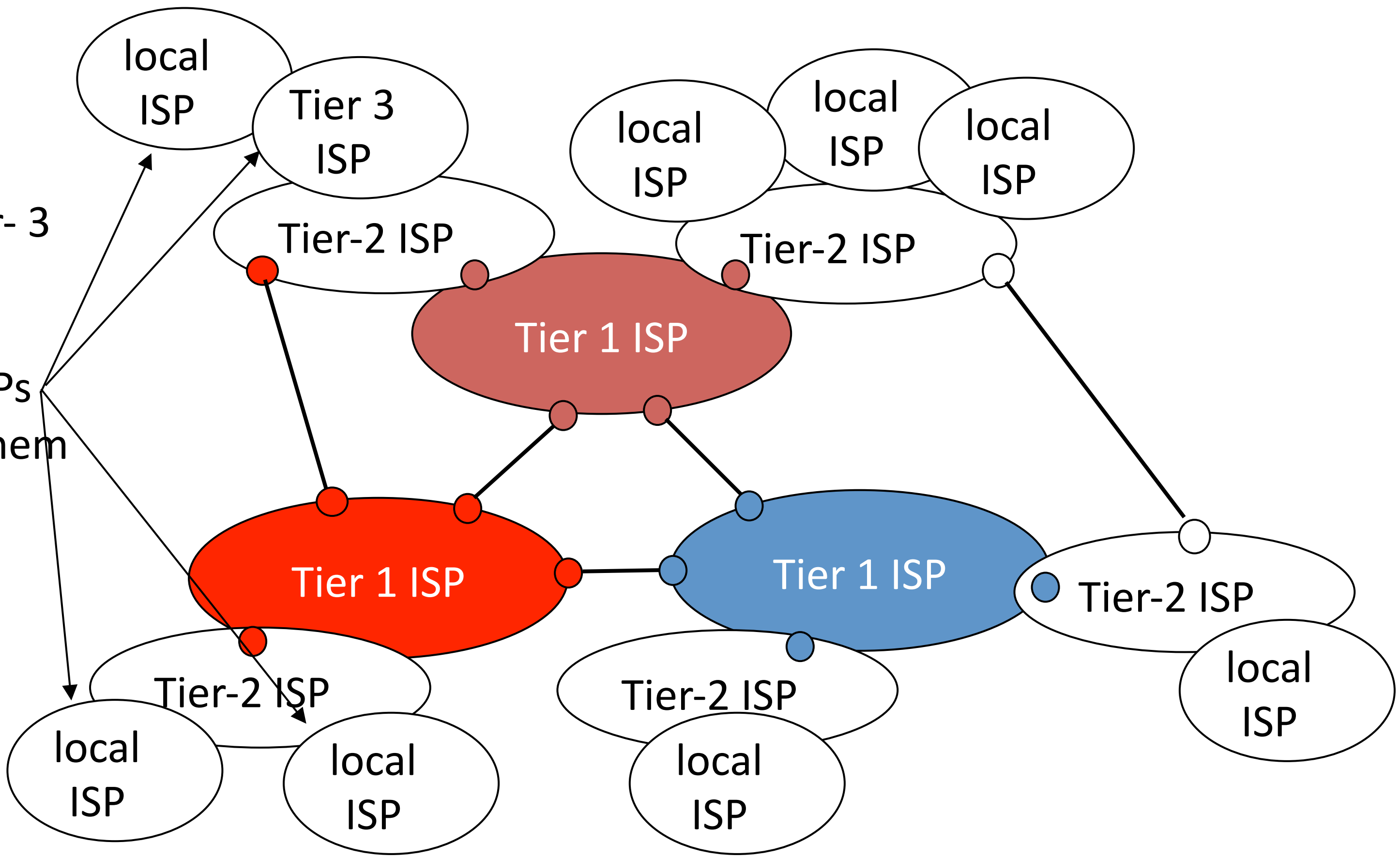
- a packet passes through many networks!



Internet structure: network of networks

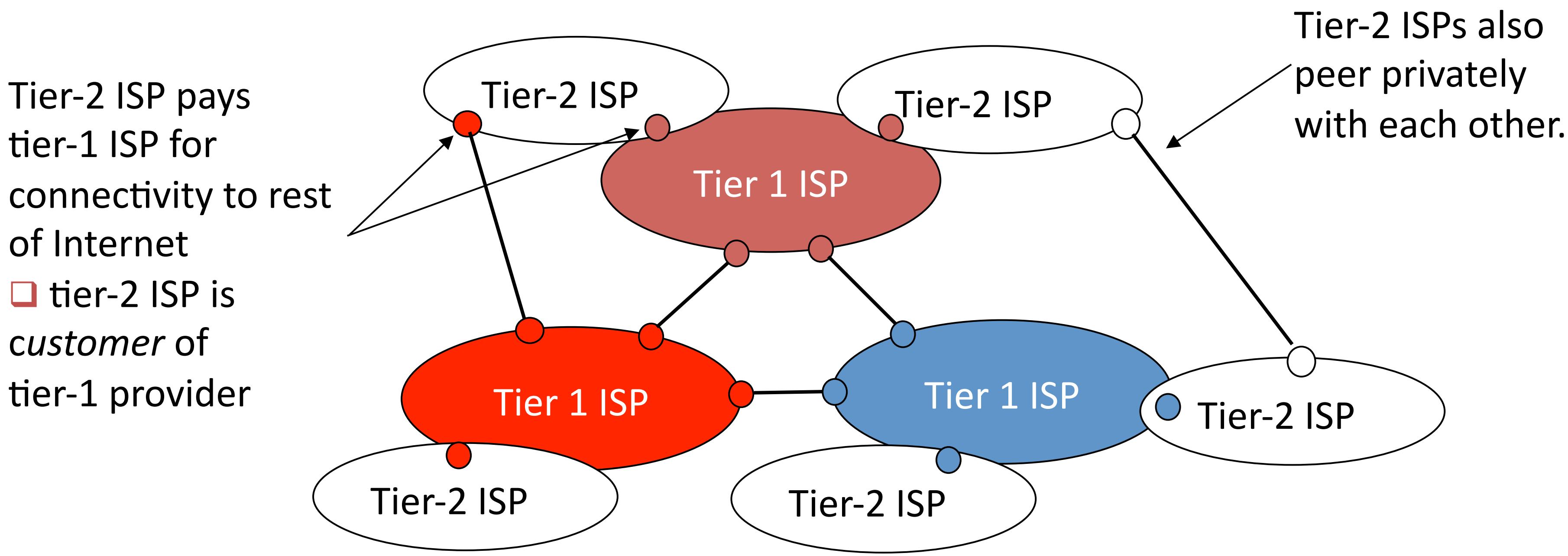
- "Tier-3" ISPs and local ISPs
  - last hop ("access") network (closest to end systems)

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



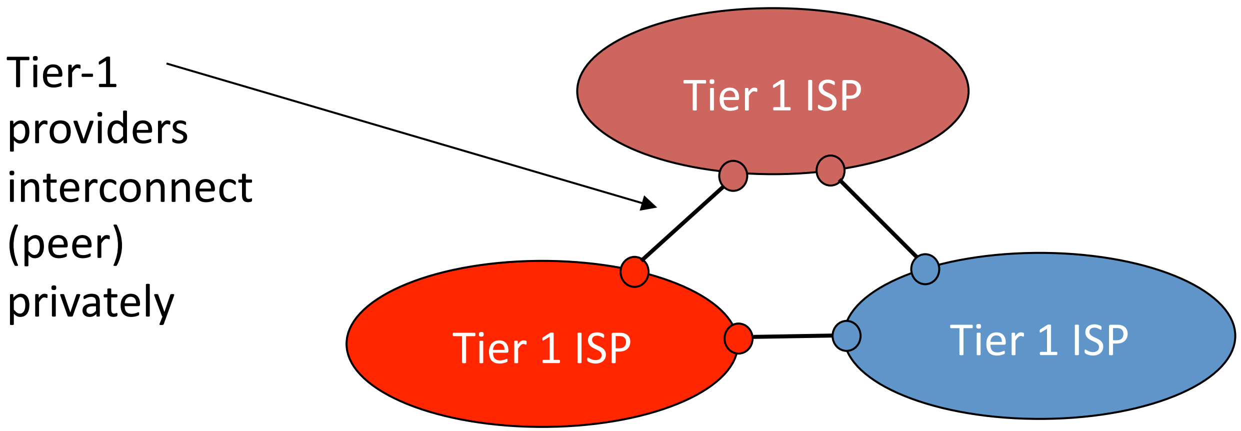
Internet structure: network of networks

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

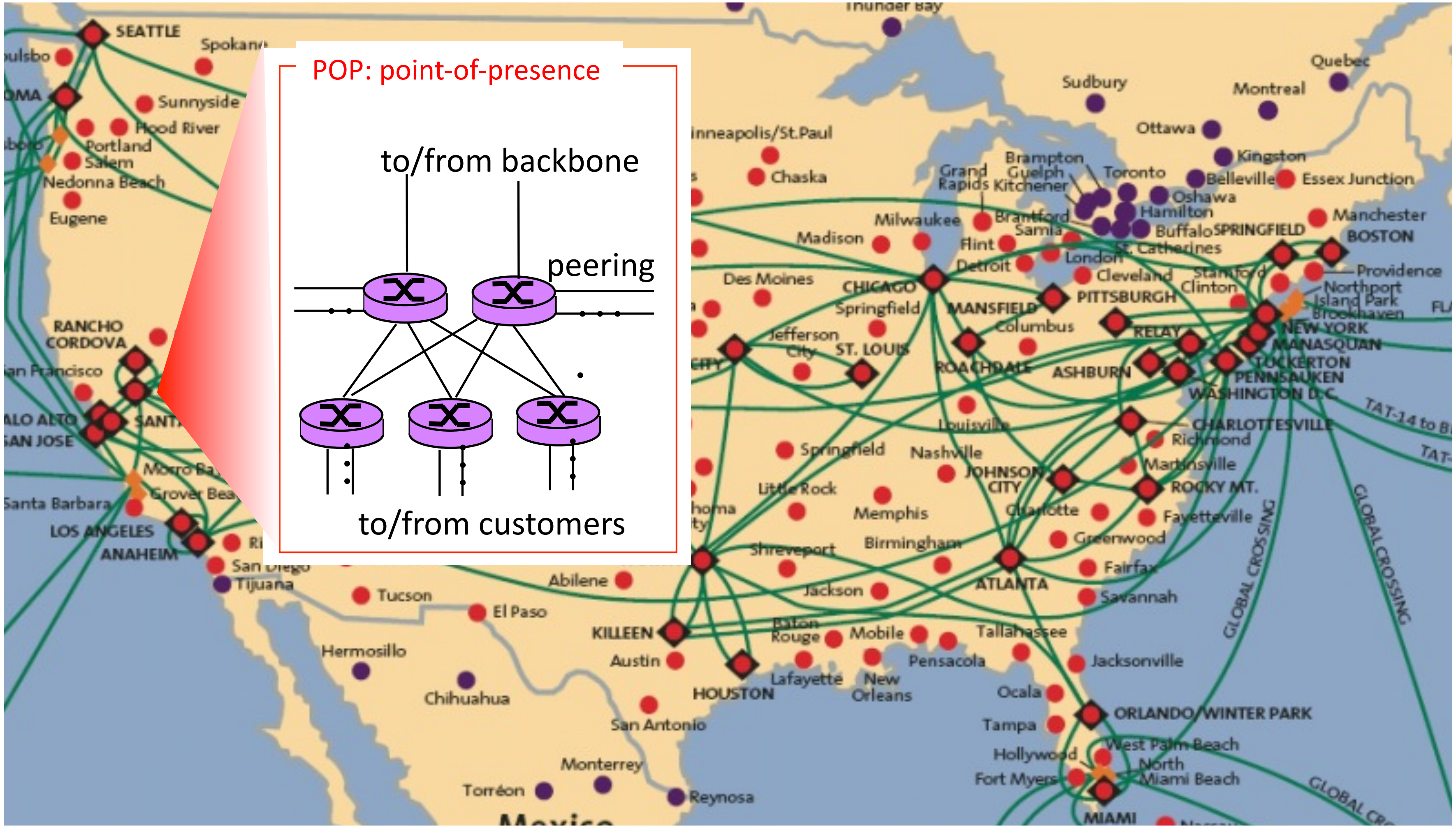


Internet structure: network of networks

- 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



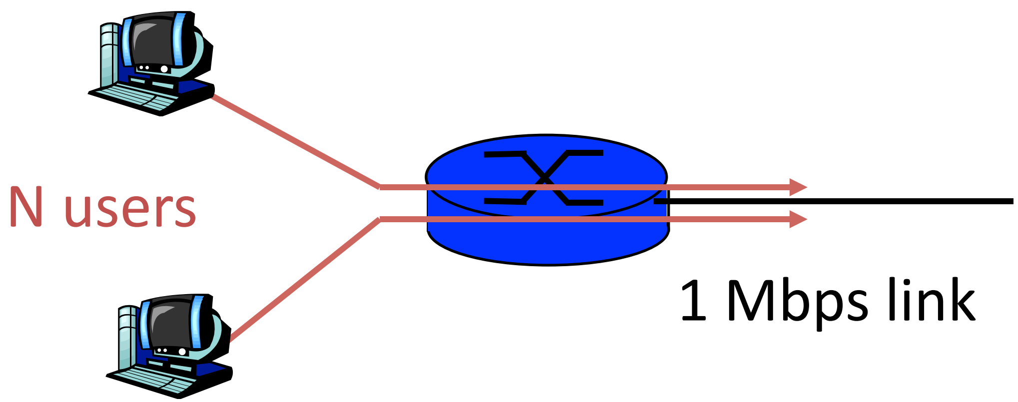
Packet Switching

- 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

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

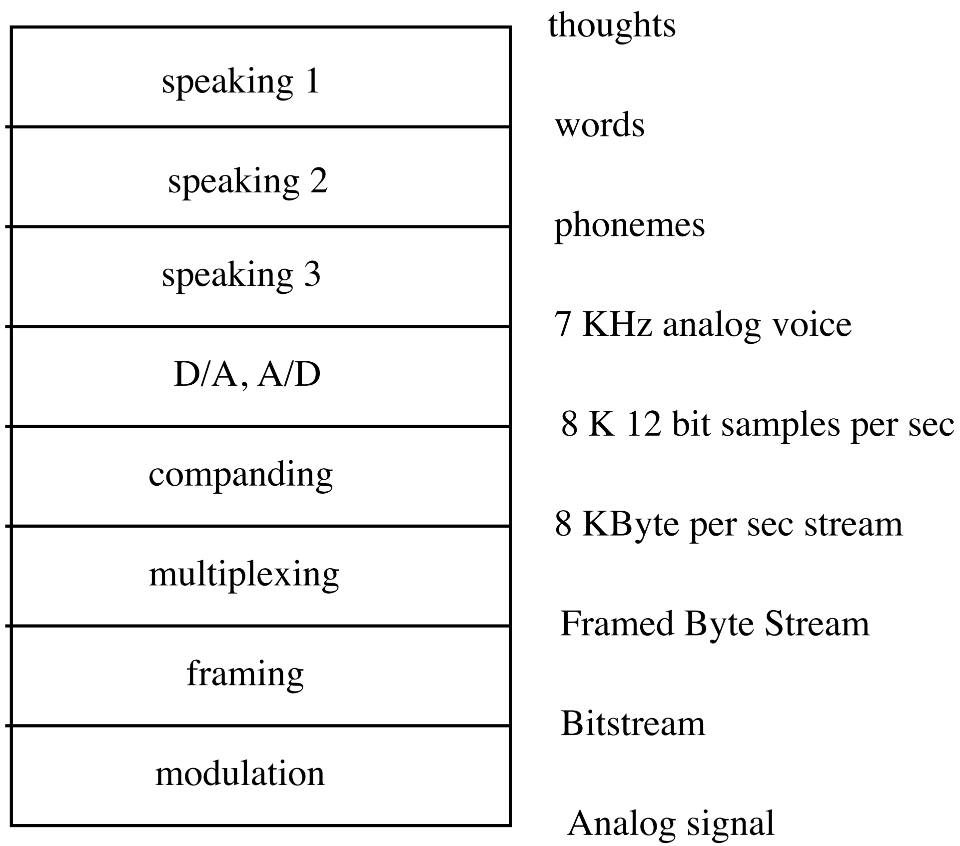
Network System Modularity

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**

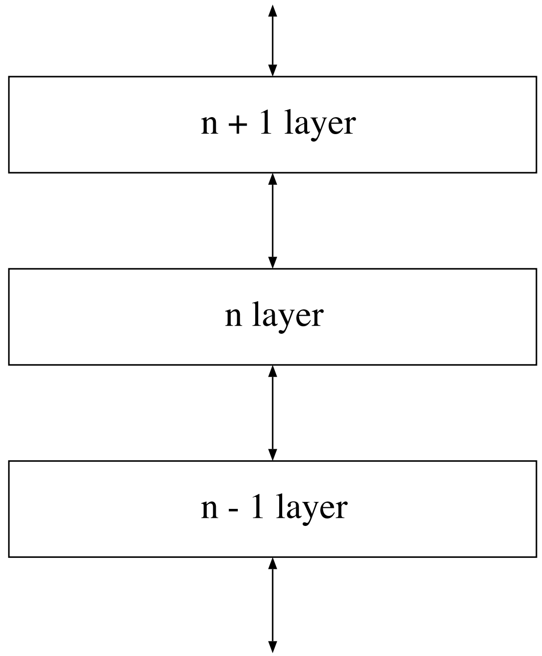
Layering Concept

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



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

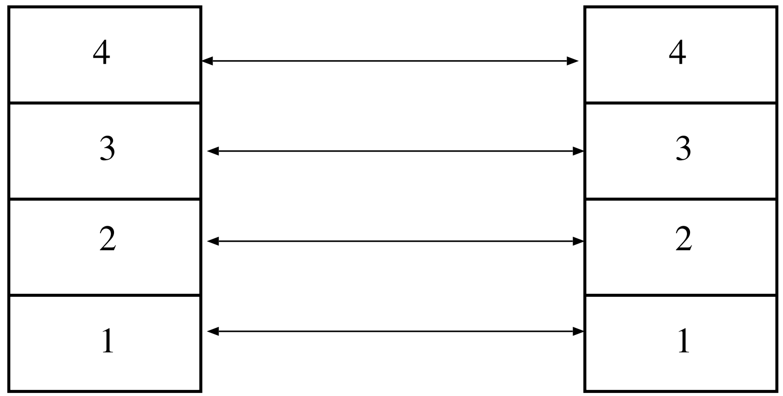
Entity – a *thing* (an independent existence)

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

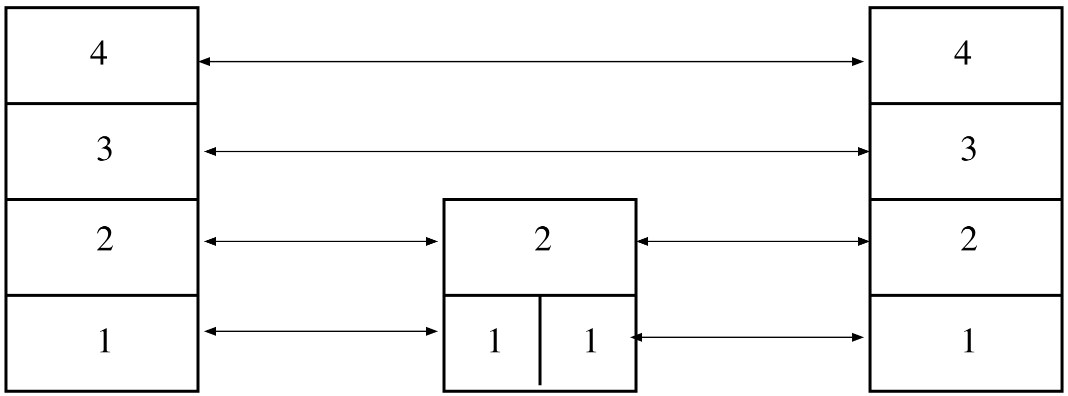
Entities usually do something useful

- 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

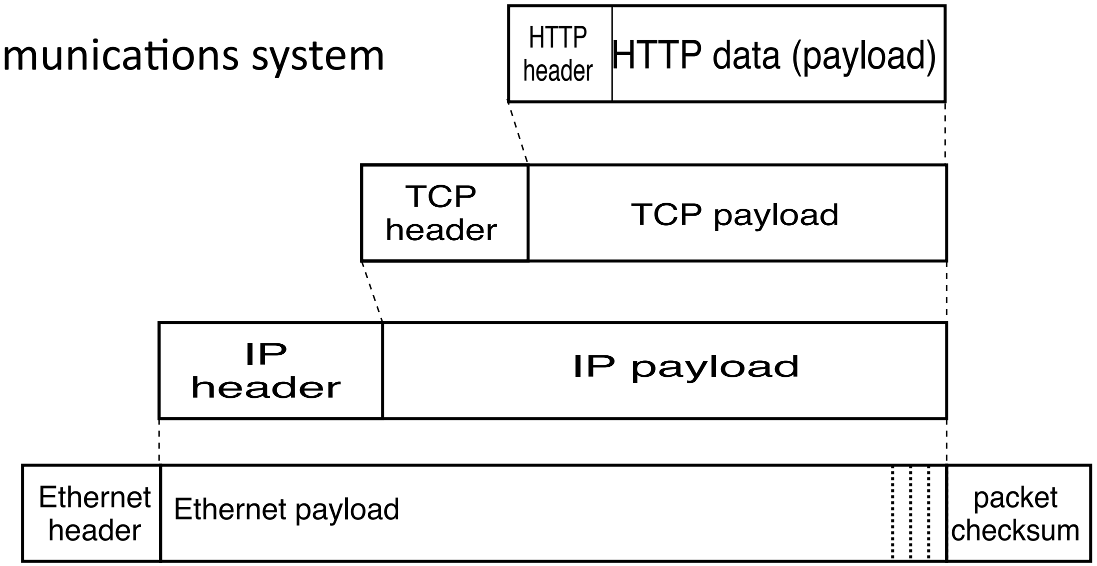
- Such embedding can be considered a form of layering
- Higher layer information is generated by stripping off headers and trailers of the current layer
- eg an IP entity only looks at the IP headers

BUT embedding is not the only form of layering

Layering is to help understand a communications system

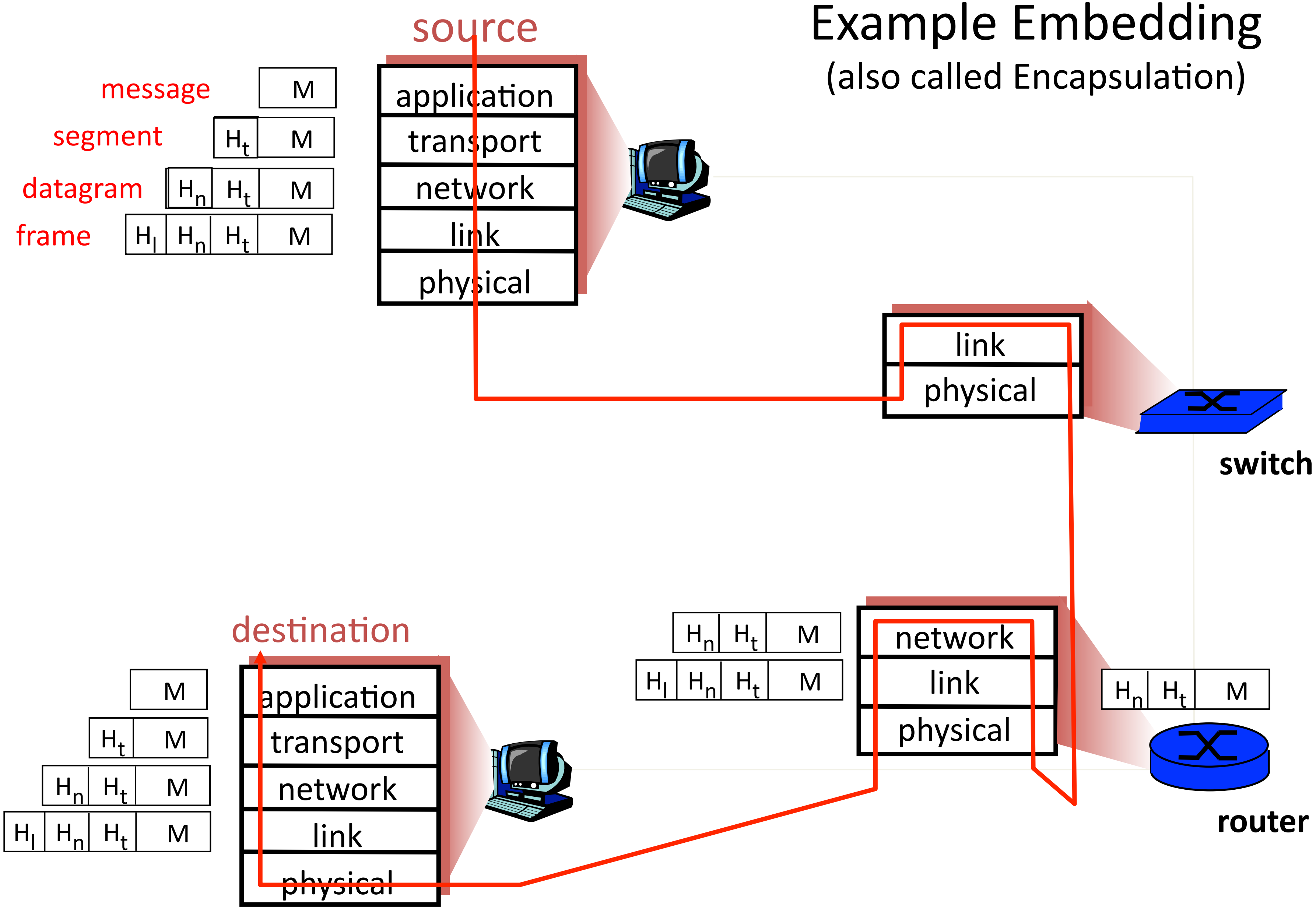
NOT

determine implementation strategy



Example Embedding

(also called Encapsulation)

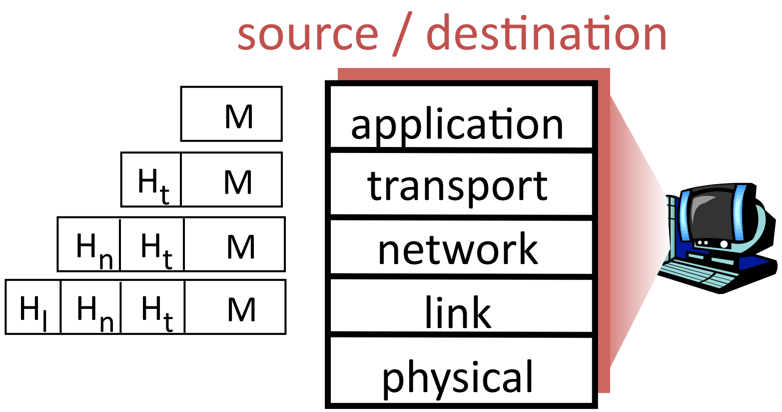


Distributing Layers Across Network

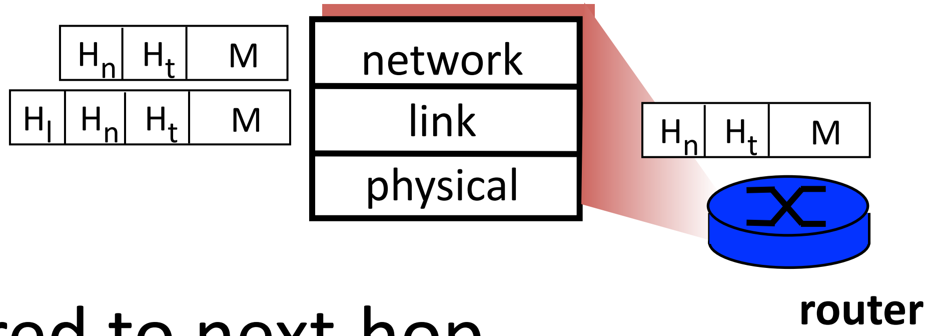
- Layers are simple if only on a single machine
  - Just stack of modules interacting with those above/below
- But we need to implement layers across machines
  - Hosts
  - Routers (switches)
- What gets implemented where?

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?

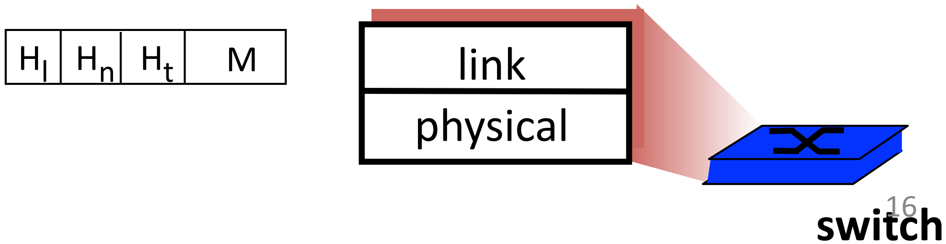


- Bits arrive on wire
  - Physical layer necessary
- Packets must be delivered to next-hop
  - Datalink layer necessary
- Routers participate in global delivery
  - Network layer necessary
- Routers don't support reliable delivery
  - Transport layer (and above) ***not*** supported

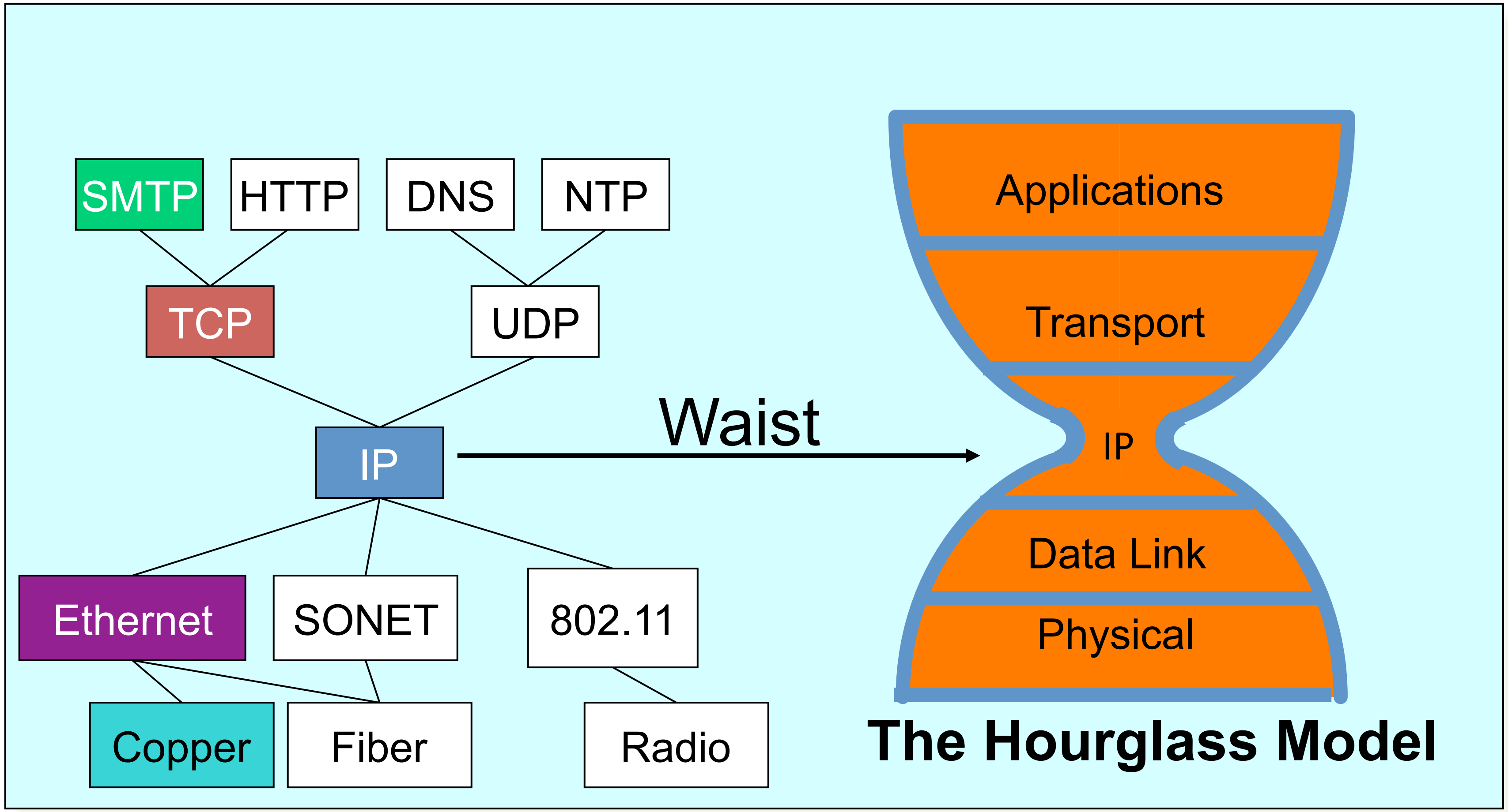
What Gets Implemented on Switches?

- 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

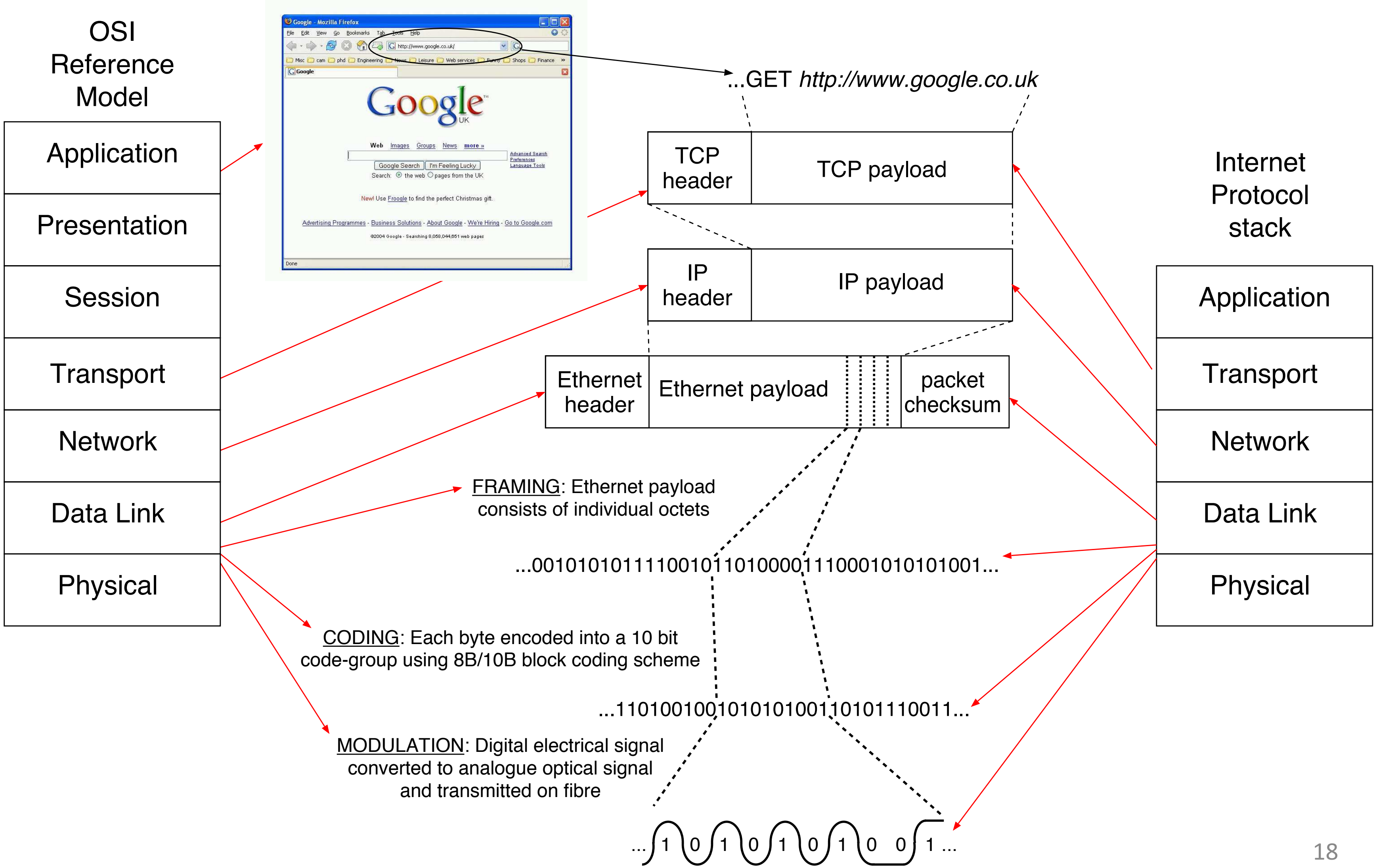


The Internet *Hourglass*



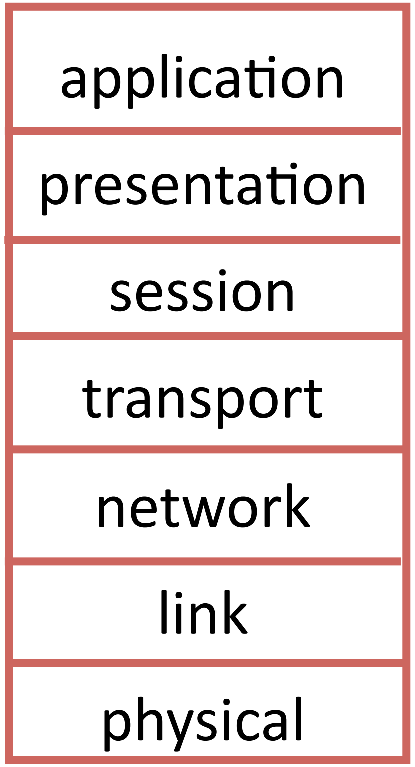
There is just one network-layer protocol, **IP**.
The "narrow waist" facilitates interoperability.

Internet protocol stack *versus* OSI Reference Model



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?



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

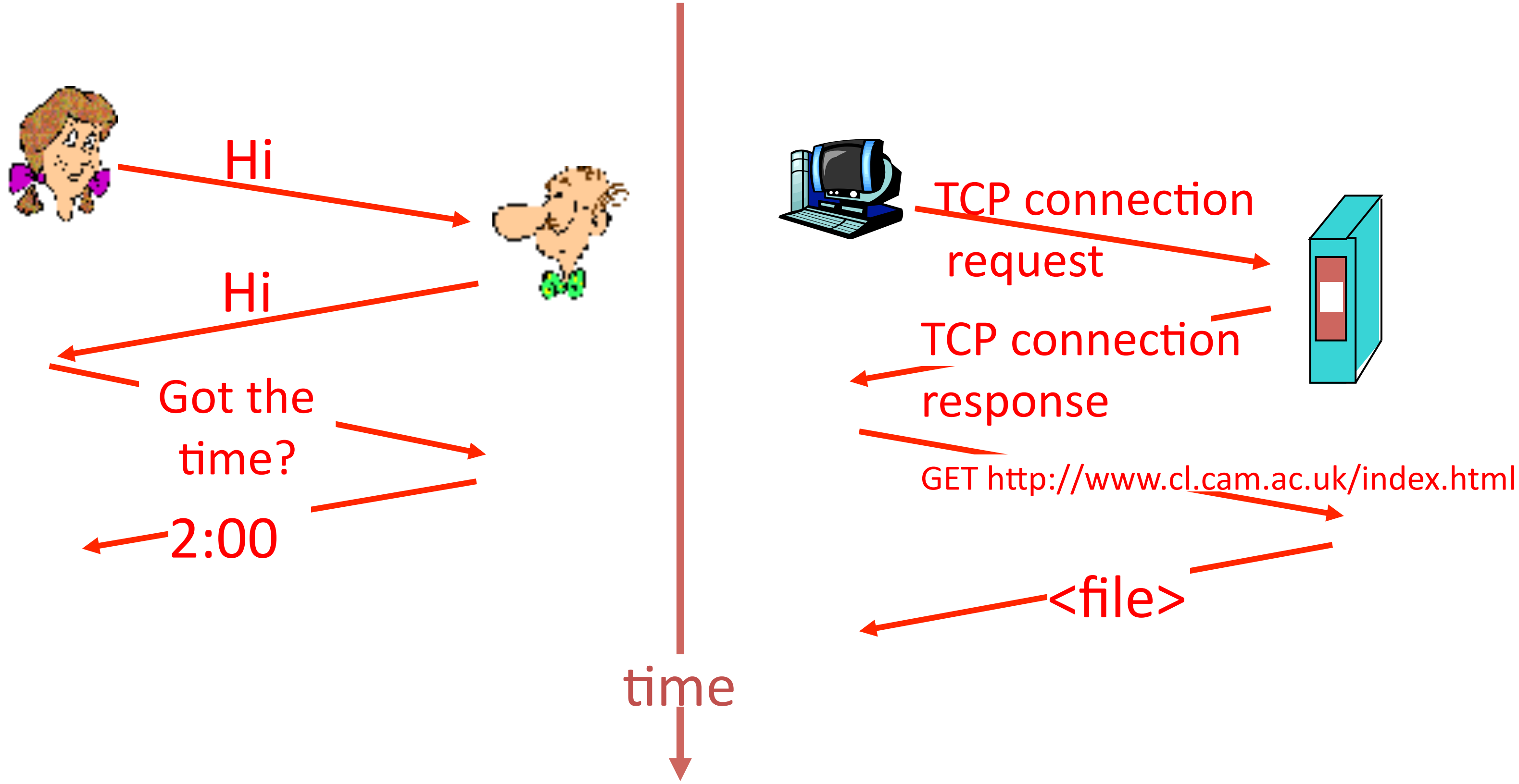
network protocols:

- 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:



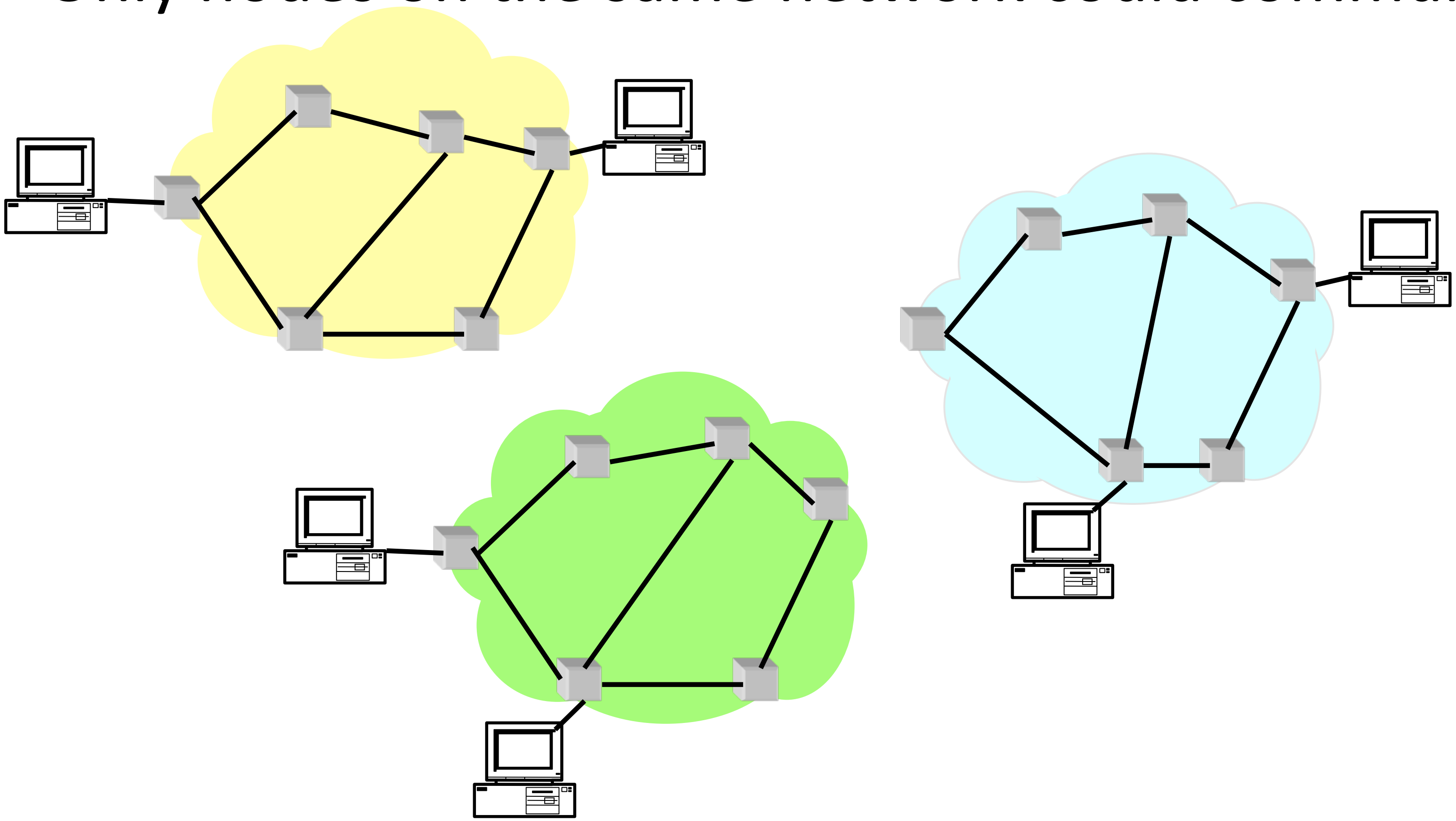
Q: Other human protocols?

Protocol Standardization

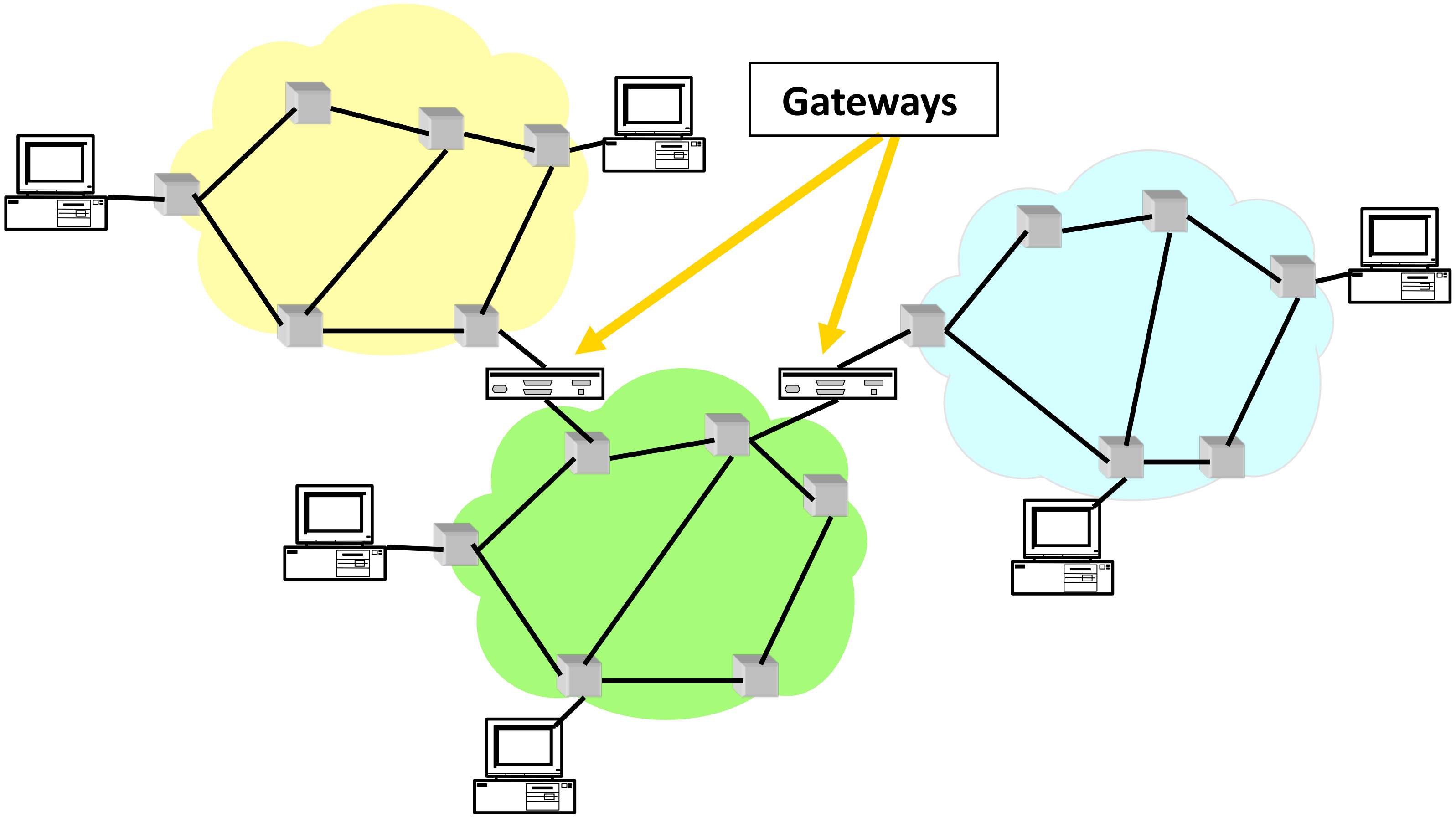
- All hosts must follow same protocol
  - Very small modifications can make a big difference
  - Or prevent it from working altogether
  - Cisco bug compatible!
- This is why we have standards
  - Can have multiple implementations of protocol
- Internet Engineering Task Force
  - Based on working groups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



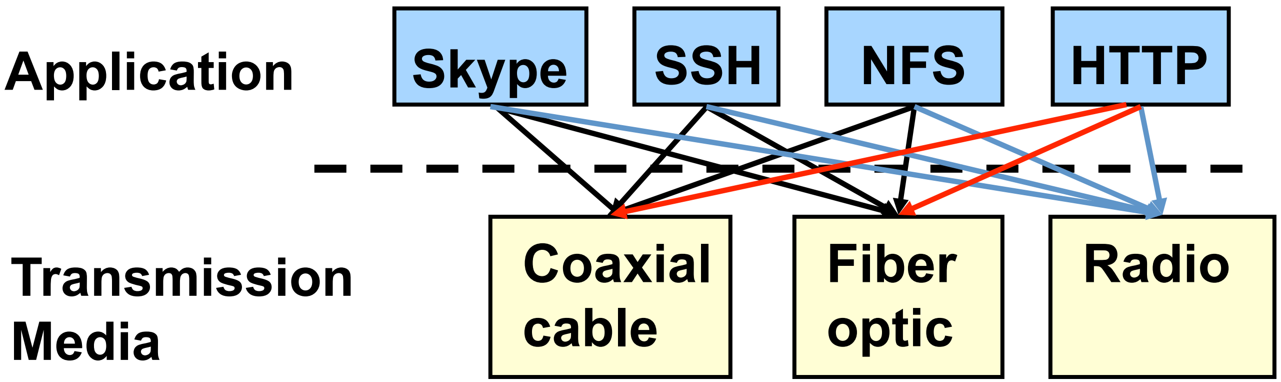
INTERNet Solution



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.

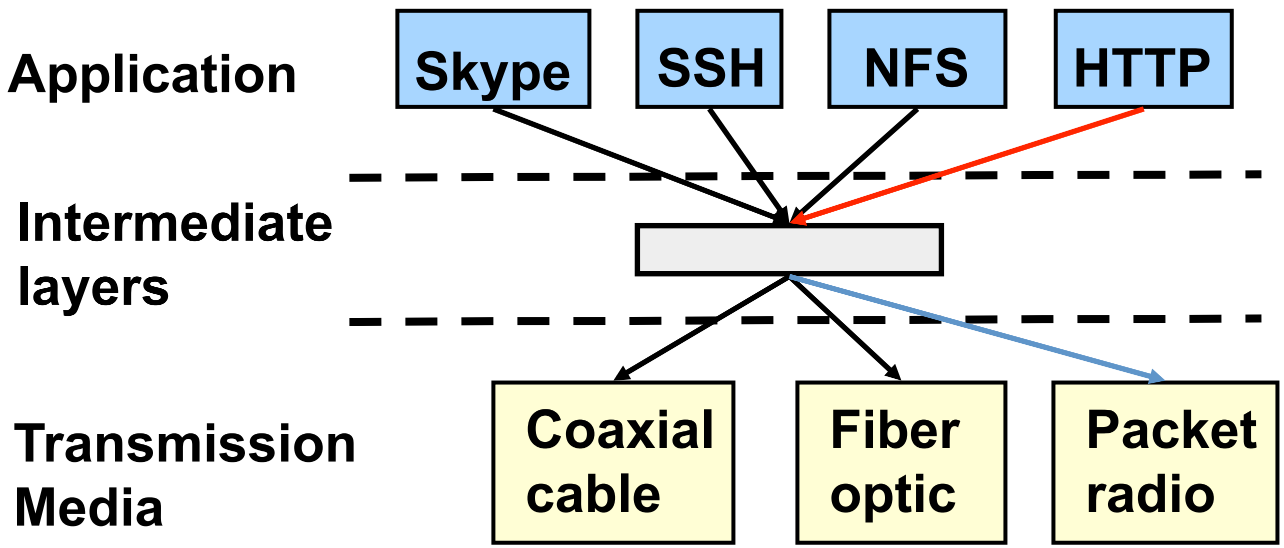
A Multitude of Apps Problem



- 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…

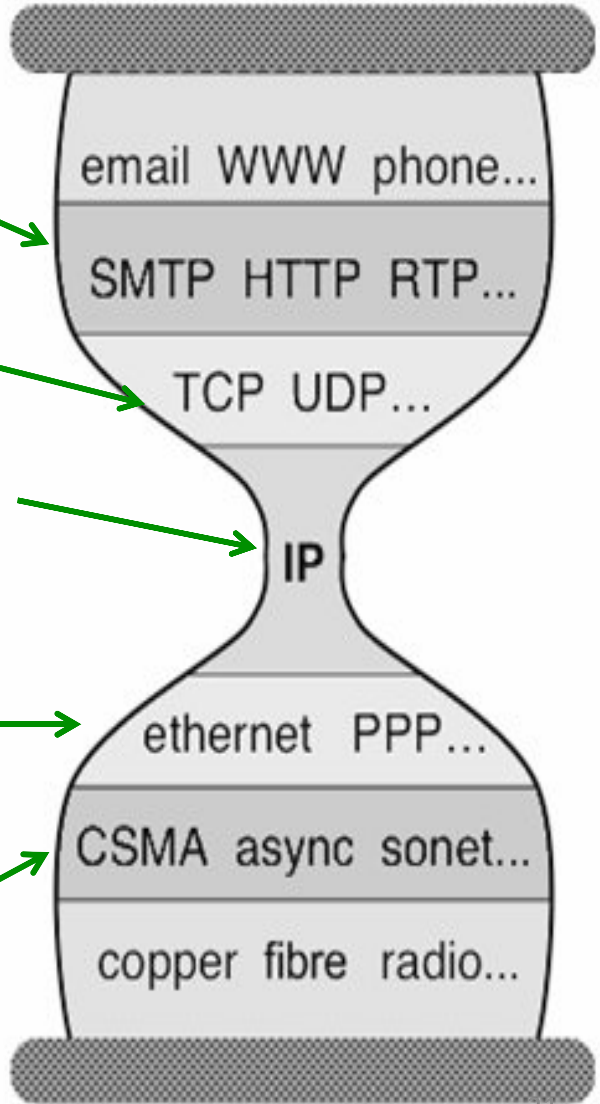
Best-effort global packet delivery

…built on…

Best-effort local packet delivery

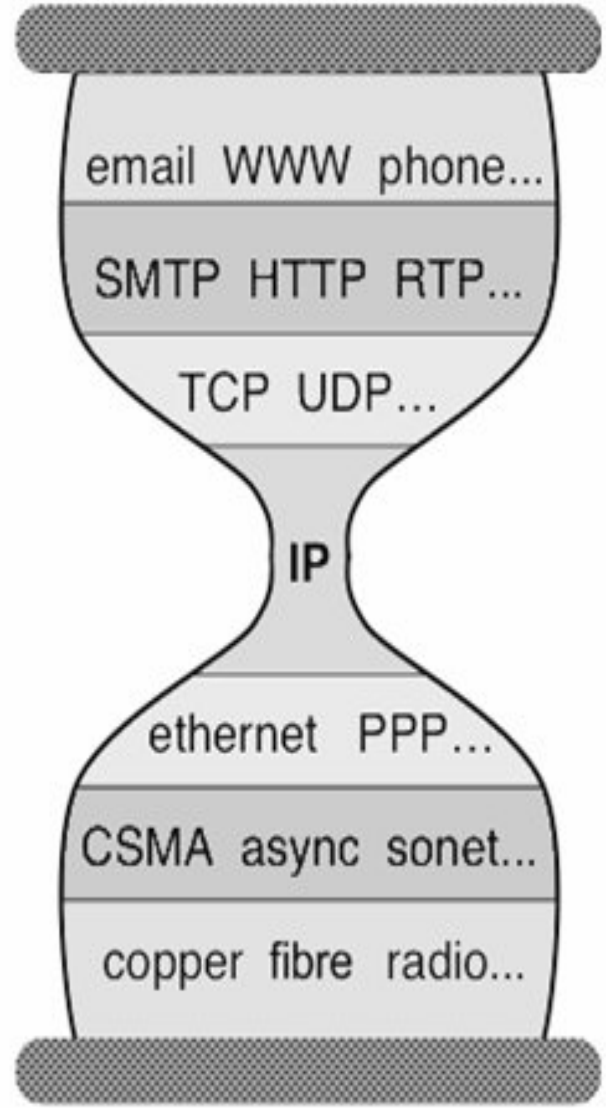
…built on…

Physical transfer of bits



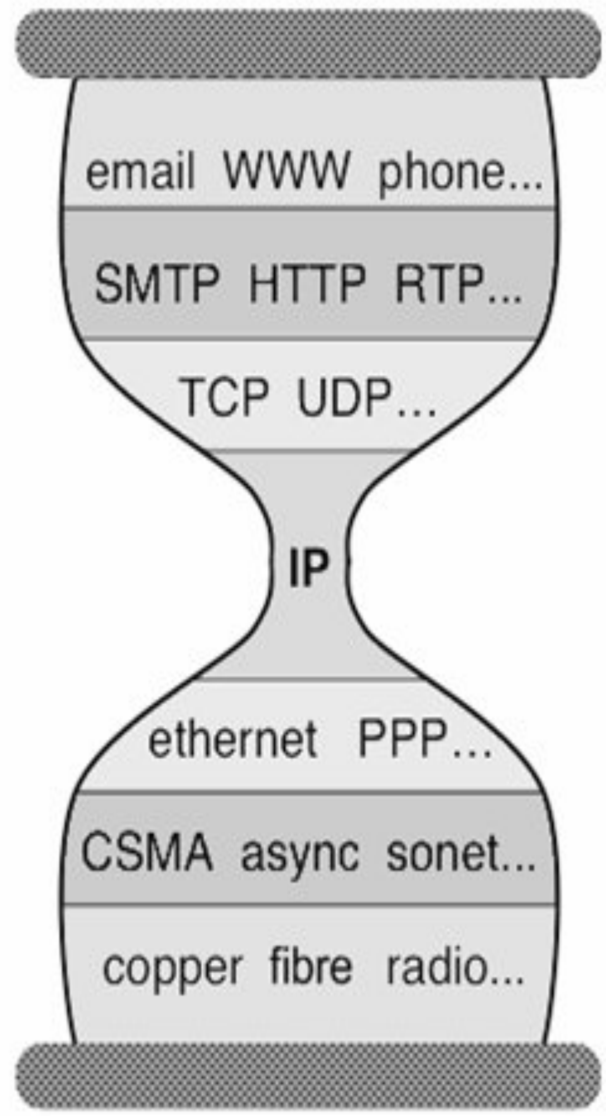
Three Observations

- 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!)

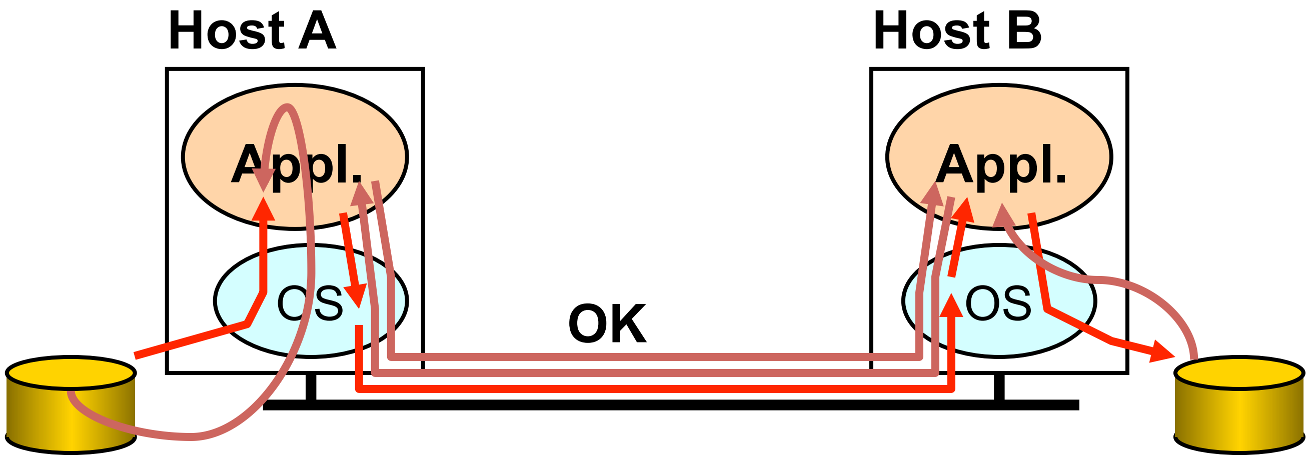
Basic Observation

- 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

Topic 3: The Data Link Layer

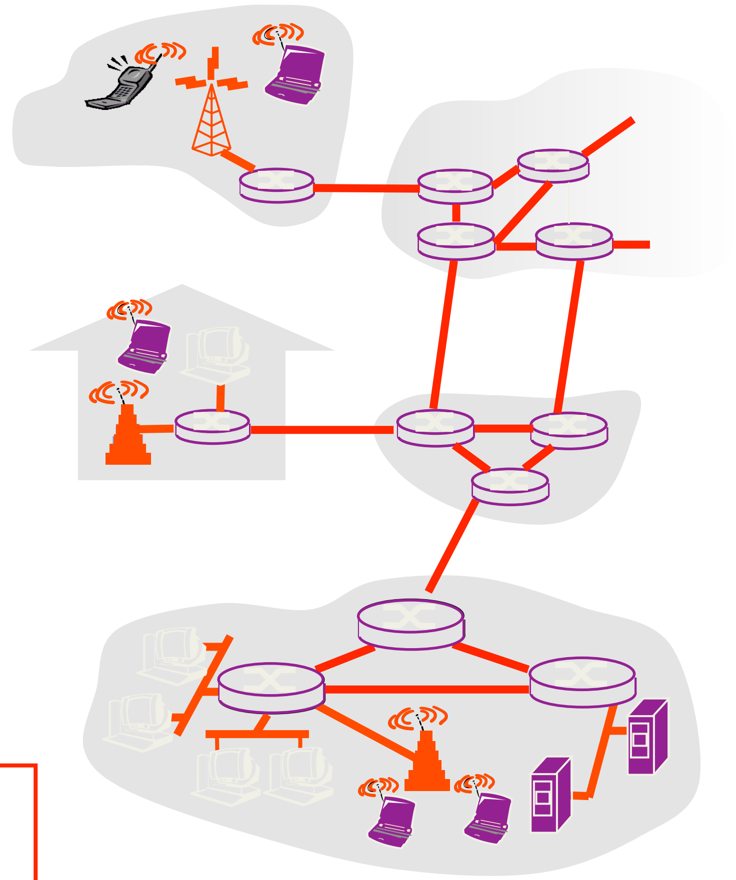
Our goals:

- understand principles behind data link layer services:
  (these are methods & mechanisms in your networking toolbox)
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control:
- instantiation and implementation of various link layer technologies
  - Wired Ethernet (aka 802.3)
  - Wireless Ethernet (aka 802.11 WiFi)
- Algorithms
  - Binary Exponential Backoff
  - Spanning Tree

Link Layer: Introduction

Some terminology:

- hosts and routers are **nodes**
- communication channels that connect adjacent nodes along communication path are **links**
  - wired links
  - wireless links
  - LANs
- layer-2 packet is a **frame**, encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to adjacent node over a link

Link Layer (Channel) Services

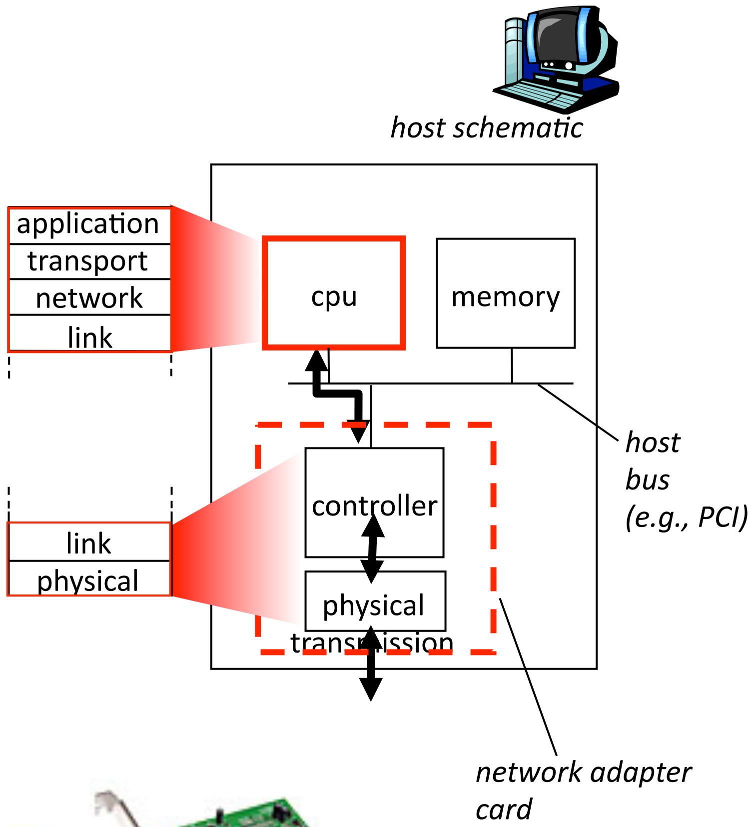
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we see some of this again in the Transport Topic
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?

Link Layer (Channel) Services - 2

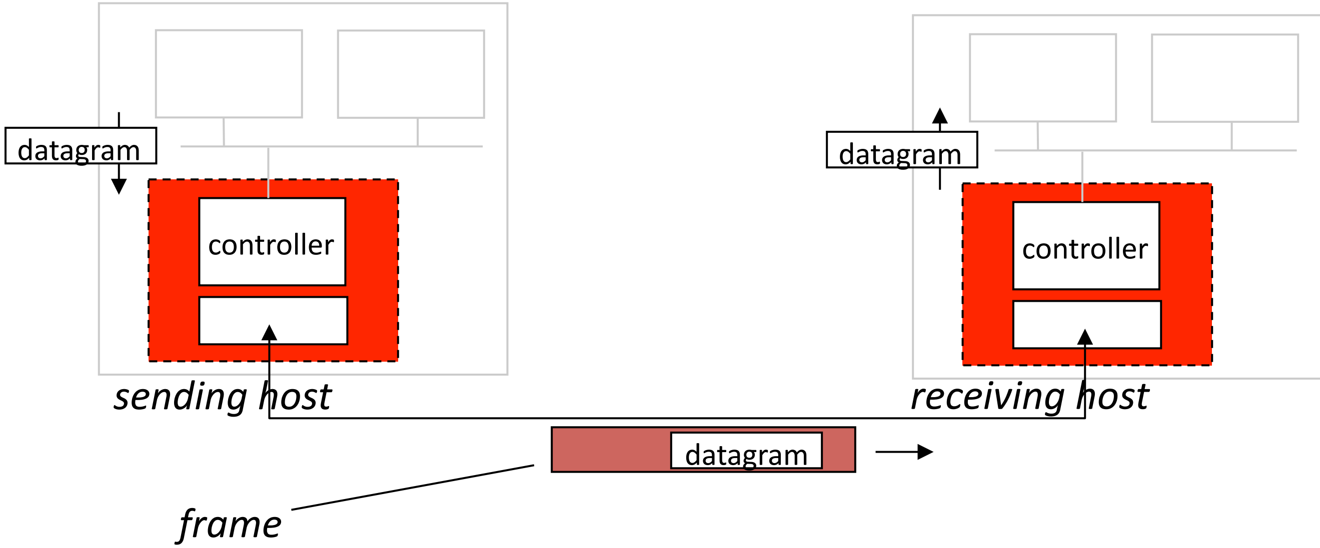
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- error correction:
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC)
  - Ethernet card, PCMCI card, 802.11 card
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



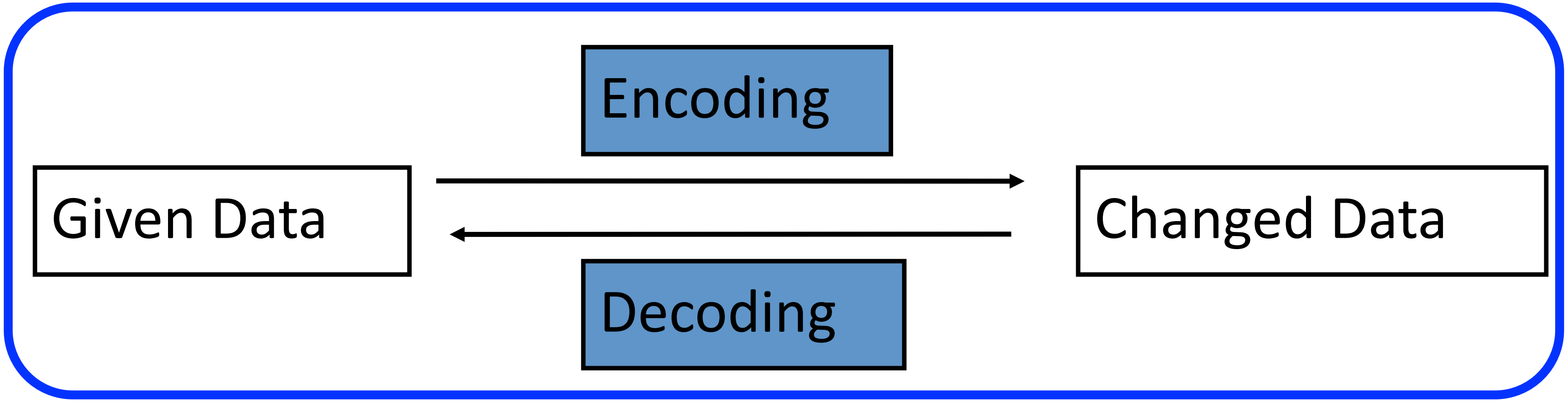
Adaptors Communicating

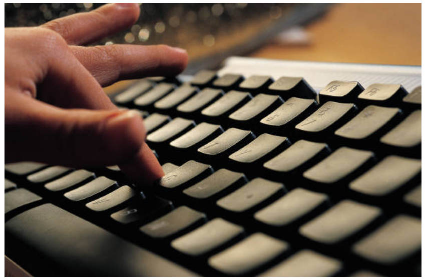


- sending side:
  - encapsulates datagram in frame
  - encodes data for the physical layer
  - adds error checking bits, provide reliability, flow control, etc.
- receiving side
  - decodes data from the physical layer
  - looks for errors, provide reliability, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

Coding – a channel function

Change the representation of data.

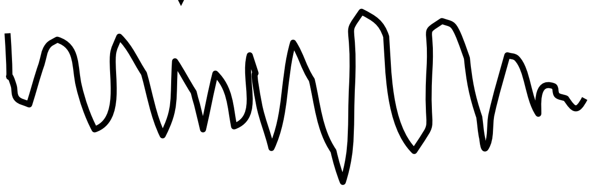




MyPasswd

AA$$$$ff

AA$$$$ffff



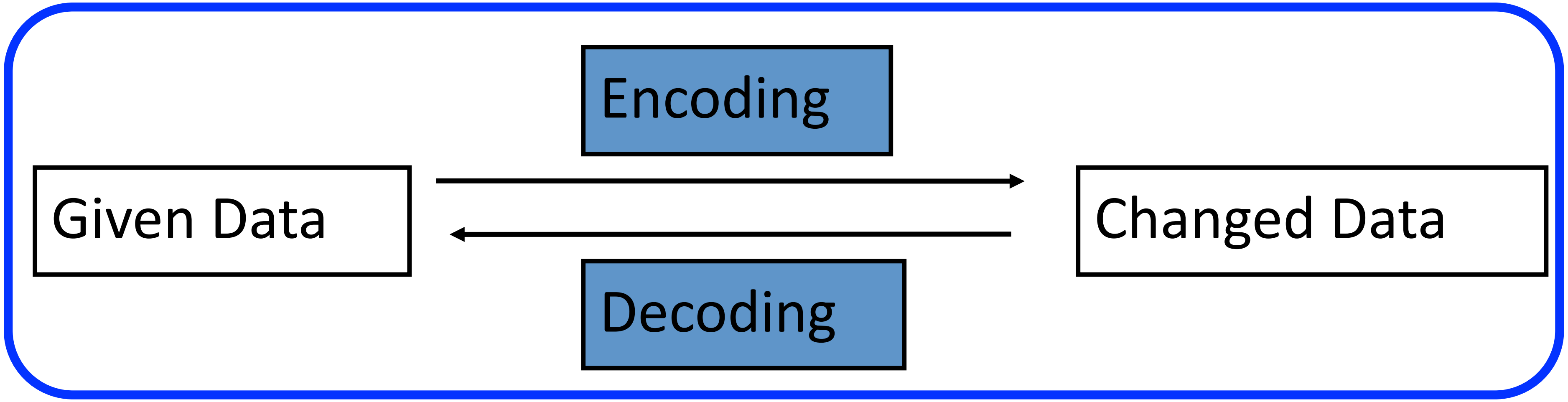
MyPasswd

AA$$$$ff

AA$$$$ffff

Coding

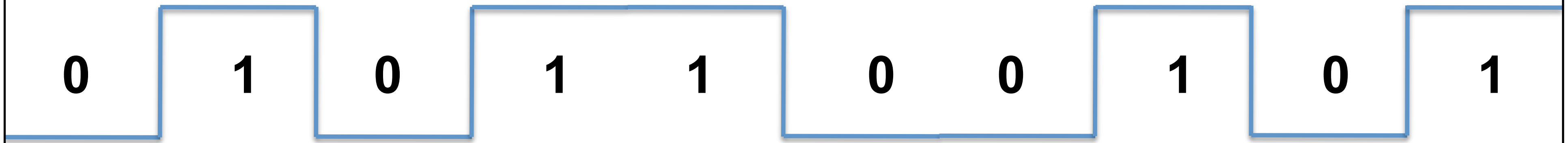
Change the representation of data.



1. Encryption: MyPasswd <-> AA$$$$ff
2. Error Detection: AA$$$$ff <-> AA$$$$ffff
3. Compression: AA$$$$ffff <-> A2$4f4
4. Analog: A2$4f4 <-> 

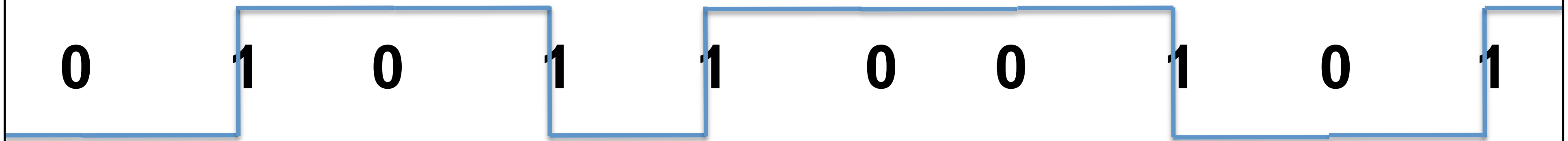
Line Coding Examples
where Baud=bit-rate

Non-Return-to-Zero (NRZ)

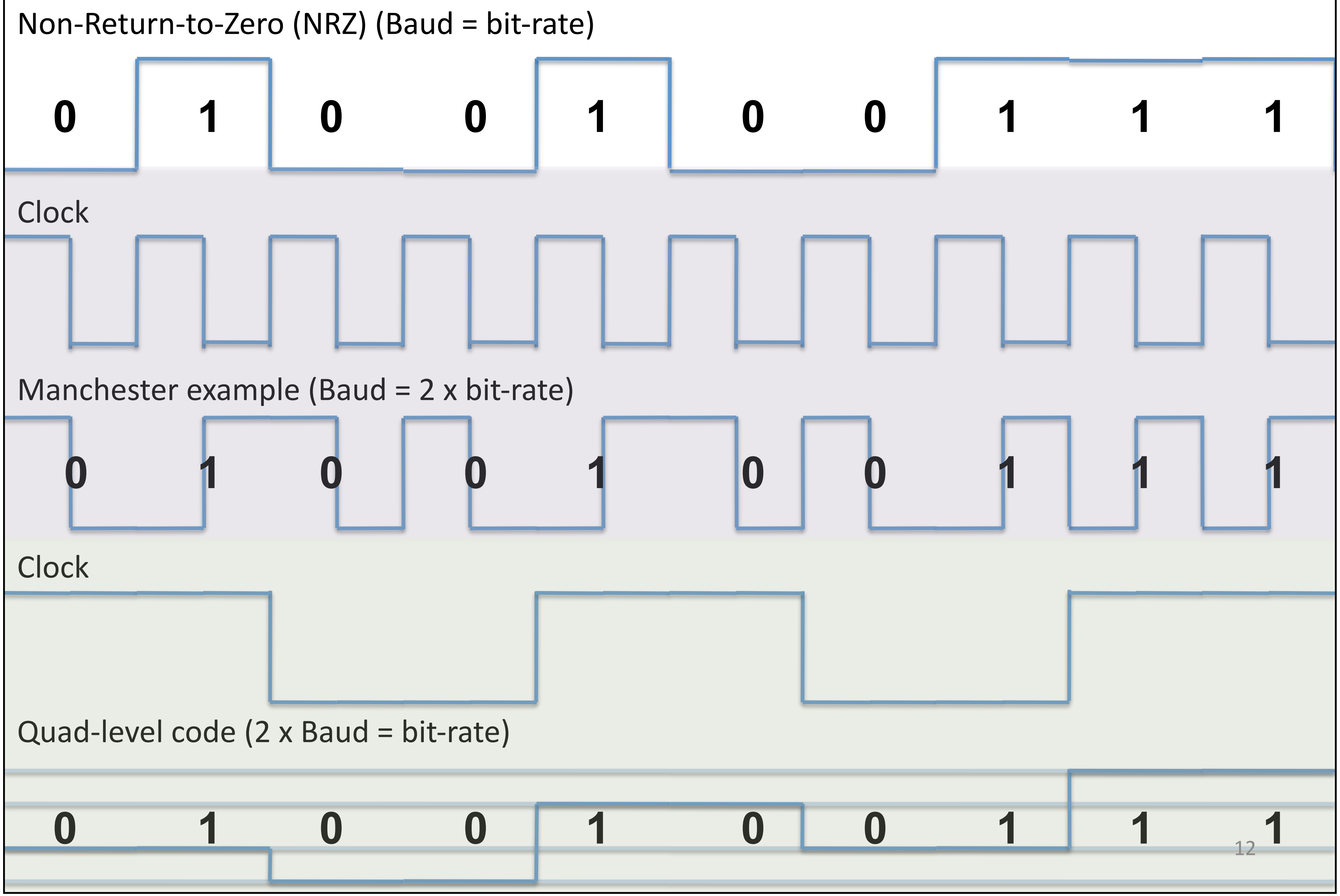


Non-Return-to-Zero-Mark (NRZM) 1 = transition 0 = no transition

Non-Return-to-Zero Inverted (NRZI) (note transitions on the 1)

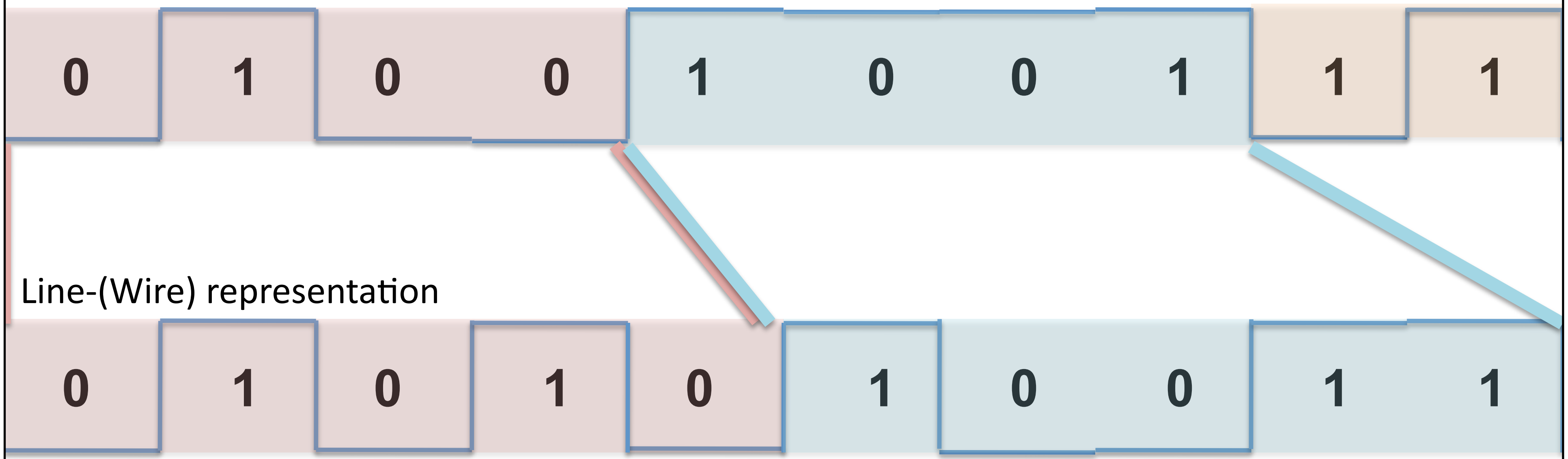


Line Coding Examples



Line Coding – Block Code example

Data to send



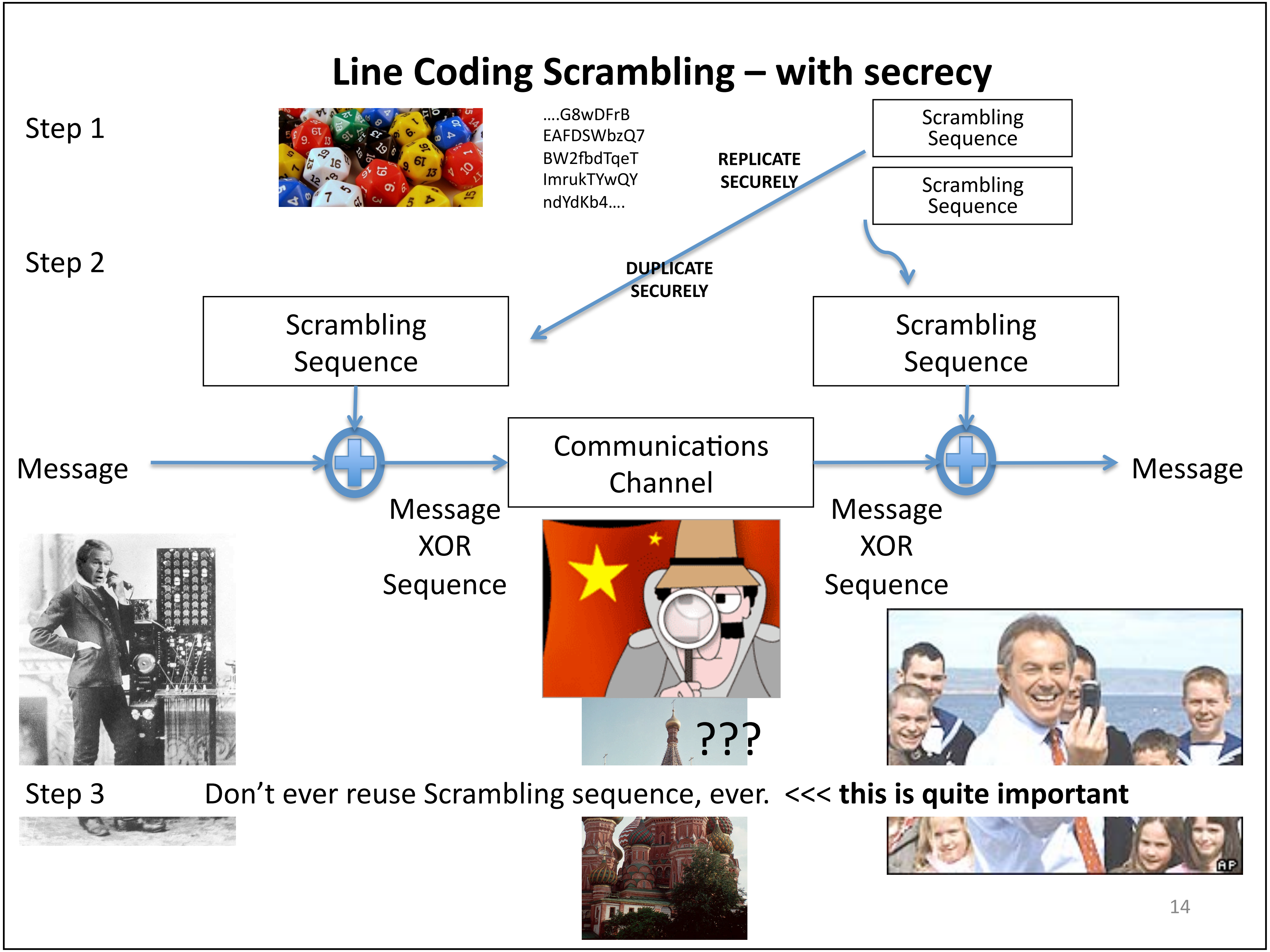
Name	4b	5b	Description	
0	0000	11110	hex data 0	
1	0001	01001	hex data 1	
2	0010	10100	hex data 2	
3	0011	10101	hex data 3	
4	0100	01010	hex data 4	
5	0101	01011	hex data 5	
6	0110	01110	hex data 6	
7	0111	01111	hex data 7	
8		1000	10010	hex data 8
9		1001	10011	hex data 9
A	1010	10110	hex data A	
B	1011	10111	hex data B	
C	1100	11010	hex data C	
D	1101	11011	hex data D	
E	1110	11100	hex data E	
F	1111	11101	hex data F	

Name	4b	5b	Description
Q	-NONE-	00000	Quiet
I	-NONE-	11111	Idle
J	-NONE-	11000	SSD #1
K	-NONE-	10001	SSD #2
T	-NONE-	01101	ESD #1
R	-NONE-	00111	ESD #2
H	-NONE-	00100	Halt

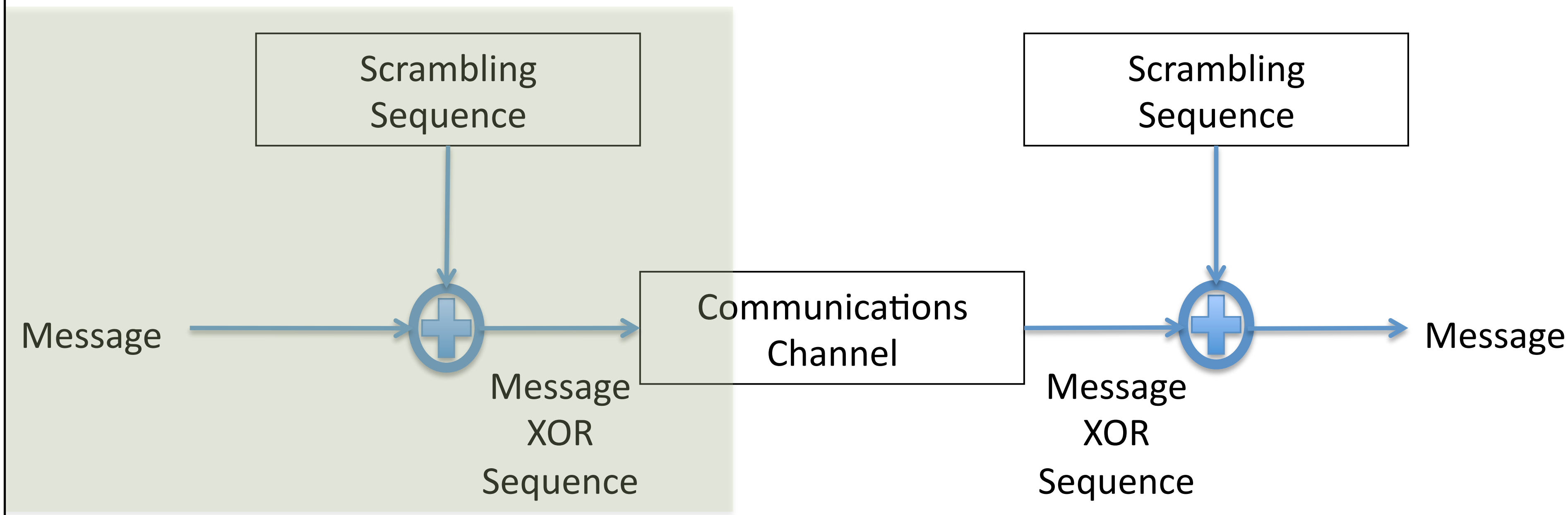
Block coding transfers data with a fixed overhead: 20% less information per Baud in the case of 4B/5B

So to send data at 100Mbps; the line rate (the Baud rate) must be 125Mbps.

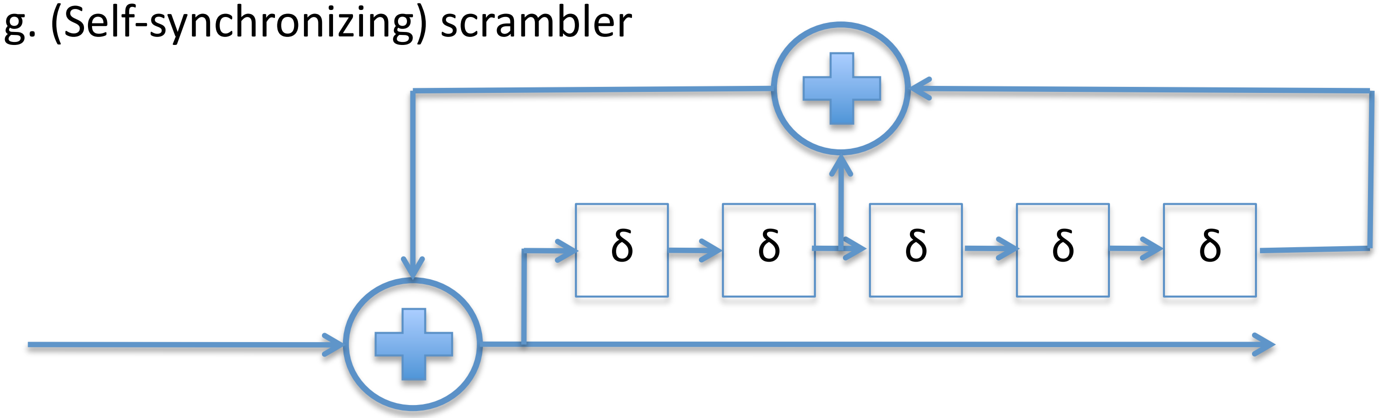
1Gbps uses an 8b/10b codec; encoding entire bytes at a time but with 25% overhead


Line Coding Scrambling – with secrecy
Step 1
....G8wDFrB
EAFDSWbzQ7
BW2fbdTqeT
ImrukTYwQY
ndYdKb4....
REPLICATE SECURELY
Scrambling Sequence
Scrambling Sequence
Step 2
DUPLICATE SECURELY
Scrambling Sequence
Scrambling Sequence
Message
Communications Channel
Message
Message XOR Sequence
Message XOR Sequence
???
Step 3
Don’t ever reuse Scrambling sequence, ever. <<< this is quite important
AP
14

Line Coding Scrambling– no secrecy



e.g. (Self-synchronizing) scrambler



Line Coding Examples (Hybrid)

…100111101101010001000101100111010001010010110101001001110101110100…

…10011110110101000101000101100111010001010010110101001001110101110100…

Inserted bits marking "start of frame/block/sequence"

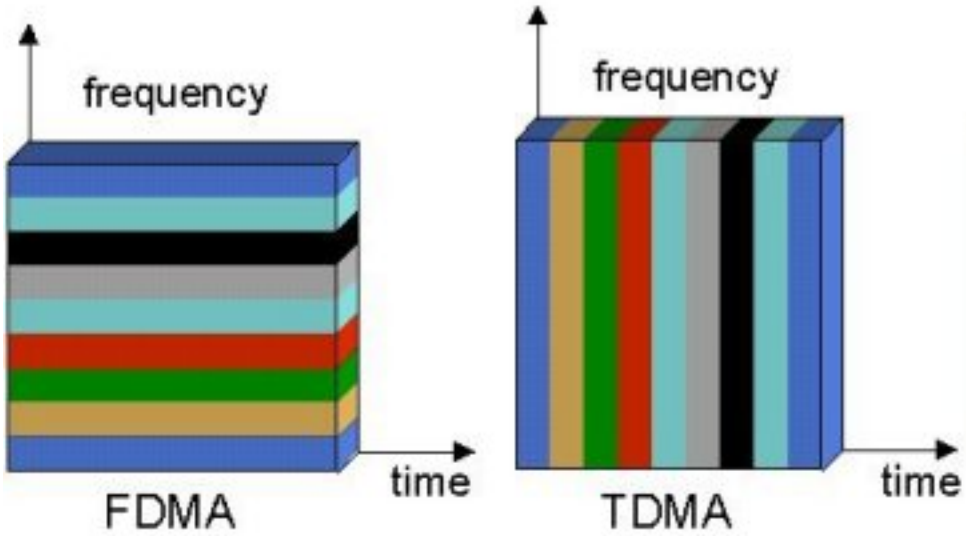
Scramble / Transmit / Unscramble

…0100010110011101000101001011010100100111010111010010010111011101111000…

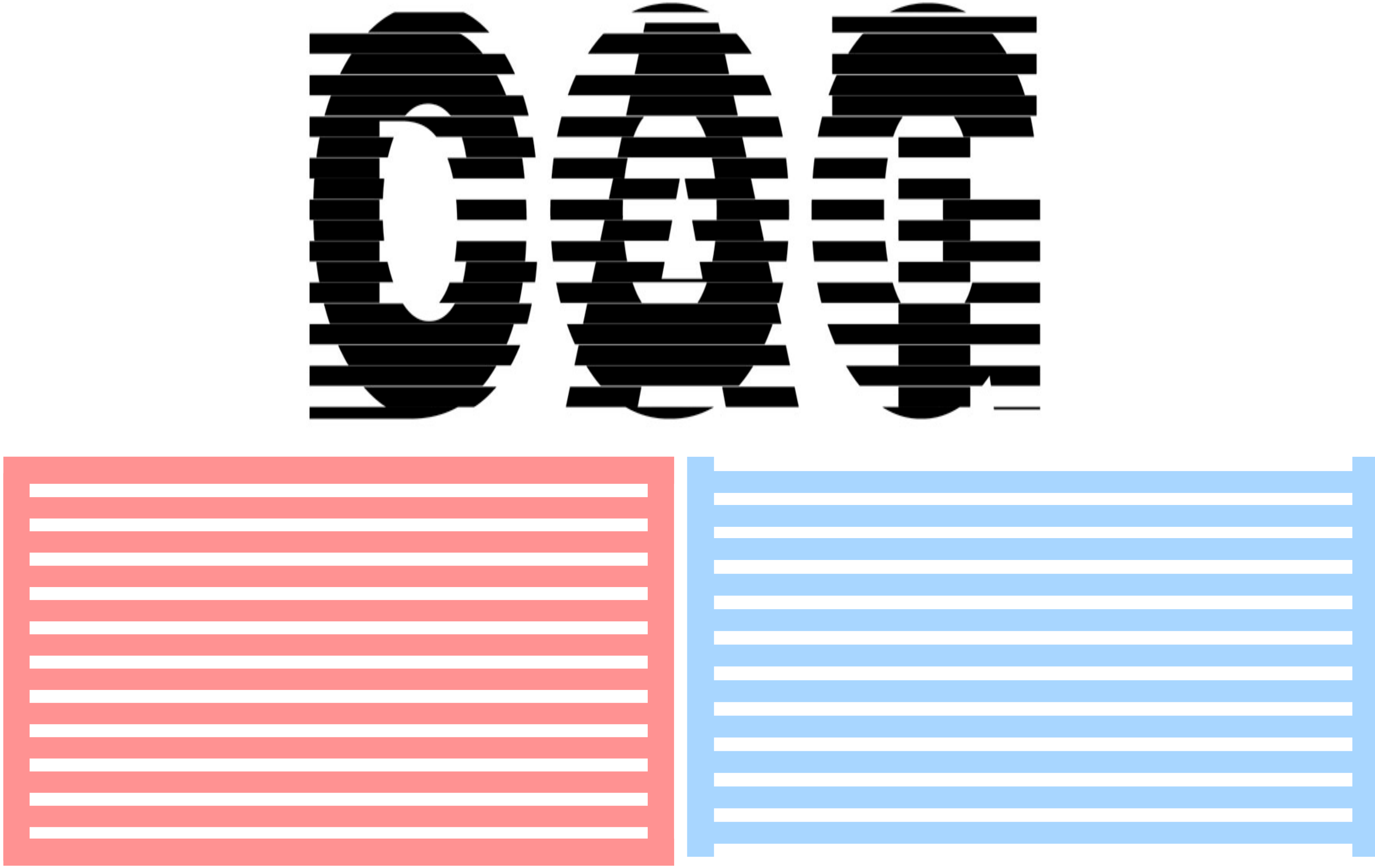
Identify (and remove) "start of frame/block/sequence"
This gives you the Byte-delineations for *free*

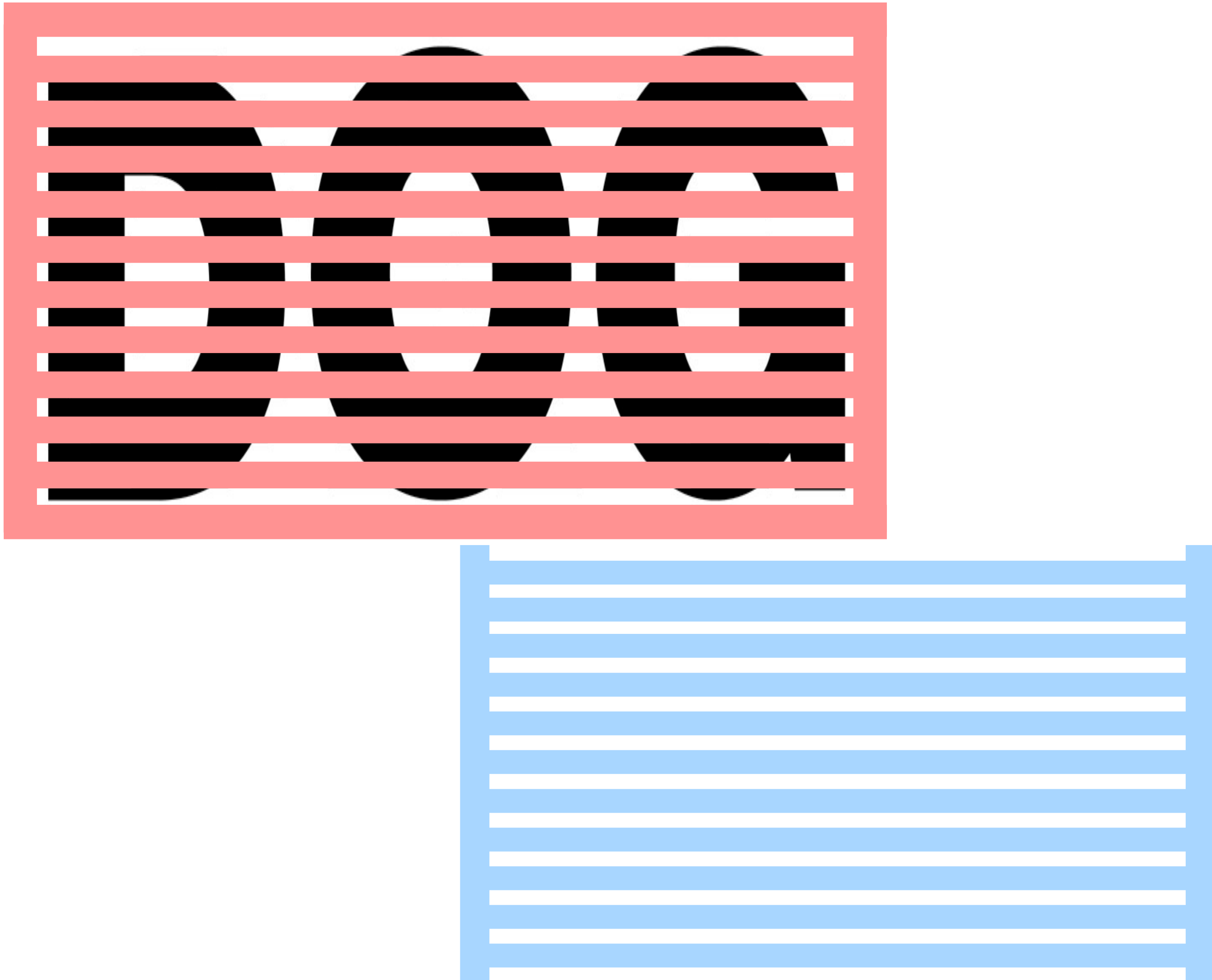
64b/66b combines a scrambler and a framer. The start of frame is a pair of bits 01 or 10: 01 means "this frame is data" 10 means "this frame contains data and control" – control could be configuration information, length of encoded data or simply "this line is idle" (no data at all)

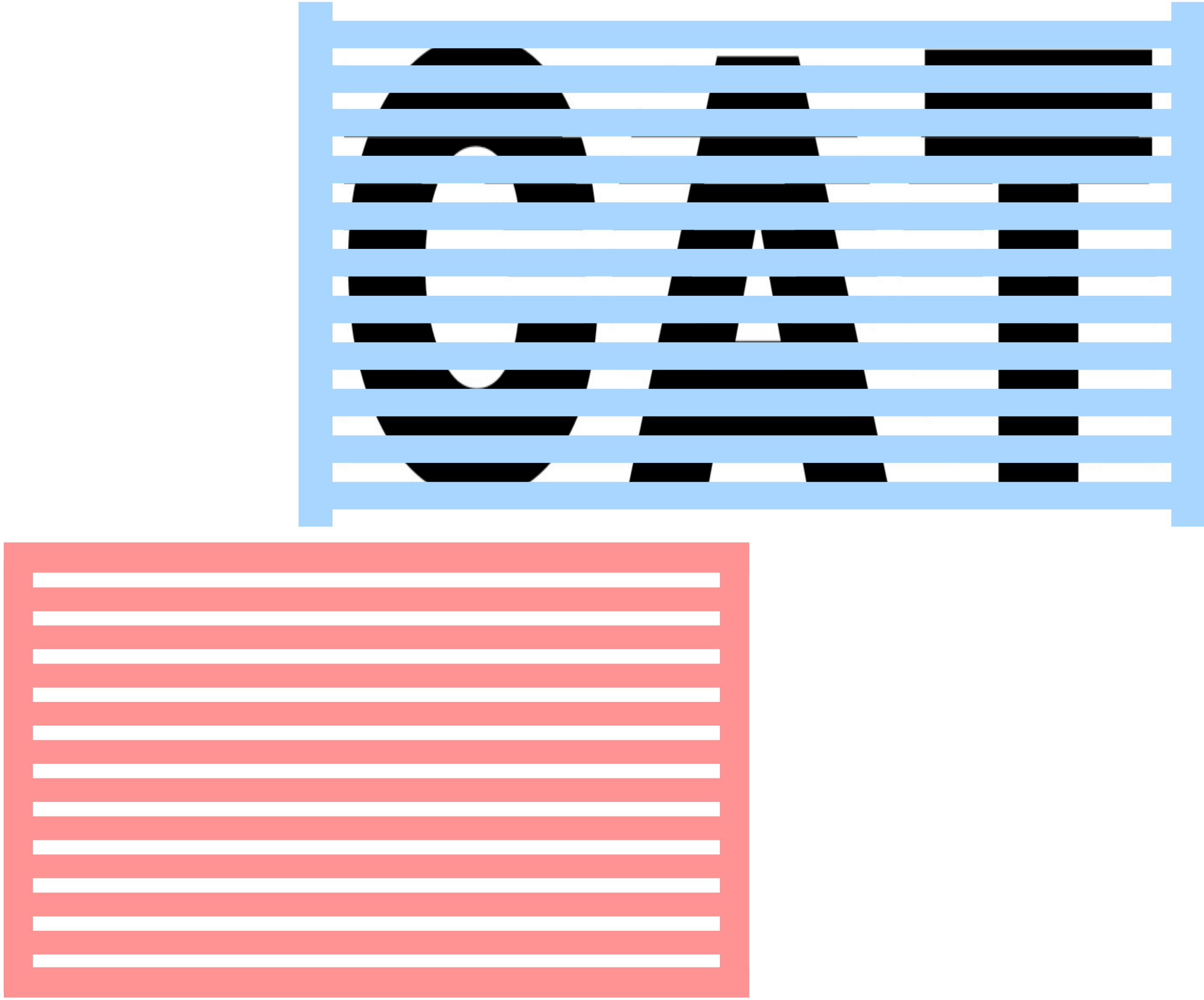
Multiple Access Mechanisms



Each dimension is orthogonal (so may be trivially combined)
There are other dimensions too; can you think of them?



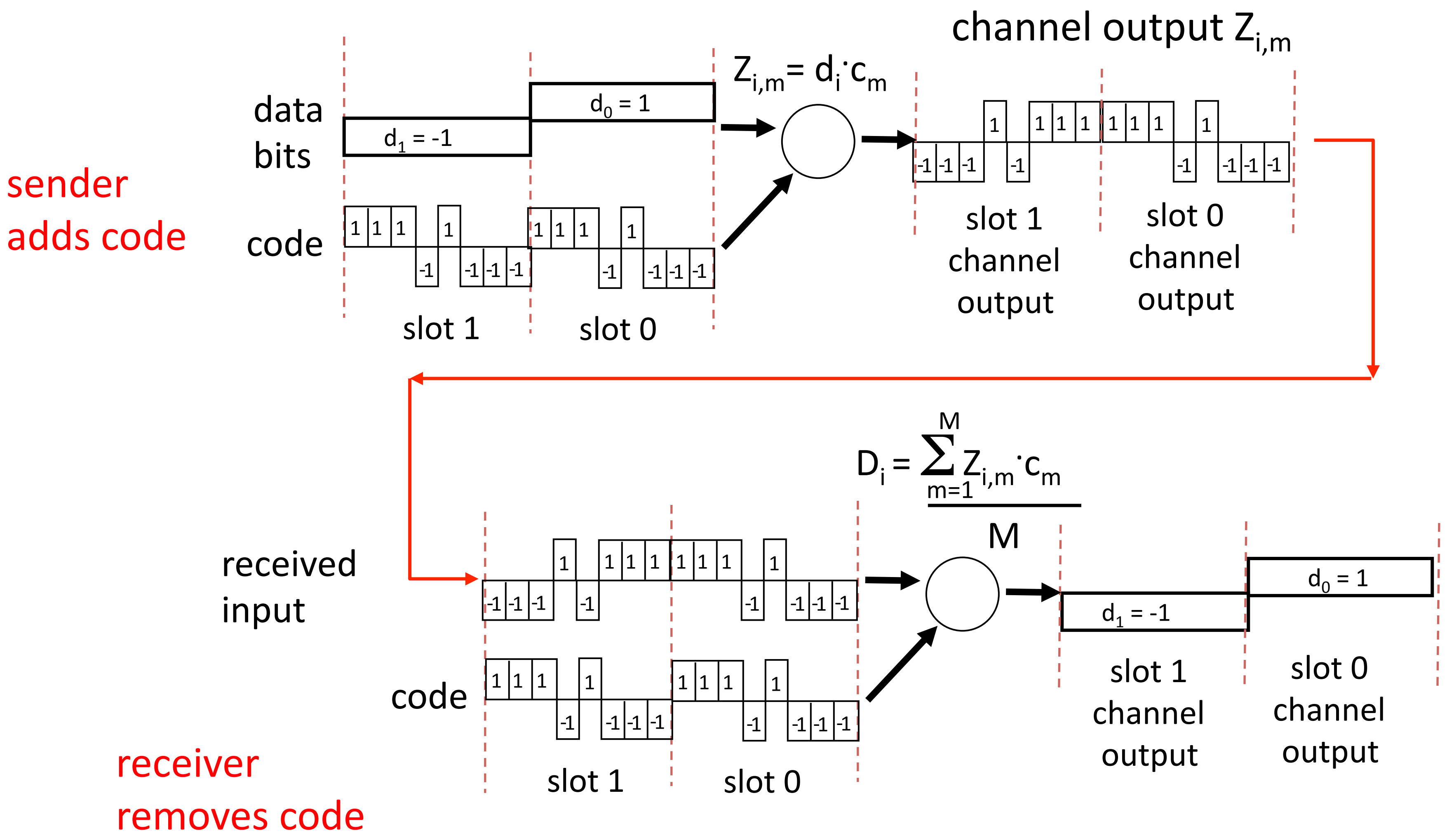

DOC


CAT

Code Division Multiple Access (CDMA) (not to be confused with CSMA!)

- used in several wireless broadcast channels (cellular, satellite, etc) standards
- unique "code" assigned to each user; i.e., code set partitioning
- all users share same frequency, but each user has own "chipping" sequence (i.e., code) to encode data
- *encoded signal* = (original data) XOR (chipping sequence)
- *decoding:* inner-product of encoded signal and chipping sequence
- allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal")

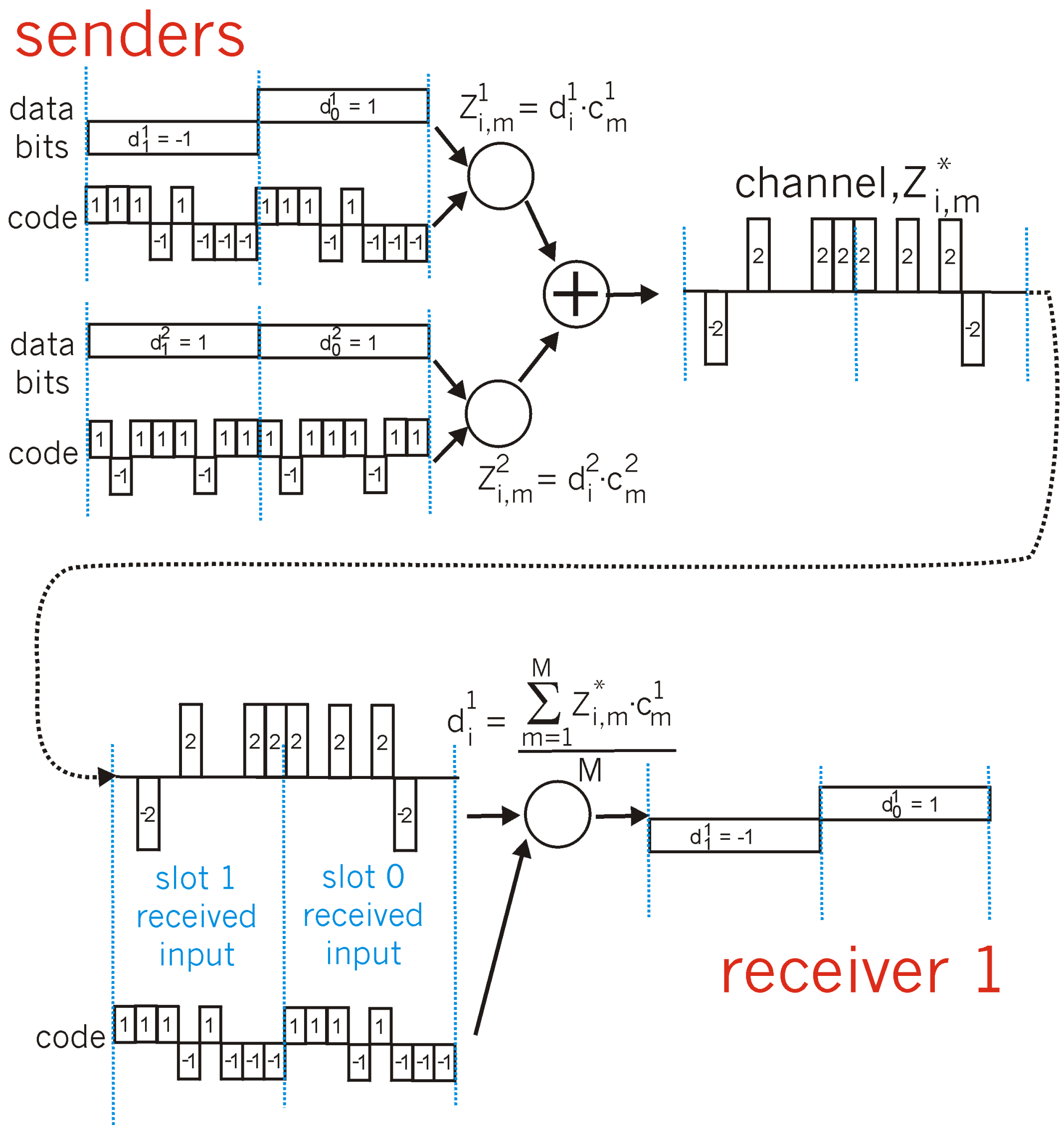
CDMA Encode/Decode



CDMA: two-sender interference

Each sender adds a ***unique*** code

sender removes its ***unique*** code



Coding Examples summary

- Common Wired coding
  - Block codecs: table-lookups
    - fixed overhead, inline control signals
  - Scramblers: shift registers
    - overhead free

Like earlier coding schemes and error correction/detection; you can combine these

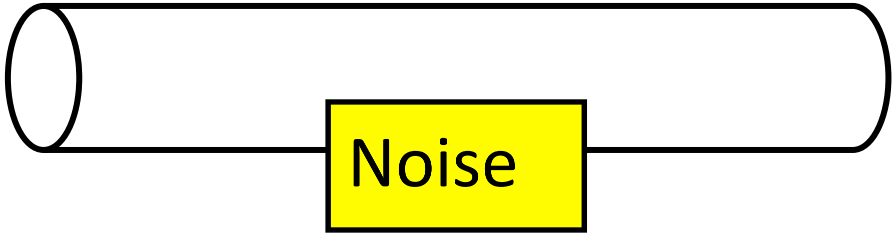
- e.g, 10Gb/s Ethernet may use a hybrid

CDMA (Code Division Multiple Access)

- coping intelligently with competing sources
- Mobile phones

Error Detection and Correction

How to use coding to deal with errors in data communication?



0000	0000

0001	0000

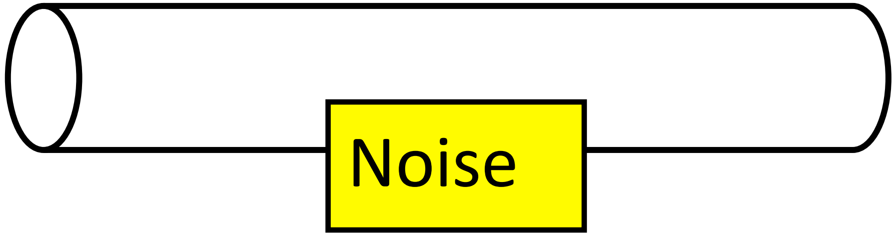
Basic Idea :

1. Add additional information to a message.
2. Detect an error and re-send a message.

   Or, fix an error in the received message.

Error Detection and Correction

How to use coding to deal with errors in data communication?



0000	0000

0000	0000

Basic Idea :

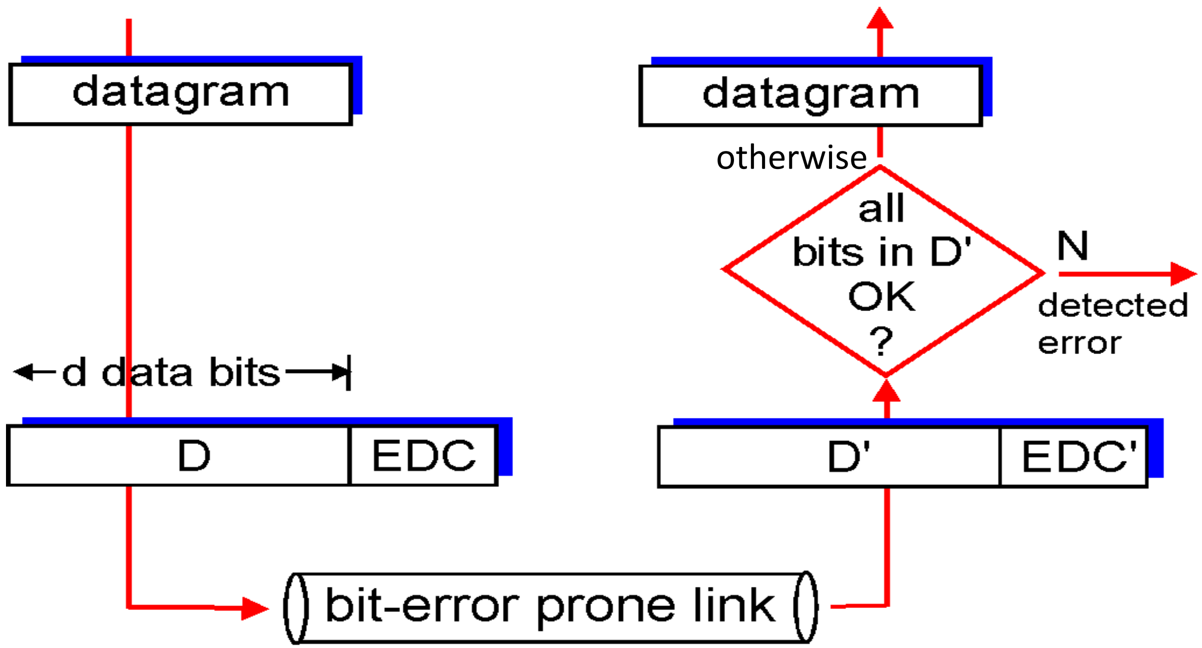
1. Add additional information to a message.
2. Detect an error and re-send a message.

   Or, fix an error in the received message.

Error Detection

EDC= Error Detection and Correction bits (redundancy = overhead)
D = Data protected by error checking, may include header fields

- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Error Detection Code

Sender:

Y = generateCheckBit(X);

send(XY);

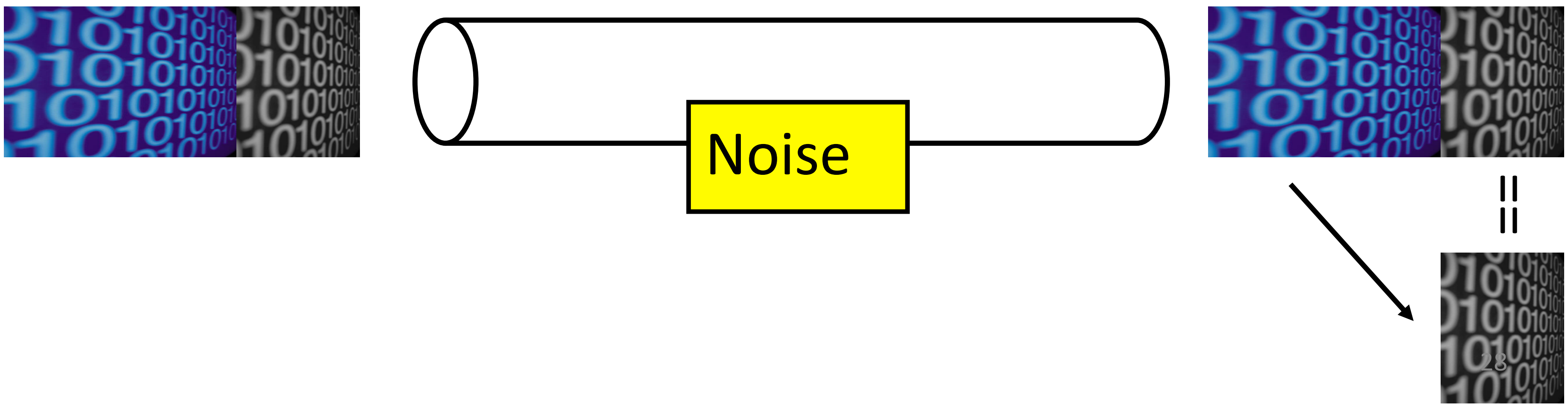
Receiver:

receive(X1Y1);

Y2=generateCheckBit(X1);

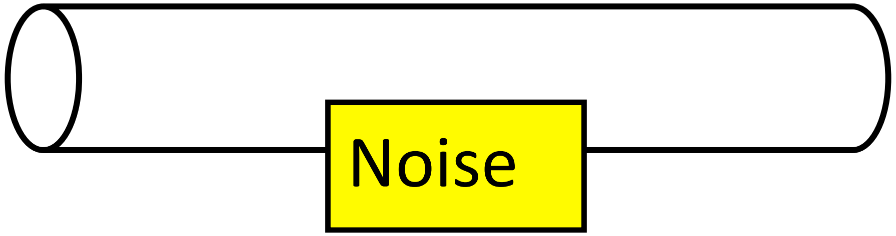
if (Y1 != Y2) ERROR;

else NOERROR



Error Detection Code: Parity

Add one bit, such that the number of 1's is even.



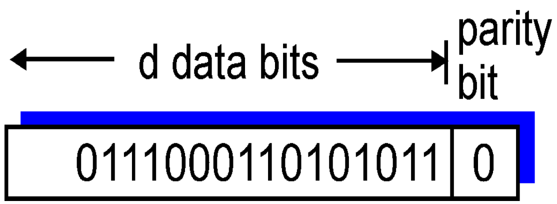
Sent			Received	
0000	0	✖	0001	0
0001	1	✔	0001	1
1001	0	✔	1111	0

Problem: This simple parity cannot detect two-bit errors.

Parity Checking

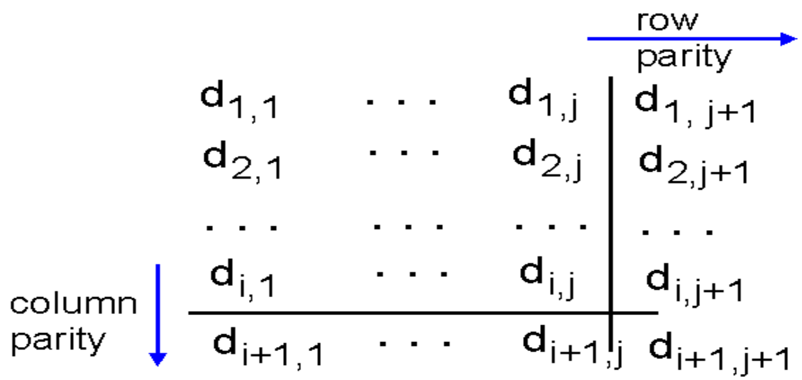
Single Bit Parity:

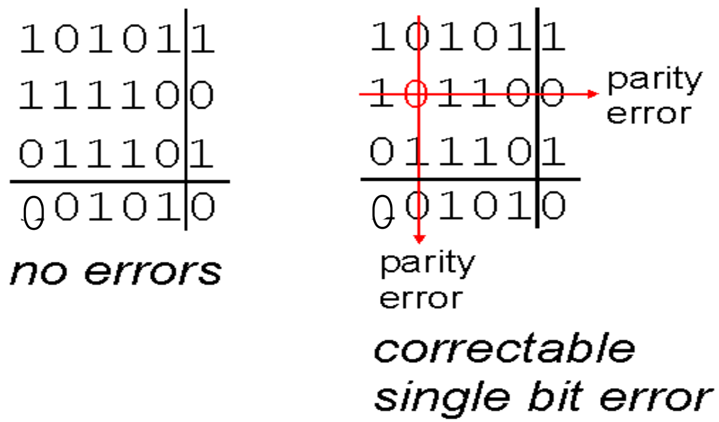
Detect single bit errors



Two Dimensional Bit Parity:

Detect *and correct* single bit errors





Internet checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

Sender:

- treat segment contents as sequence of 1bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

Error Detection Code: CRC

- CRC means "Cyclic Redundancy Check".
- More powerful than parity.
  - It can detect various kinds of errors, including 2-bit errors.
- More complex: multiplication, binary division.
- Parameterized by n-bit divisor P.
  - Example: 3-bit divisor 101.
  - Choosing good P is crucial.

CRC with 3-bit Divisor 101

1111

1001

CRC	Parity
00	0
11	0

111

100

same check bits from Parity,
but different ones from CRC

Multiplication by 2^3

$D2 = D * 2^3$

→

Binary Division by 101

CheckBit = (D2) rem (101)

Add three 0's at the end

Kurose p478 §5.2.3
Peterson p97 §2.4.3

The divisor (G) – Secret sauce of CRC

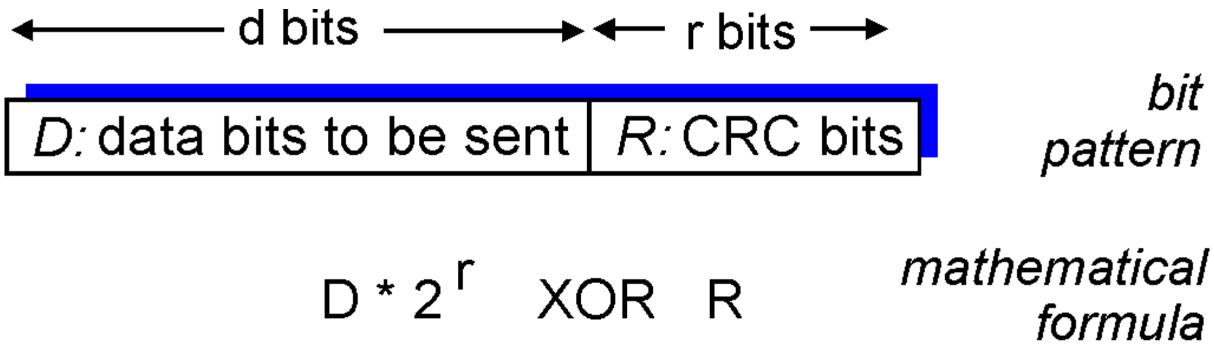
- If the divisor were 100, instead of 101, data 1111 and 1001 would give the same check bit 00.
- Mathematical analysis about the divisor:
  - Last bit should be 1.
  - Should contain at least two 1's.
  - Should be divisible by 11.
- ATM, HDLC, Ethernet each use a CRC with well-chosen fixed divisors

Divisor analysis keeps mathematicians in jobs

(a branch of *pure* math: combinatorial mathematics)

Checksumming: Cyclic Redundancy Check recap

- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



CRC Another Example – this time with long division

Want:

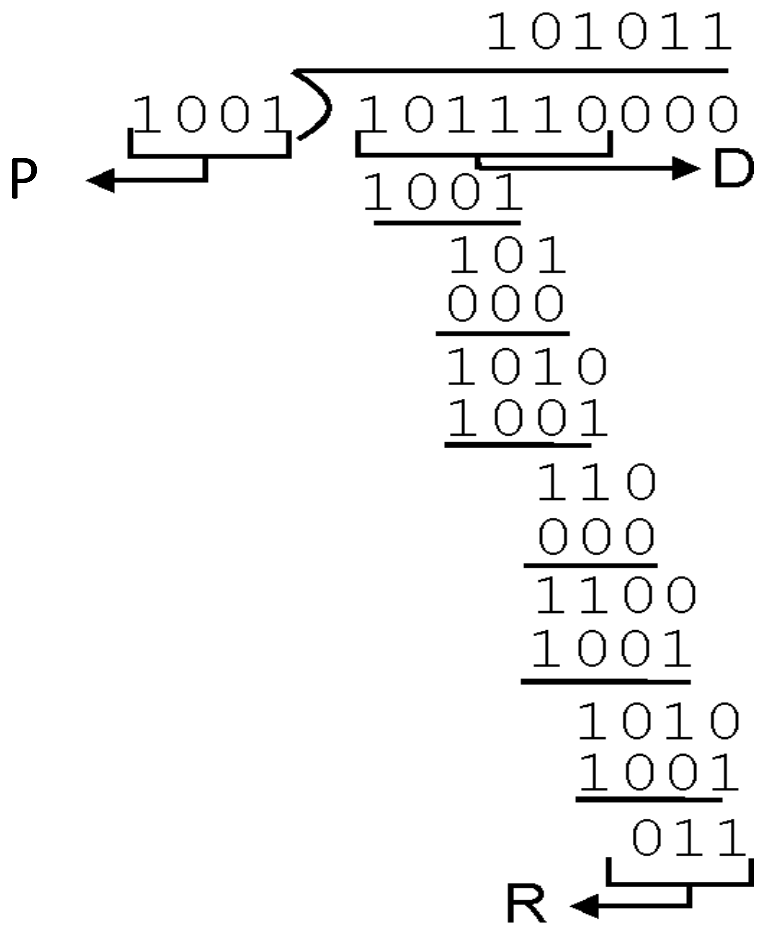
$D \cdot 2^r$ XOR R = nP

equivalently:

$D \cdot 2^r$ = nP XOR R

equivalently:

if we divide $D \cdot 2^r$ by P, want remainder R



$$R = \text{remainder}\left[\frac{D \cdot 2^r}{P}\right]$$

FYI: *in K&R P is called the Generator: G*

Error Detection Code becomes….

Sender:

Y = generateCheckBit(X);

send(XY);

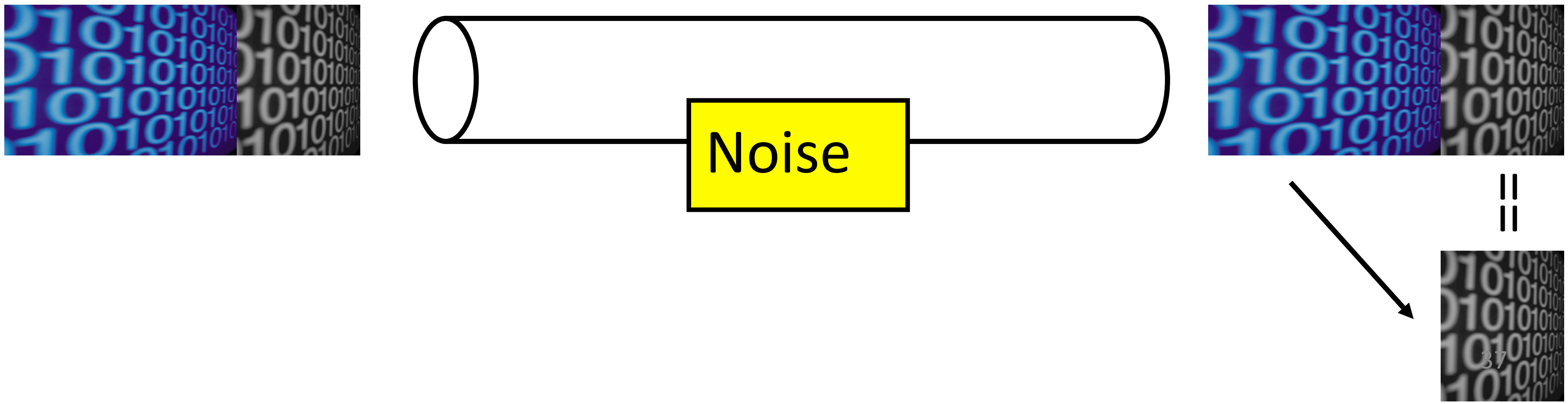
Receiver:

receive(X1Y1);

Y2=generateCheckBit(X1);

if (Y1 != Y2) ERROR;

else NOERROR



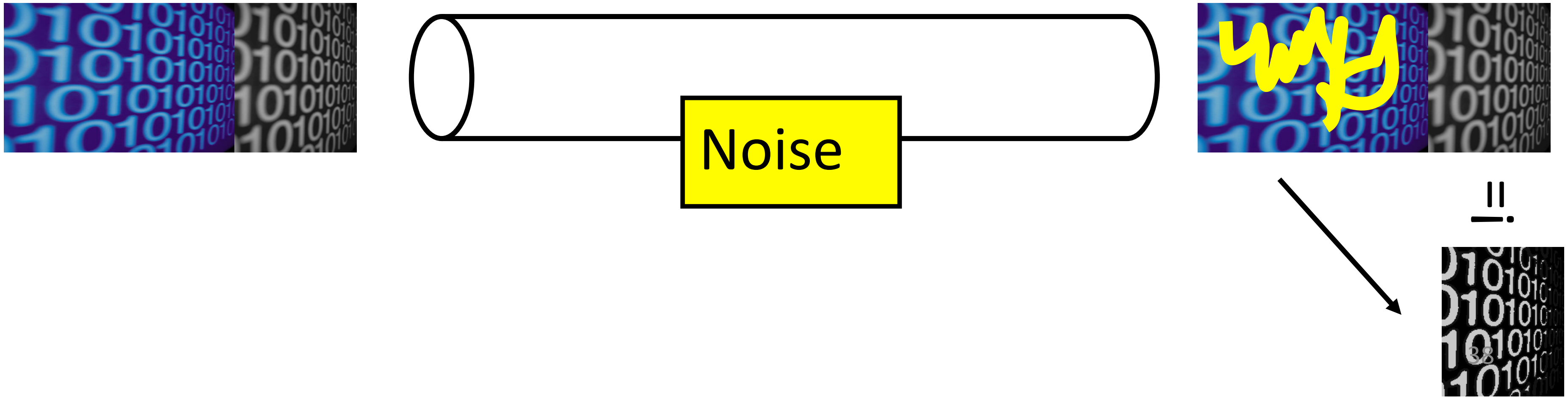
Forward Error Correction (FEC)

Sender:

```
Y = generateCheckBit(X);
send(XY);
```

Receiver:

```
receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) FIXERROR(X1Y1);
else NOERROR
```



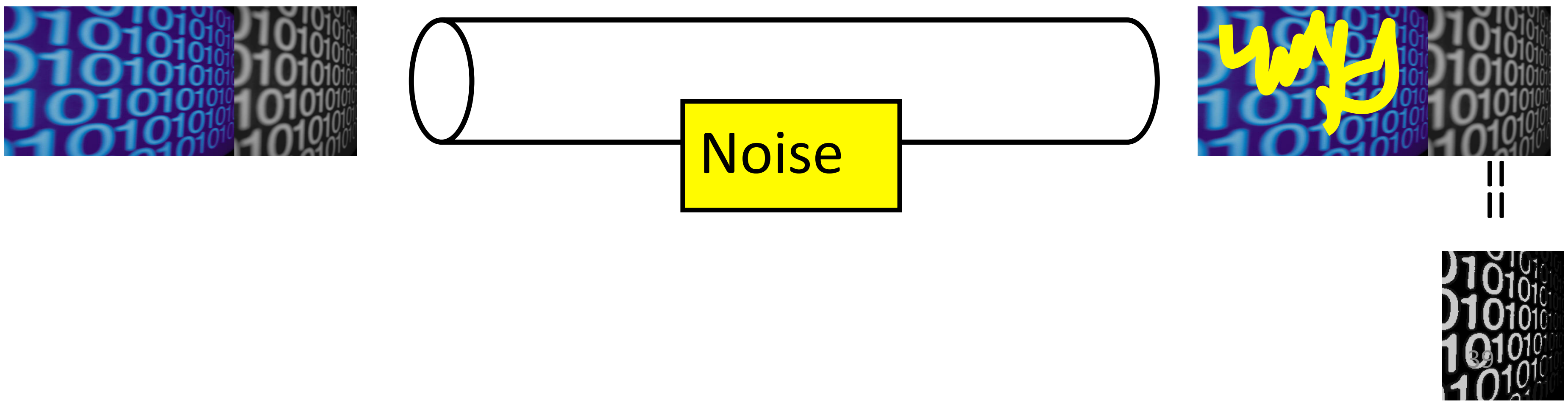
Forward Error Correction (FEC)

Sender:

```
Y = generateCheckBit(X);
send(XY);
```

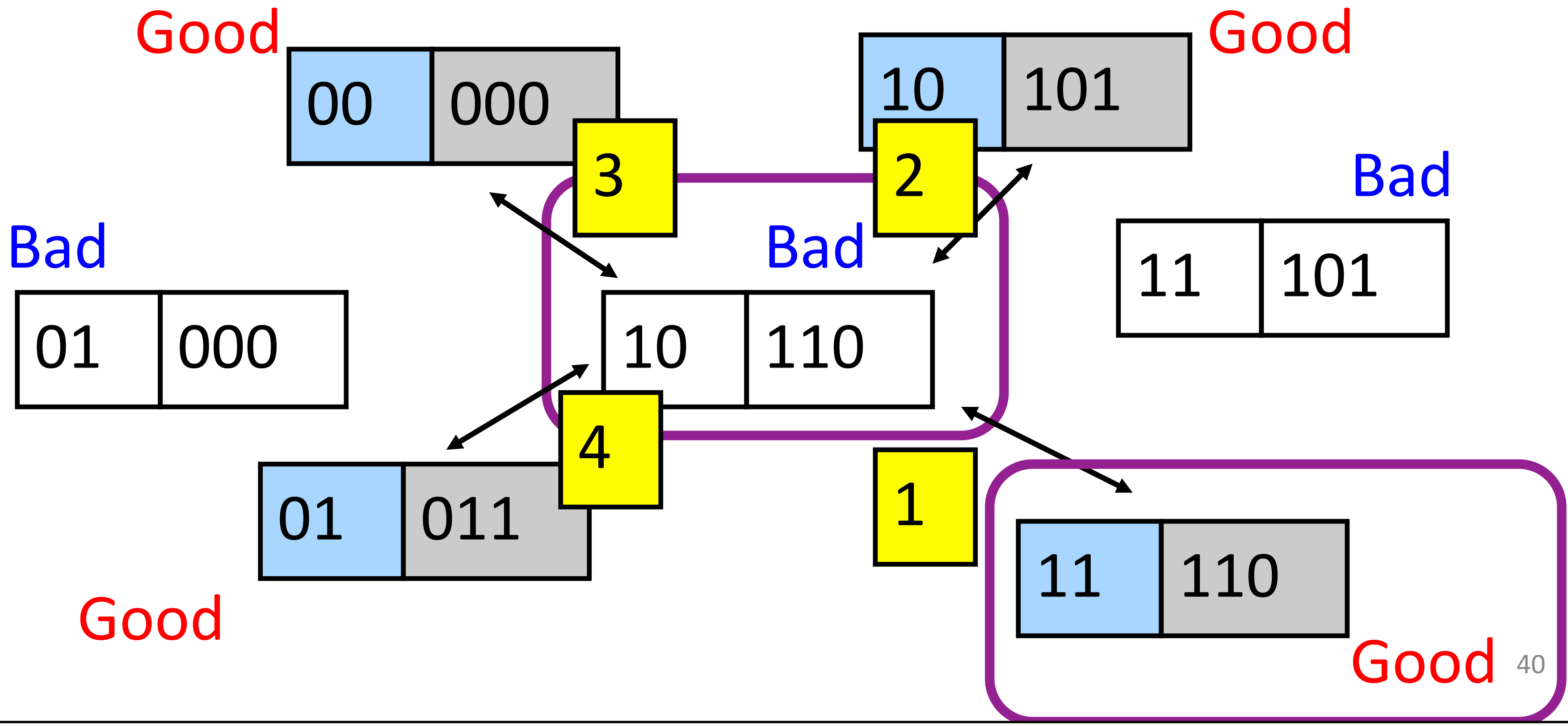
Receiver:

```
receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) FIXERROR(X1Y1);
else NOERROR
```



Basic Idea of Forward Error Correction

Replace erroneous data
by its "closest" error-free data.



Error Detection vs Correction

Error Correction:

- Cons: More check bits. False recovery.
- Pros: No need to re-send.

Error Detection:

- Cons: Need to re-send.
- Pros: Less check bits.

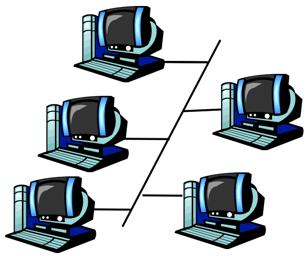
Usage:

- Correction: A lot of noise. Expensive to re-send.
- Detection: Less noise. Easy to re-send.
- Can be used together.

Multiple Access Links and Protocols

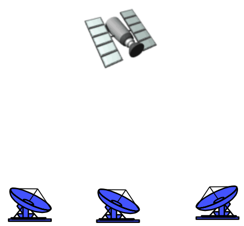
Two types of "links":

- point-to-point
  - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
  - old-fashioned wired Ethernet (*here be dinosaurs* – extinct)
  - upstream HFC (Hybrid Fiber-Coax – the Coax may be broadcast)
  - Home plug / Powerline networking
  - 802.11 wireless LAN



shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

Multiple Access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - collision if node receives two or more signals at the same time

multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

Ideal Multiple Access Protocol

Broadcast channel of rate *R* bps

1. when one node wants to transmit, it can send at rate *R*
2. when *M* nodes want to transmit, each can send at average rate *R/M*
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. simple

MAC Protocols: a taxonomy

Three broad classes:

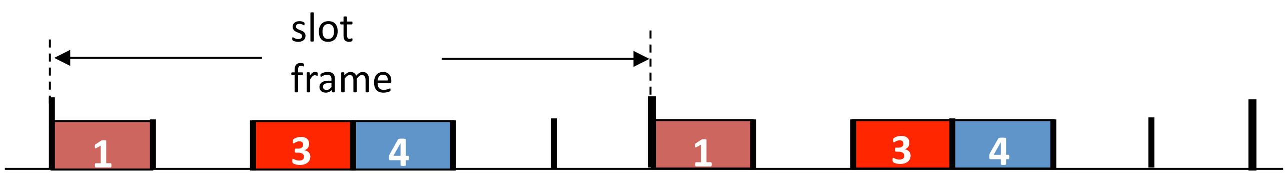
- Channel Partitioning
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- Random Access
  - channel not divided, allow collisions
  - "recover" from collisions
- "Taking turns"
  - nodes take turns, but nodes with more to send can take longer turns

Channel Partitioning MAC protocols: TDMA

(time travel warning – we mentioned this earlier)

TDMA: time division multiple access

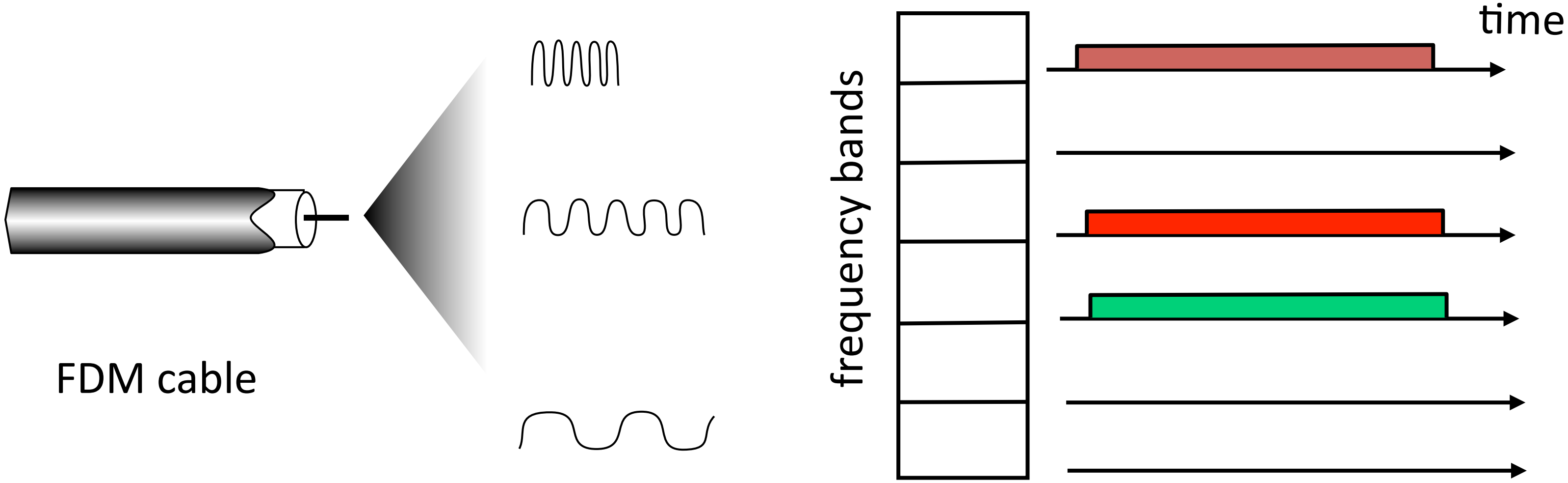
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel Partitioning MAC protocols: FDMA
(time travel warning – we mentioned this earlier)

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



"Taking Turns" MAC protocols

channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

Random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

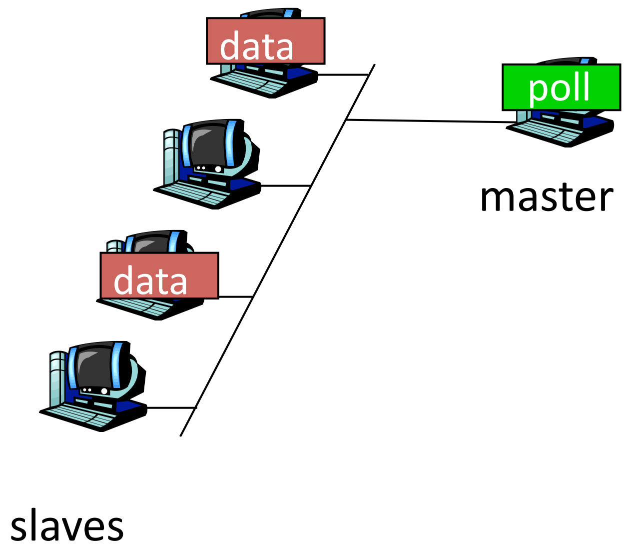
"taking turns" protocols

look for best of both worlds!

"Taking Turns" MAC protocols

Polling:

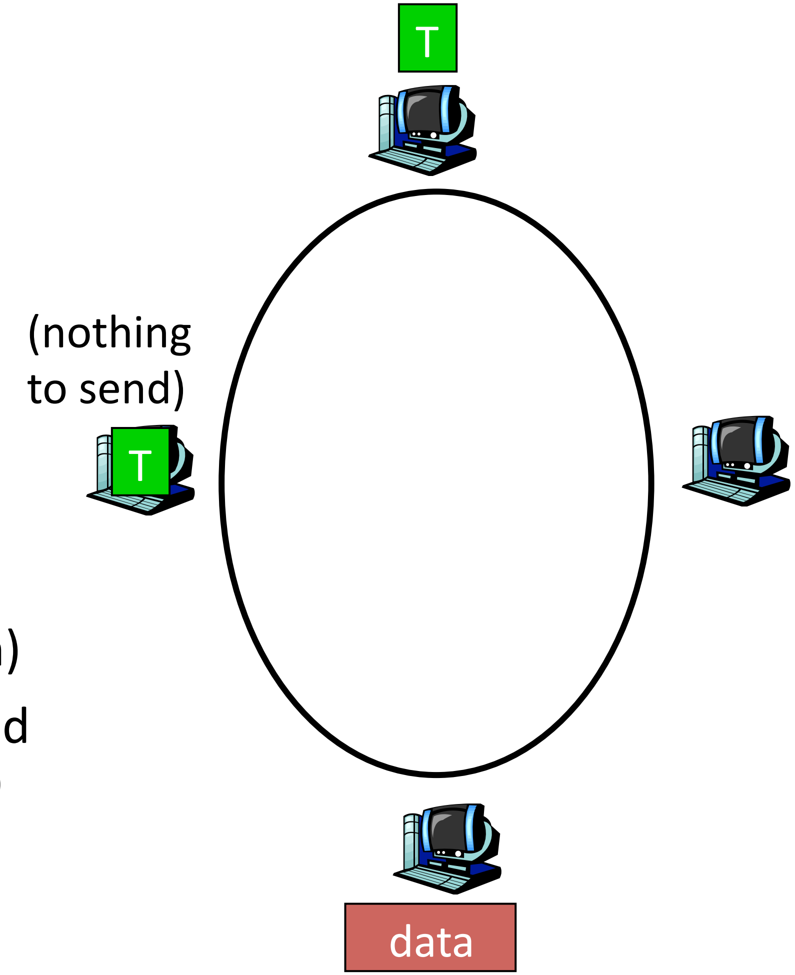
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"Taking Turns" MAC protocols

Token passing:

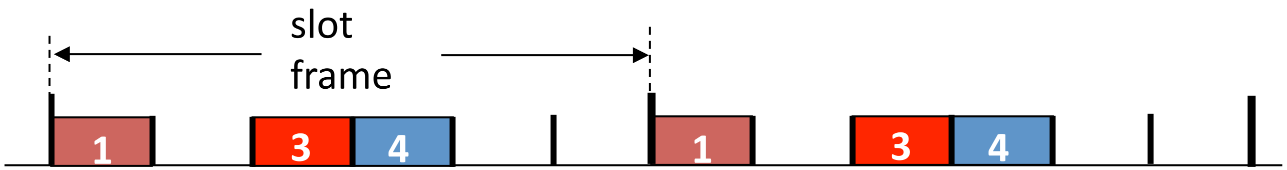
- control **token** passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)
- concerns fixed in part by a slotted ring (many simultaneous *tokens)*



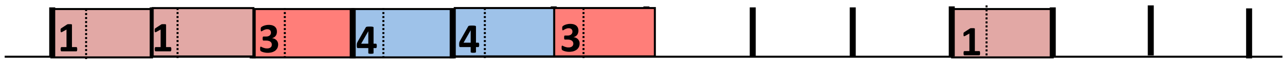
Cambridge students – this is YOUR heritage
Cambridge RING, Cambridge Fast RING,
Cambridge Backbone RING, these things gave us ATDM (and ATM)

ATM

In TDM a sender may only use a pre-allocated slot



In ATM a sender transmits labeled cells whenever necessary



ATM = Asynchronous Transfer Mode – an ugly expression
think of it as ATDM – Asynchronous Time Division Multiplexing

That's a variant of **PACKET SWITCHING** to the rest of us – just like Ethernet
but using fixed length slots/packets/cells

Use the media when you need it, but
ATM had virtual circuits and these needed setup….
Worse ATM had an utterly irrational size

Random Access MAC Protocols

- When node has packet to send
  - Transmit at full channel data rate
  - No *a priori* coordination among nodes
- Two or more transmitting nodes ⇒ collision
  - Data lost
- Random access MAC protocol specifies:
  - How to detect collisions
  - How to recover from collisions
- Examples
  - ALOHA and Slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA (wireless)

Key Ideas of Random Access

- Carrier sense
  - *Listen before speaking, and don't interrupt*
  - Checking if someone else is already sending data
  - … and waiting till the other node is done
- Collision detection
  - *If someone else starts talking at the same time, stop*
  - Realizing when two nodes are transmitting at once
  - …by detecting that the data on the wire is garbled
- Randomness
  - *Don't start talking again right away*
  - Waiting for a random time before trying again

CSMA (Carrier Sense Multiple Access)

- CSMA: listen before transmit
  - If channel sensed idle: transmit entire frame
  - If channel sensed busy, defer transmission
- Human analogy: don't interrupt others!
- Does this eliminate all collisions?
  - No, because of nonzero propagation delay

CSMA Collisions

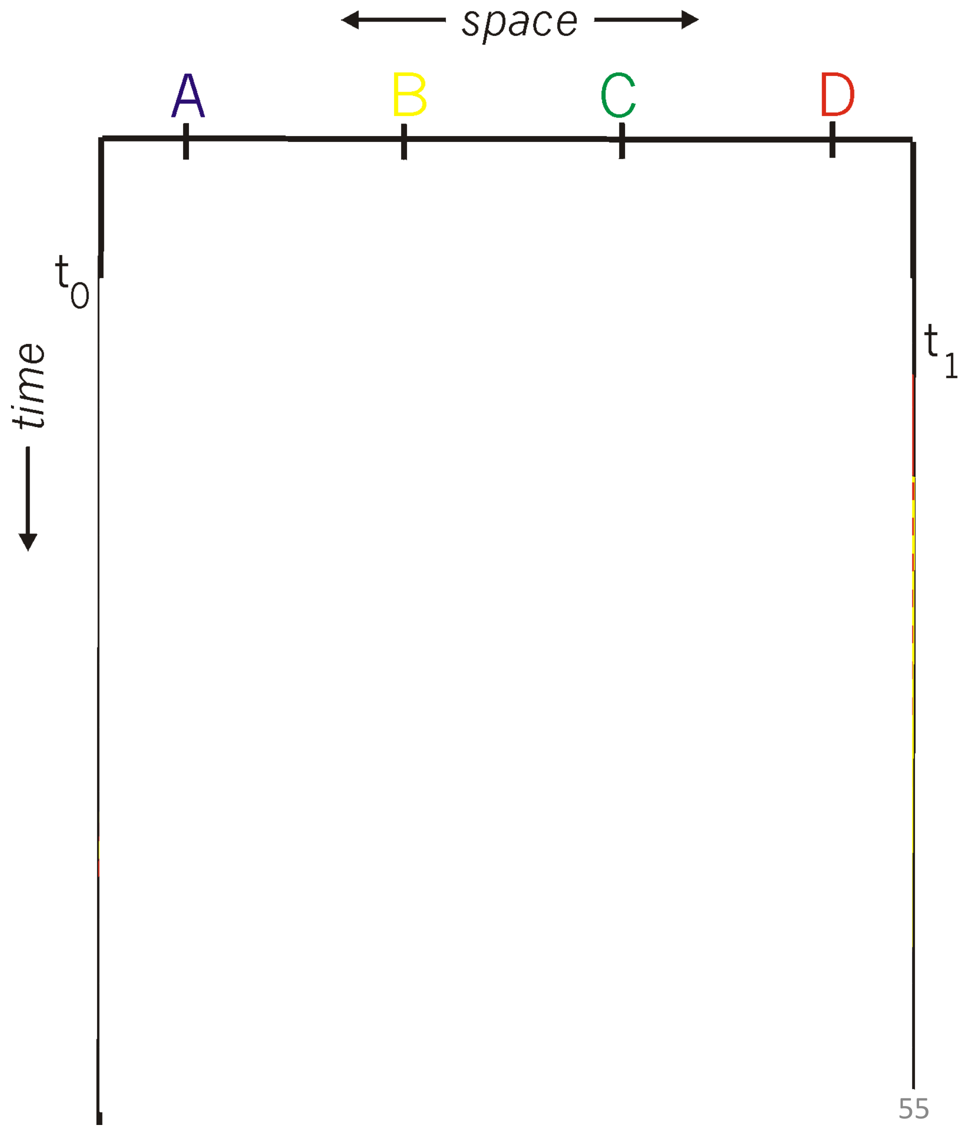
Propagation delay: two nodes may not hear each other's before sending.

Would slots hurt or help?

CSMA reduces but does not eliminate collisions

Biggest remaining problem?

Collisions still take full slot! How do you fix that?



CSMA/CD (Collision Detection)

- CSMA/CD: carrier sensing, deferral as in CSMA
  - **Collisions detected within short time**
  - Colliding transmissions aborted, reducing wastage

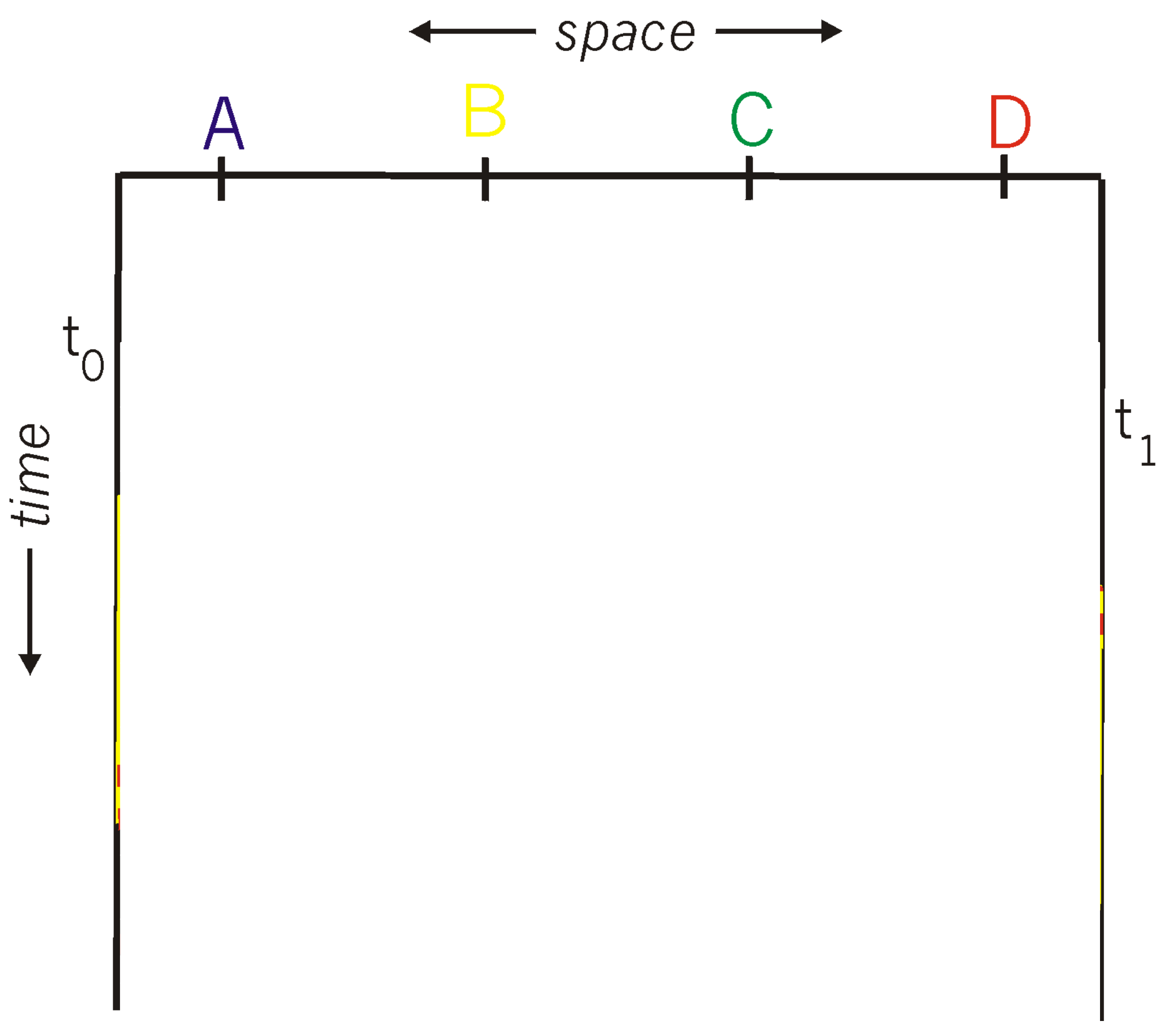
- Collision detection easy in wired LANs:
  - Compare transmitted, received signals

- Collision detection difficult in wireless LANs:
  - Reception shut off while transmitting (well, perhaps not)
  - Not perfect broadcast (limited range) so collisions local
  - Leads to use of *collision avoidance* instead (later)

CSMA/CD Collision Detection

B and D can tell that collision occurred.

Note: for this to work, need restrictions on minimum frame size and maximum distance. Why?



Limits on CSMA/CD Network Length

- Latency depends on physical length of link
  - Time to propagate a packet from one end to the other
- Suppose *A* sends a packet at time ***t***
  - And *B* sees an idle line at a time just before ***t+d***
  - … so *B* happily starts transmitting a packet
- *B* detects a collision, and sends jamming signal
  - But *A* can't see collision until ***t+2d***

Performance of CSMA/CD

- Time wasted in collisions
  - Proportional to distance d
- Time spend transmitting a packet
  - Packet length p divided by bandwidth b
- Rough estimate for efficiency (K some constant)

$$E \sim \frac{\frac{p}{b}}{\frac{p}{b} + Kd}$$

- Note:
  - For large packets, small distances, E ~ 1
  - As bandwidth increases, E decreases
  - That is why high-speed LANs are all switched

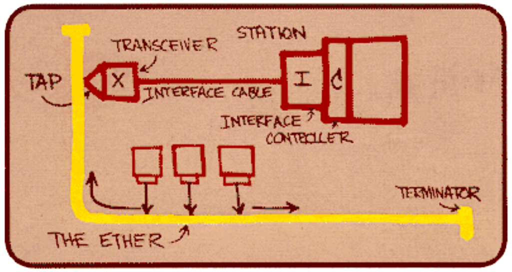
Benefits of Ethernet

- Easy to administer and maintain
- Inexpensive
- Increasingly higher speed
- Evolvable!

Evolution of Ethernet

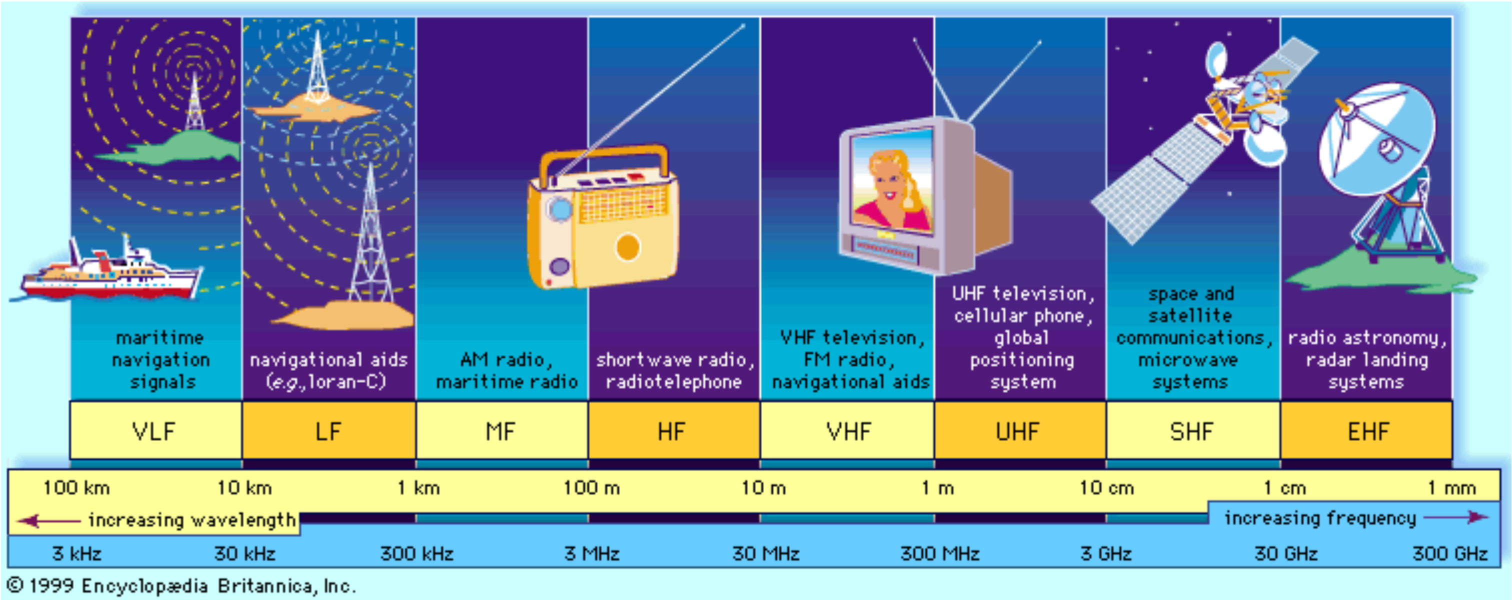
- Changed everything except the frame format
  - From single coaxial cable to hub-based star
  - From shared media to switches
  - From electrical signaling to optical
- Lesson #1
  - The right interface can accommodate many changes
  - Implementation is hidden behind interface
- Lesson #2
  - Really hard to displace the dominant technology
  - Slight performance improvements are not enough

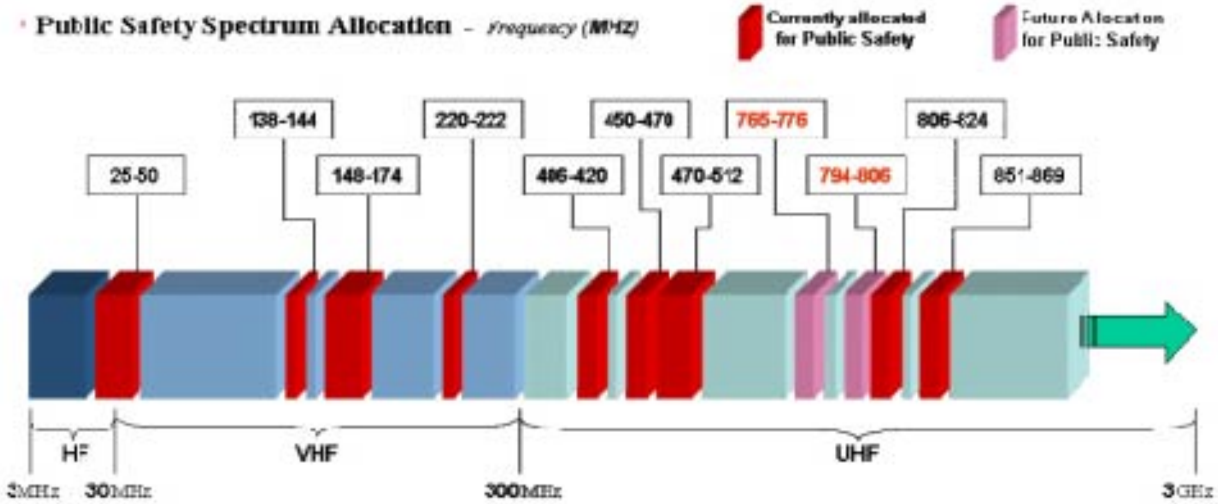
Ethernet: CSMA/CD Protocol



- **Carrier sense**: wait for link to be idle
- **Collision detection**: listen while transmitting
  - No collision: transmission is complete
  - Collision: abort transmission & send **jam** signal
- **Random access**: binary exponential back-off
  - After collision, wait a random time before trying again
  - After m^{th} collision, choose K randomly from {0, …, 2^m-1}
  - … and wait for K*512 bit times before trying again
    - Using min packet size as "slot"
    - **If transmission occurring when ready to send, wait until end of transmission (CSMA)**

The Wireless Spectrum





Metrics for evaluation / comparison of wireless technologies

- Bitrate or Bandwidth
- Range - PAN, LAN, MAN, WAN
- Two-way / One-way
- Multi-Access / Point-to-Point
- Digital / Analog
- Applications and industries
- Frequency – Affects most physical properties:
  Distance (free-space loss)
  Penetration, Reflection, Absorption
  Energy proportionality
  Policy: Licensed / Deregulated
  Line of Sight (Fresnel zone)
  Size of antenna

➢ Determined by wavelength – $\lambda = \frac{v}{f},$)

Wireless Communication Standards

- Cellular (800/900/*1700*/1800/1900Mhz):
  - 2G: GSM / CDMA / GPRS /EDGE
  - 3G: CDMA2000/UMTS/HSDPA/EVDO
  - 4G: LTE, WiMax
- IEEE 802.11 (aka WiFi):
  - b: 2.4Ghz band, 11Mbps (~*4.5 Mbps operating rate*)
  - g: 2.4Ghz, 54-108Mbps (~*19 Mbps operating rate*)
  - a: 5.0Ghz band, 54-108Mbps (~*25 Mbps operating rate*)
  - n: 2.4/5Ghz, 150-600Mbps (4x4 mimo).
- IEEE 802.15 – lower power wireless:
  - 802.15.1: 2.4Ghz, 2.1 Mbps (Bluetooth)
  - 802.15.4: 2.4Ghz, 250 Kbps (Sensor Networks)

What Makes Wireless Different?

- Broadcast and multi-access medium…
  - err, so….
- BUT, Signals sent by sender don't always end up at receiver intact
  - Complicated physics involved, which we won't discuss
  - But what can go wrong?

Path Loss / Path Attenuation

- Free Space Path Loss:

  d = distance

  λ = wave length

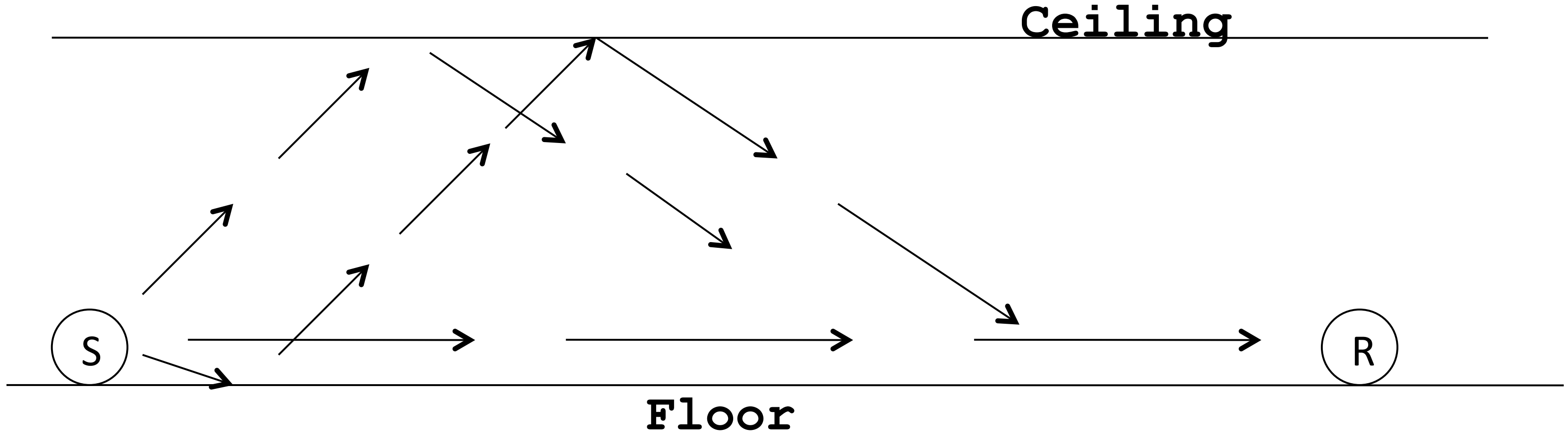
  f = frequency

  c = speed of light

$$\mathrm{FSPL} = \left(\frac{4\pi d}{\lambda}\right)^2 = \left(\frac{4\pi d f}{c}\right)^2$$

- Reflection, Diffraction, Absorption
- Terrain contours (Urban, Rural, Vegetation).
- Humidity

Multipath Effects



- Signals bounce off surface and interfere with one another
- Self-interference

Interference from Other Sources

- External Interference
  - Microwave is turned on and blocks your signal
  - Would that affect the sender or the receiver?
- Internal Interference
  - Hosts within range of each other collide with one another's transmission

- We have to tolerate path loss, multipath, etc., but we can try to avoid internal interference

Wireless Bit Errors

- The lower the SNR (Signal/Noise) the higher the Bit Error Rate (BER)
- We could make the signal stronger…
- Why is this not always a good idea?
  - Increased signal strength requires more power
  - Increases the interference range of the sender, so you interfere with more nodes around you
    - And then they increase their power…….
- Local link-layer Error Correction schemes can correct some problems

Lets focus on 802.11

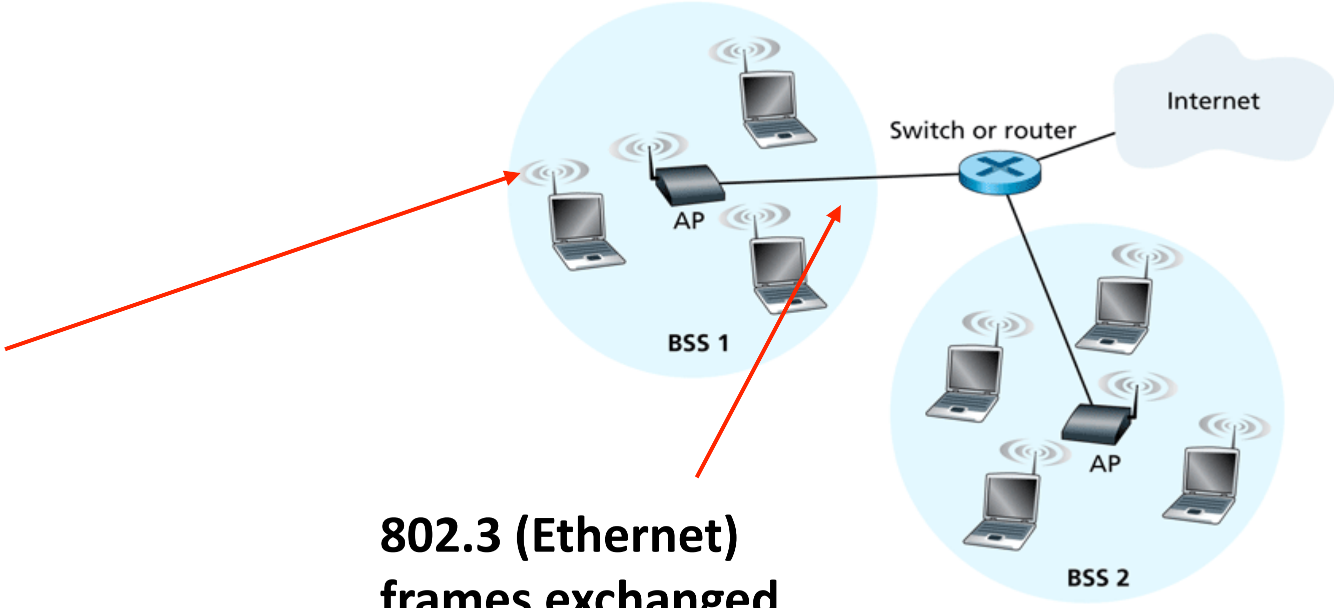
aka - WiFi …
What makes it special?

Deregulation > Innovation > Adoption > Lower cost = Ubiquitous technology

JUST LIKE ETHERNET – not lovely but sufficient

802.11 Architecture

802.11 frames exchanges



802.3 (Ethernet) frames exchanged

Figure 6.7 ♦ IEEE 802.11 LAN architecture

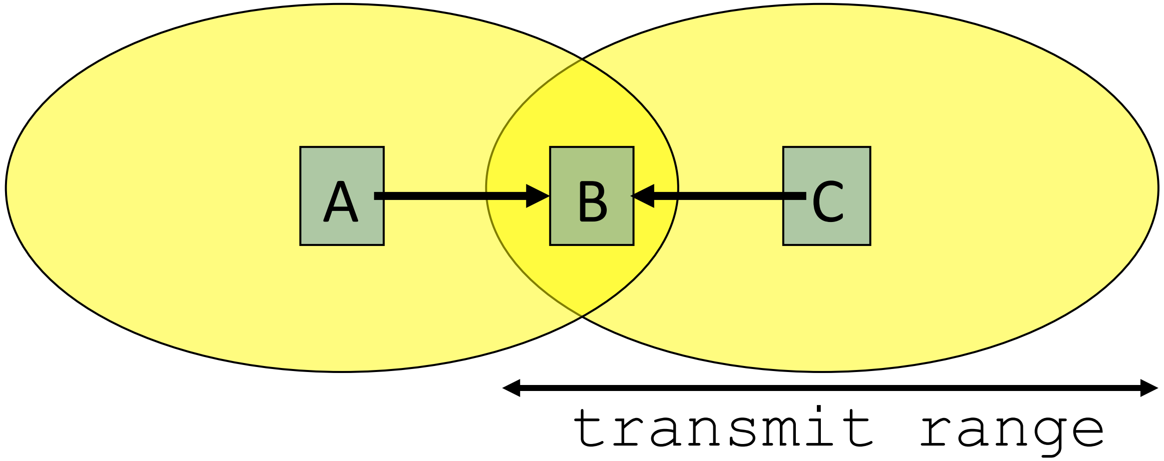
- Designed for limited area
- AP's (Access Points) set to specific channel
- Broadcast beacon messages with SSID (Service Set Identifier) and MAC Address periodically
- Hosts scan all the channels to discover the AP's
  - Host associates with AP

Wireless Multiple Access Technique?

- Carrier Sense?
  - Sender can listen before sending
  - What does that tell the sender?

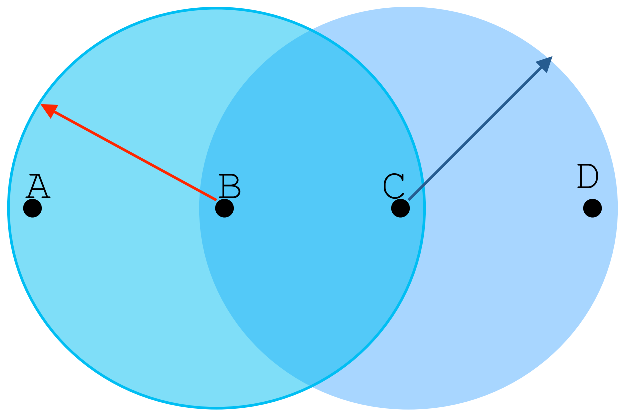
- Collision Detection?
  - Where do collisions occur?
  - How can you detect them?

Hidden Terminals



- A and C can both send to B but can't hear each other
  - A is a *hidden terminal* for C and vice versa
- Carrier Sense will be ineffective

Exposed Terminals



- Exposed node: B sends a packet to A; C hears this and decides not to send a packet to D (despite the fact that this will not cause interference)!
- Carrier sense would prevent a successful transmission.

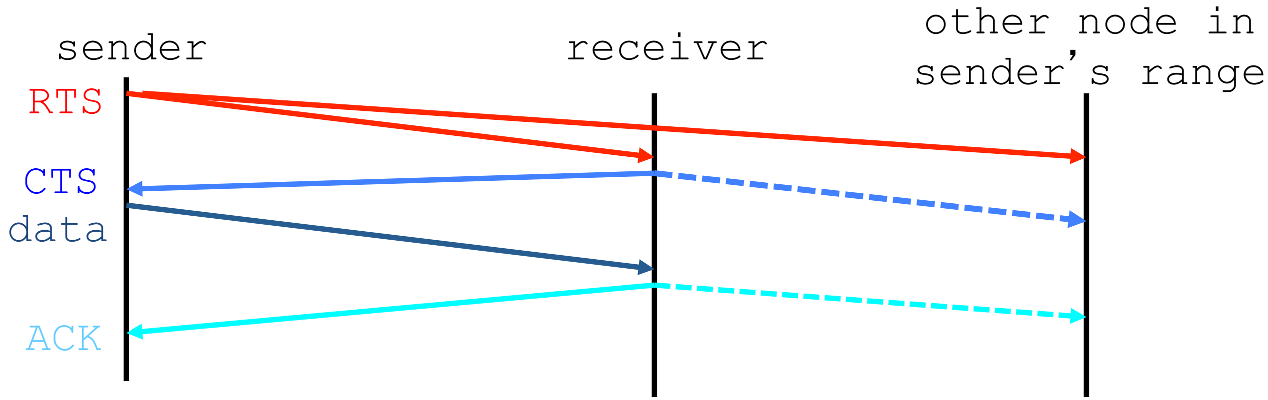
Key Points

- No concept of a global collision
  - Different receivers hear different signals
  - Different senders reach different receivers
- Collisions are at receiver, not sender
  - Only care if receiver can hear the sender clearly
  - It does not matter if sender can hear someone else
  - As long as that signal does not interfere with receiver
- Goal of protocol:
  - Detect if receiver can hear sender
  - Tell senders who might interfere with receiver to shut up

Basic Collision Avoidance

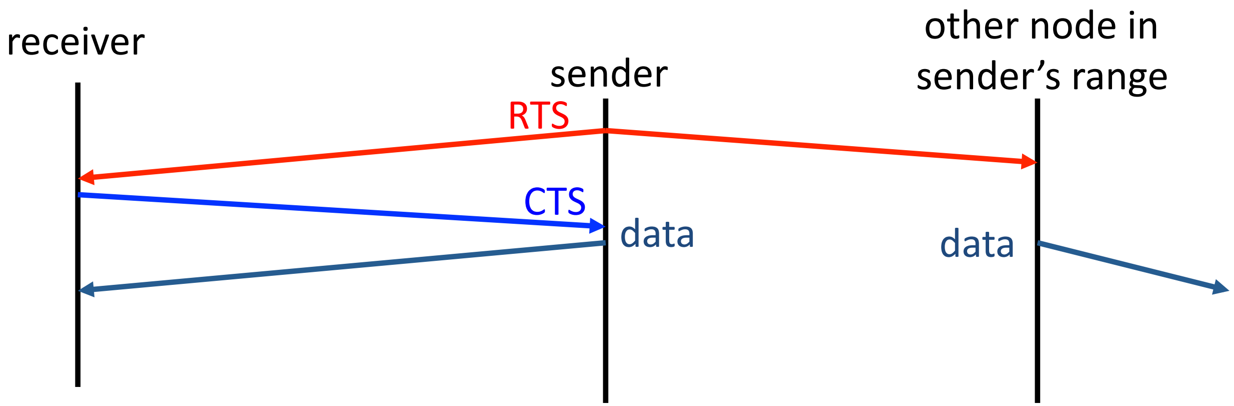
- Since can't detect collisions, we try to *avoid* them
- Carrier sense:
  - When medium busy, choose random interval
  - Wait that many **idle** timeslots to pass before sending
- When a collision is inferred, retransmit with binary exponential backoff (like Ethernet)
  - Use ACK from receiver to infer "no collision"
  - Use exponential backoff to adapt contention window

CSMA/CA -MA with Collision Avoidance



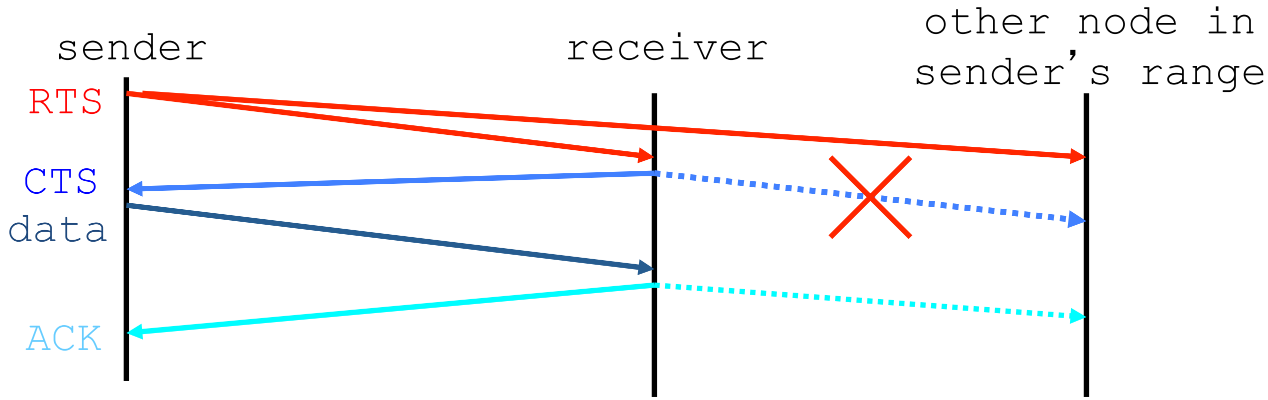
- Before every data transmission
  - Sender sends a Request to Send (RTS) frame containing the length of the transmission
  - Receiver respond with a Clear to Send (CTS) frame
  - Sender sends data
  - Receiver sends an ACK; now another sender can send data
- When sender doesn't get a CTS back, it assumes collision

CSMA/CA, con’t



- If other nodes hear RTS, but not CTS: send
  - Presumably, destination for first sender is out of node’s range …

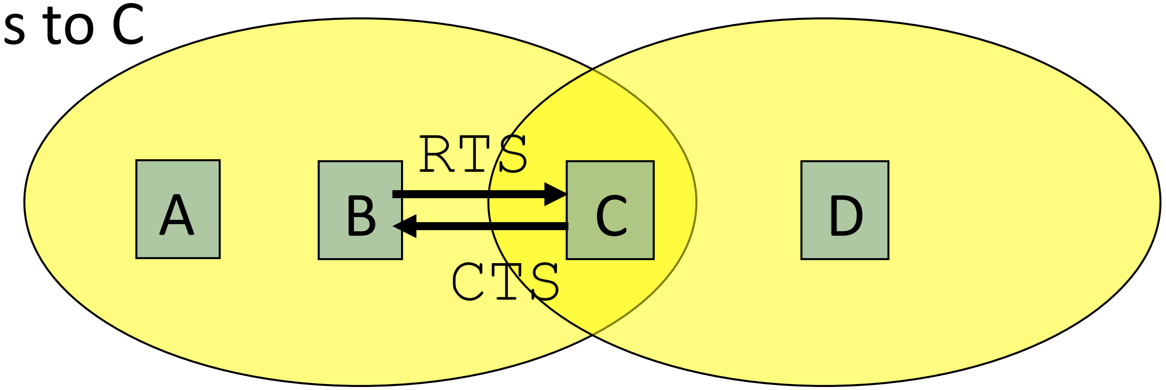
CSMA/CA, con't



- If other nodes hear RTS, but not CTS: send
  – Presumably, destination for first sender is out of node's range …
  – … Can cause problems when a CTS is lost
- When you hear a CTS, you keep quiet until scheduled transmission is over (hear ACK)

RTS / CTS Protocols (CSMA/CA)

B sends to C

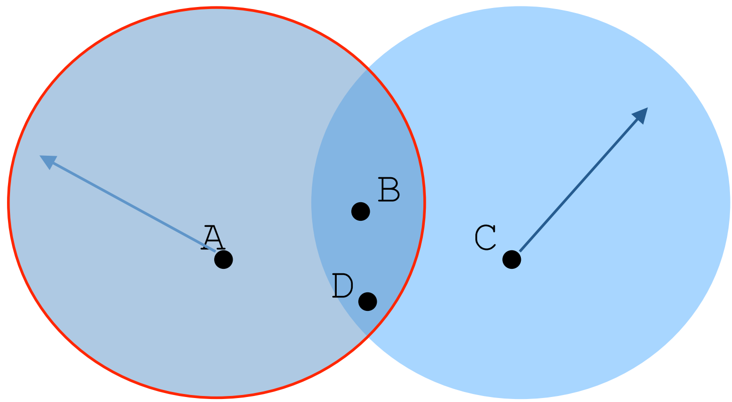


Overcome hidden terminal problems with contention-free protocol

1. B sends to C Request To Send (RTS)
2. A hears RTS and defers (to allow C to answer)
3. C replies to B with Clear To Send (CTS)
4. D hears CTS and defers to allow the data
5. B sends to C

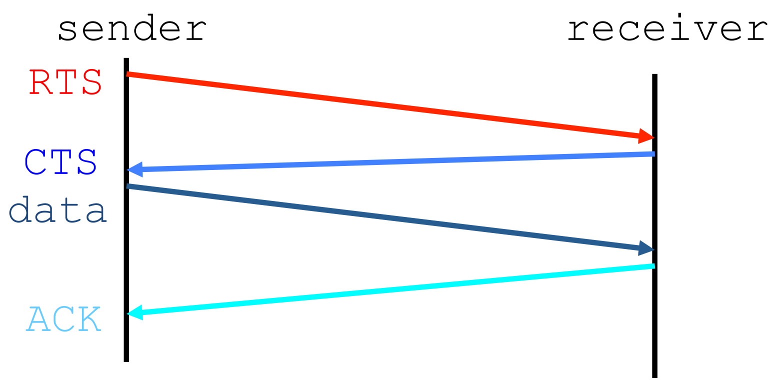
Preventing Collisions Altogether

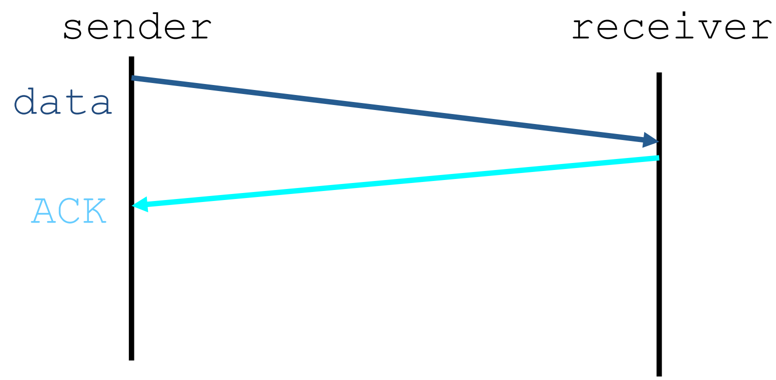
- Frequency Spectrum partitioned into several channels
  - Nodes within interference range can use separate channels



  - Now A and C can send without any interference!
- Most cards have only 1 transceiver
  - **Not Full Duplex: Cannot send and receive at the same time**
  - Aggregate Network throughput doubles

CSMA/CA and RTS/CTS





RTS/CTS

- helps with hidden terminal
- good for high-traffic Access Points
- often turned on/off dynamically

Without RTS/CTS

- lower latency -> faster!
- reduces wasted b/w

  if the *Pr(collision)* is low
- good for when net is small and not *weird*

  eg no hidden/exposed terminals

CSMA/CD vs CSMA/CA (without RTS/CTS)

CD Collision Detect

wired – listen and talk

1. Listen for others
2. Busy? goto 1.
3. Send message (and listen)
4. Collision?
   a. JAM
   b. increase your BEB
   c. sleep
   d. goto 1.

CA Collision Avoidance

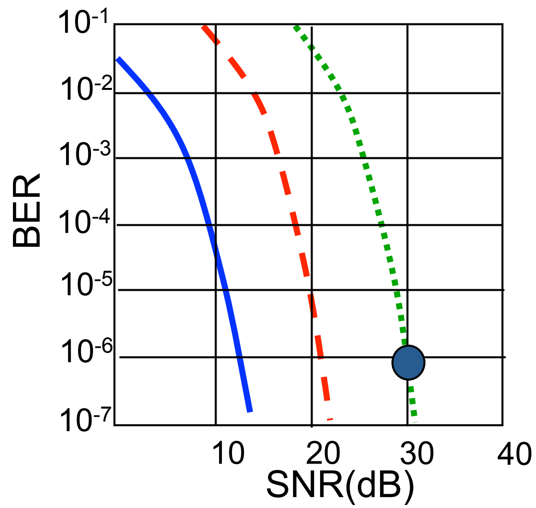
wireless – talk OR listen

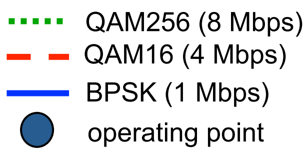
1. Listen for others
2. Busy?
   a. increase your BEB
   b. sleep
   c. goto 1.
3. Send message
4. Wait for ACK (*MAC ACK*)
5. Got No ACK from MAC?
   a. increase your BEB
   b. sleep
   c. goto 1.

Changing the rules: an 802.11 feature

Rate Adaptation

- base station, mobile dynamically change transmission rate (physical layer modulation technique) as mobile moves, SNR varies





1. SNR decreases, BER increase as node moves away from base station

2. When BER becomes too high, switch to lower transmission rate but with lower BER

Summary of MAC protocols

- *channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing
  - Bluetooth, FDDI, IBM Token Ring

MAC Addresses

- MAC (or LAN or physical or Ethernet) address:
  - function: *get frame from one interface to another physically-connected interface (same network)*
  - 48 bit MAC address (for most LANs)
    - *burned* in NIC ROM, nowadays usually software settable and set at boot time

```
awm22@rio:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:30:48:fe:c0:64
          inet addr:128.232.33.4  Bcast:128.232.47.255  Mask:255.255.240.0
          inet6 addr: fe80::230:48ff:fefe:c064/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:215084512 errors:252 dropped:25 overruns:0 frame:123
          TX packets:146711866 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:170815941033 (170.8 GB)  TX bytes:86755864270 (86.7 GB)
          Memory:f0000000-f0020000
```

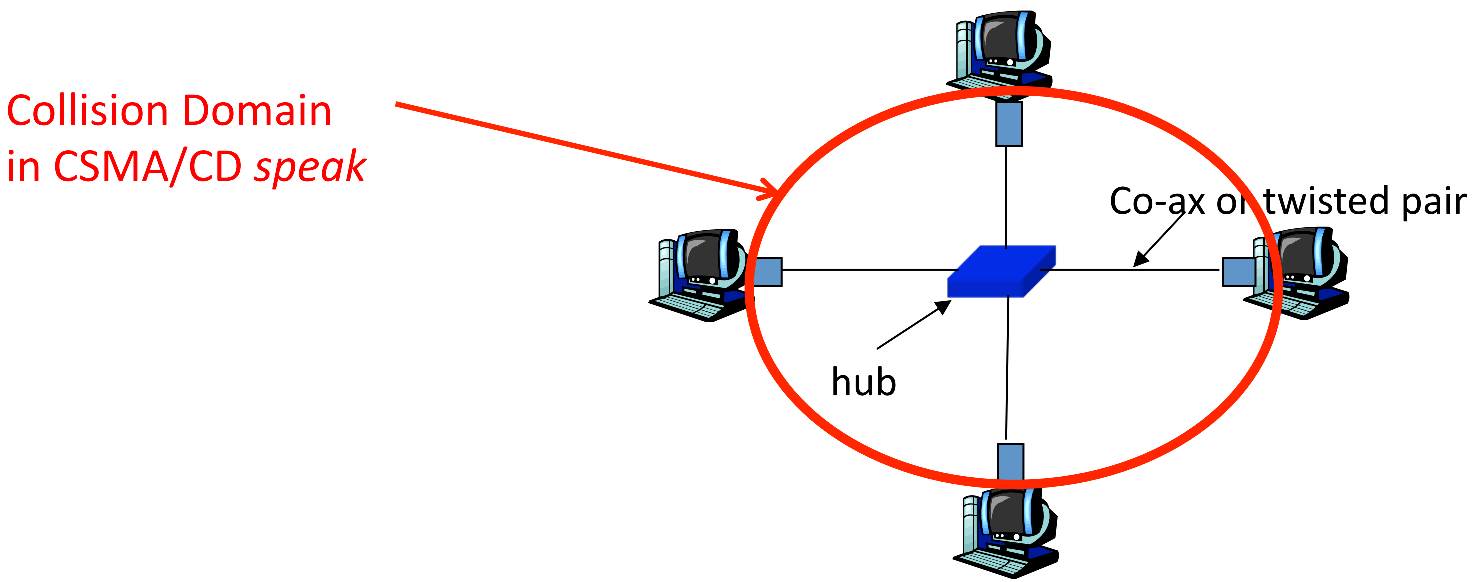
LAN Address (more)

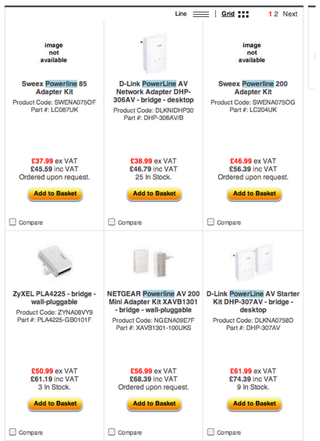
- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - (a) MAC address: like Social Security Number
  - (b) IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - address depends on IP subnet to which node is attached

Hubs

… physical-layer ("dumb") repeaters:

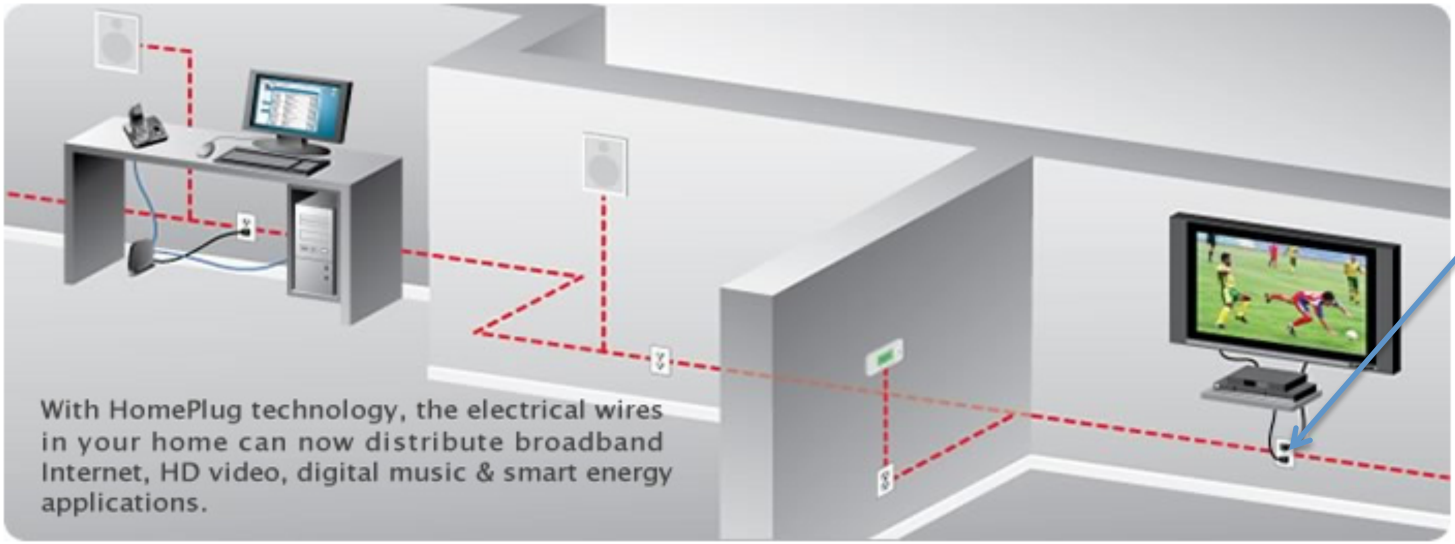
- bits coming in one link go out *all* other links at same rate
- all nodes connected to hub can collide with one another
- no frame buffering
- no CSMA/CD at hub: host NICs detect collisions





CSMA/CD Lives....

Home Plug and similar Powerline Networking....



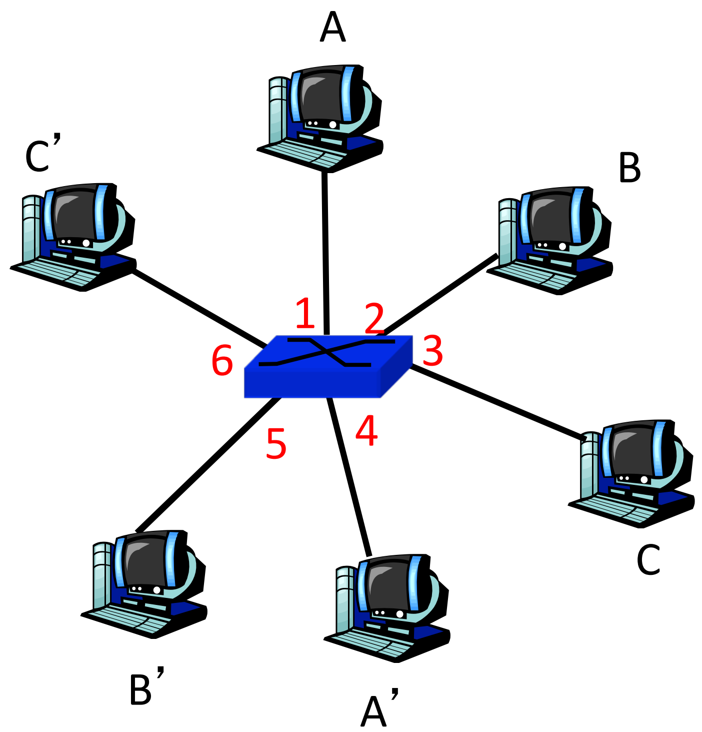
Switch

(like a Hub but smarter)

- link-layer device: smarter than hubs, take *active* role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- *transparent*
  - hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

Switch: allows *multiple* simultaneous transmissions

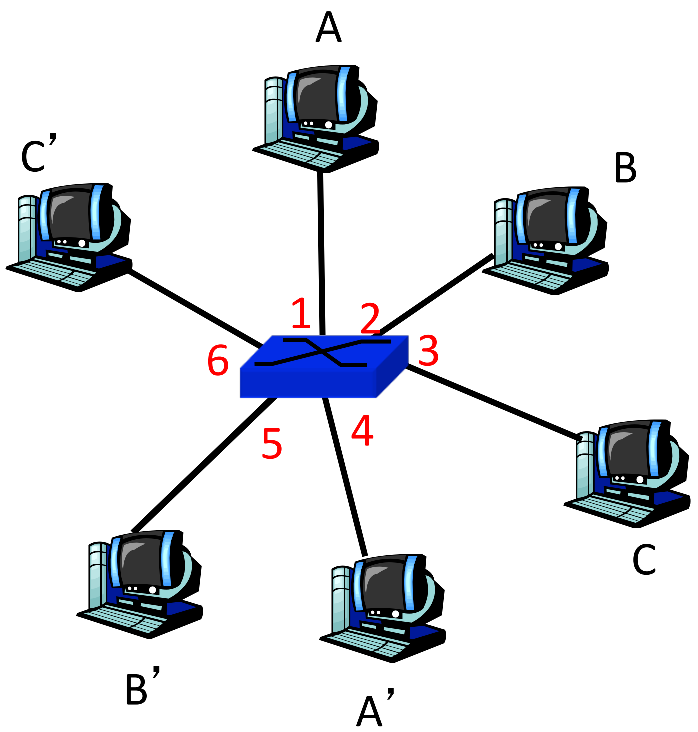
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' simultaneously, without collisions
  - not possible with dumb hub



switch with six interfaces (1,2,3,4,5,6)

Switch Table

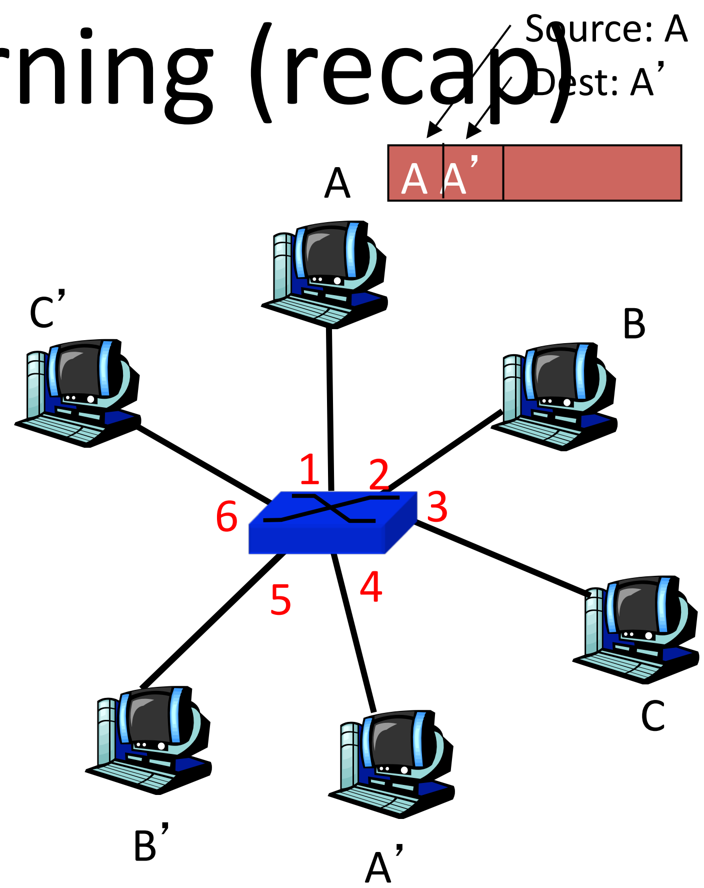
- *Q:* how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- *A:* each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- *Q:* how are entries created, maintained in switch table?
  - something like a routing protocol?



switch with six interfaces (1,2,3,4,5,6)

Switch: self-learning (recap)

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Switch: frame filtering/forwarding

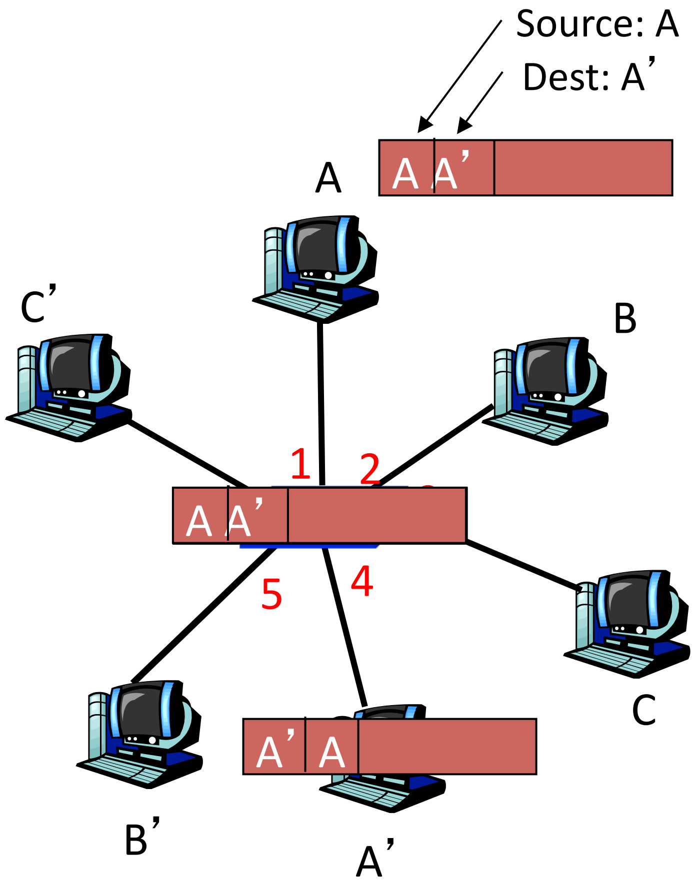
When frame received:

1. record link associated with sending host
2. index switch table using MAC dest address
3. **if** entry found for destination
   then {
   if dest on segment from which frame arrived
   then drop the frame
   else forward the frame on interface indicated
   }
   else flood

forward on all but the interface on which the frame arrived

Self-learning, forwarding: example

- frame destination unknown: *flood*
- destination A location known: selective send

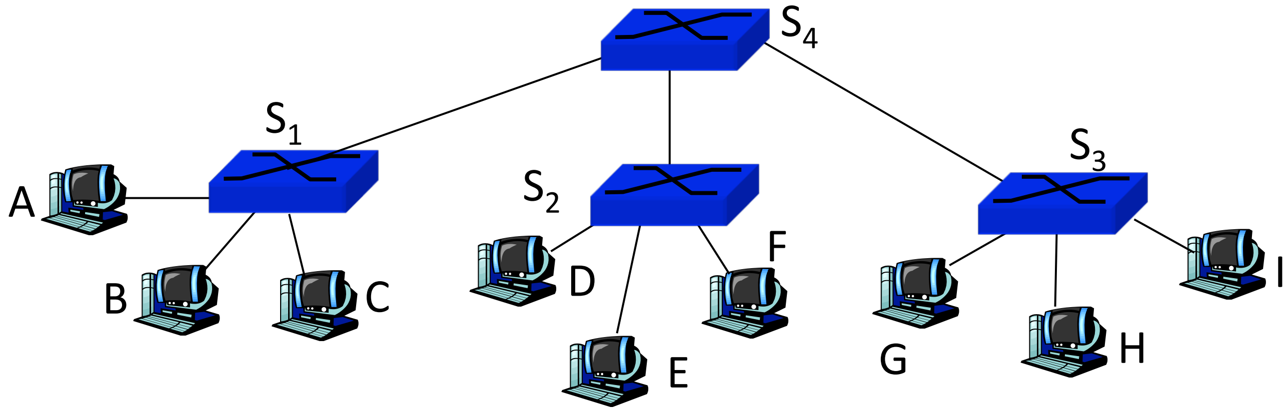


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

Switch table (initially empty)

Interconnecting switches

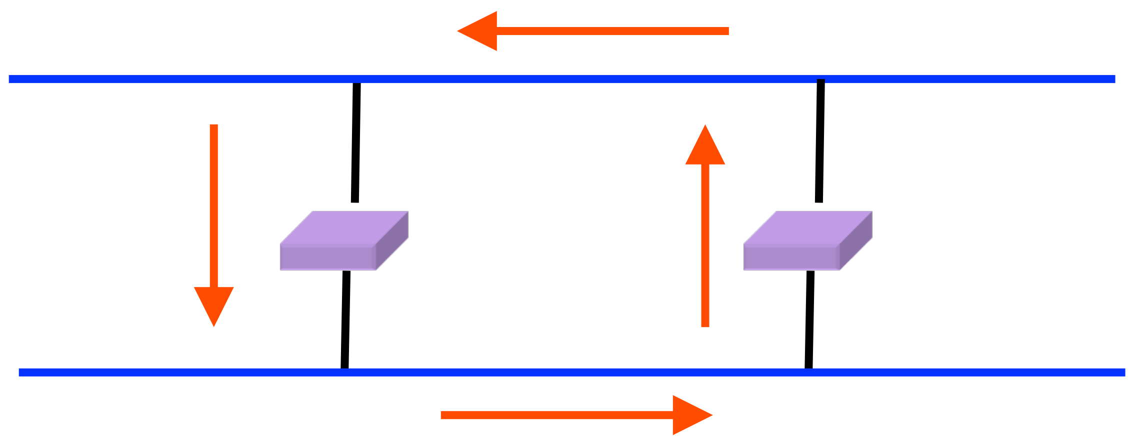
- switches can be connected together



- Q: sending from A to G - how does S_1 know to forward frame destined to F via S_4 and S_3?
- A: self learning! (works exactly the same as in single-switch case – flood/forward/drop)

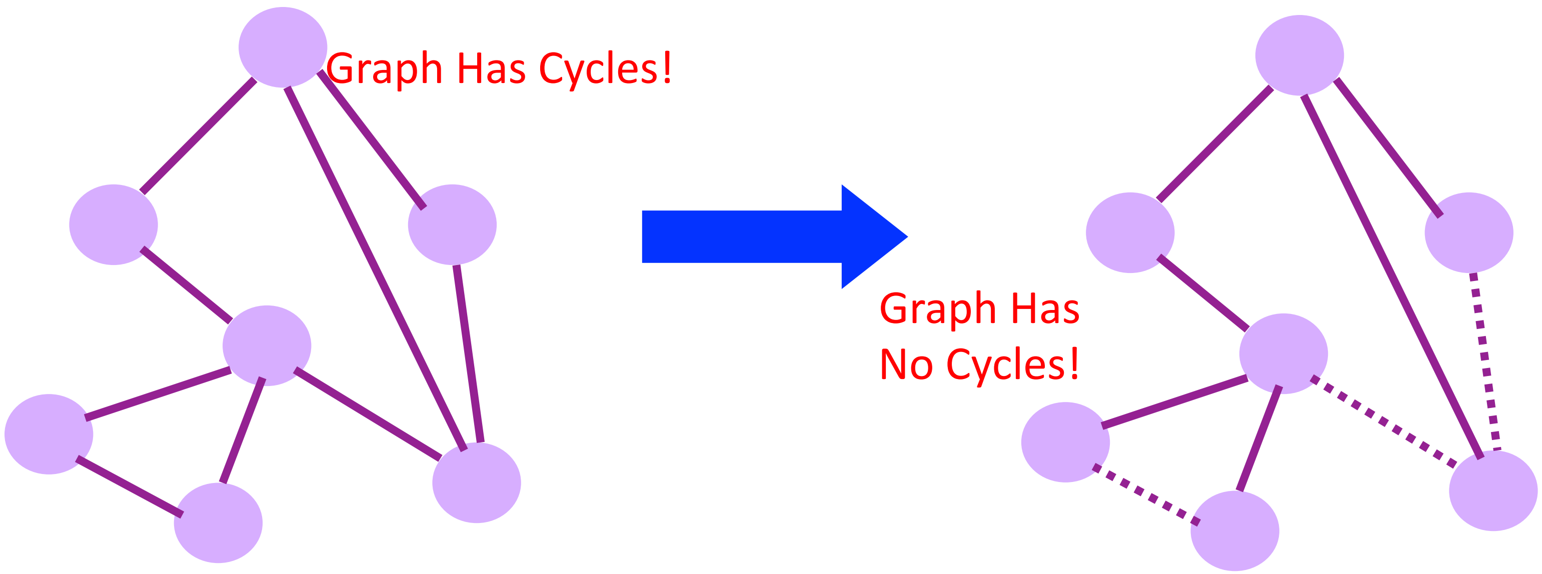
Flooding Can Lead to Loops

- Flooding can lead to forwarding loops
  - E.g., if the network contains a cycle of switches
  - "Broadcast storm"



Solution: Spanning Trees

- Ensure the forwarding topology has no loops
  - Avoid using some of the links when flooding
  - … to prevent loop from forming
- Spanning tree
  - Sub-graph that covers all vertices but *contains no cycles*
  - Links not in the spanning tree do not forward frames

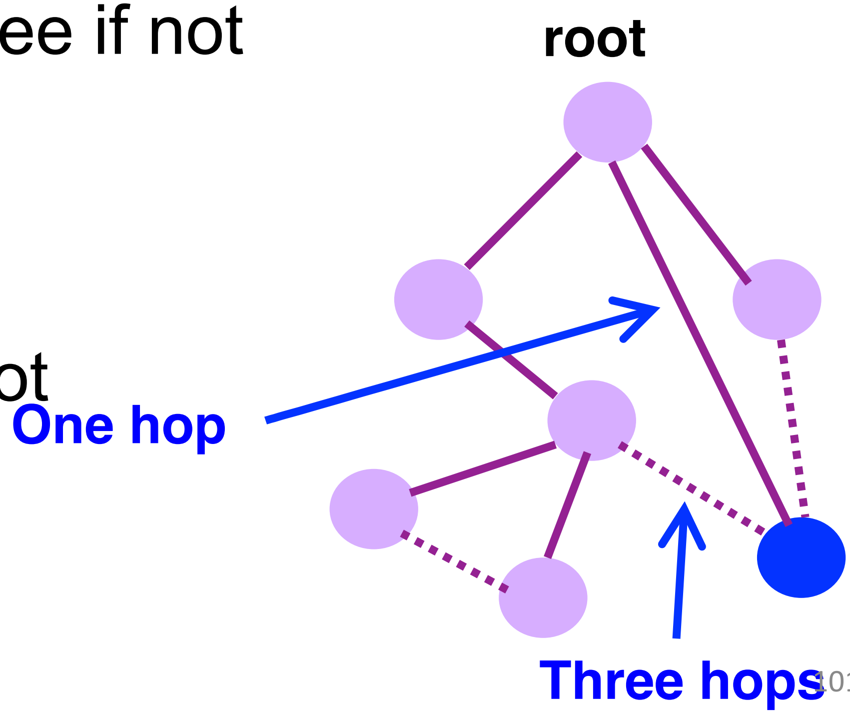


What Do We Know?

- Shortest paths to (or from) a node form a tree
- So, algorithm has two aspects :
  - Pick a root
  - Compute shortest paths to it
- Only keep the links on shortest-path

Constructing a Spanning Tree

- Switches need to elect a root
  - The switch w/ smallest identifier (MAC addr)
- Each switch determines if each interface is on the shortest path from the root
  - Excludes it from the tree if not
- Messages (Y, d, X)
  - From node X
  - Proposing Y as the root
  - And the distance is d

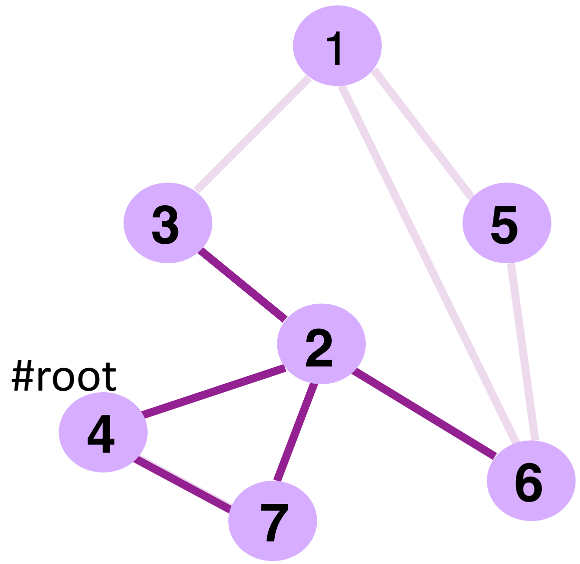


Steps in Spanning Tree Algorithm

- Initially, each switch proposes itself as the root
  - Switch sends a message out every interface
  - … proposing itself as the root with distance 0
  - Example: switch X announces (X, 0, X)
- Switches update their view of the root
  - Upon receiving message (Y, d, Z) from Z, check Y's id
  - If new id smaller, start viewing that switch as root
- Switches compute their distance from the root
  - Add 1 to the distance received from a neighbor
  - Identify interfaces not on shortest path to the root
  - … and exclude them from the spanning tree
- If root or shortest distance to it changed, "flood" updated message (Y, d+1, X)

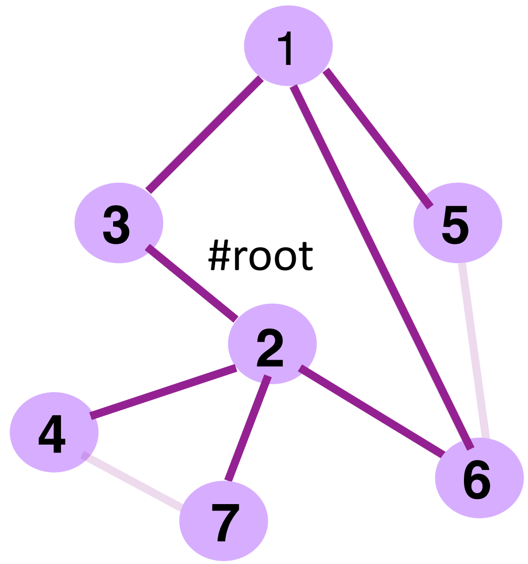
Example From Switch #4's Viewpoint

- Switch #4 thinks it is the root
  - Sends (4, 0, 4) message to 2 and 7
- Then, switch #4 hears from #2
  - Receives (2, 0, 2) message from 2
  - … and thinks that #2 is the root
  - And realizes it is just one hop away
- Then, switch #4 hears from #7
  - Receives (2, 1, 7) from 7
  - And realizes this is a longer path
  - So, prefers its own one-hop path
  - And removes 4-7 link from the tree



Example From Switch #4's Viewpoint

- Switch #2 hears about switch #1
  - Switch 2 hears (1, 1, 3) from 3
  - Switch 2 starts treating 1 as root
  - And sends (1, 2, 2) to neighbors
- Switch #4 hears from switch #2
  - Switch 4 starts treating 1 as root
  - And sends (1, 3, 4) to neighbors
- Switch #4 hears from switch #7
  - Switch 4 receives (1, 3, 7) from 7
  - And realizes this is a longer path
  - So, prefers its own three-hop path
  - And removes 4-7 link from the tree



Robust Spanning Tree Algorithm

- Algorithm must react to failures
  - Failure of the root node
    - Need to elect a new root, with the next lowest identifier
  - Failure of other switches and links
    - Need to recompute the spanning tree
- Root switch continues sending messages
  - Periodically reannouncing itself as the root (1, 0, 1)
  - Other switches continue forwarding messages
- Detecting failures through timeout (soft state)
  - If no word from root, times out and claims to be the root
  - Delay in reestablishing spanning tree is ***major problem***
  - Work on rapid spanning tree algorithms…

Topic 3: Summary

- principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation and implementation of various link layer technologies
  - Ethernet
  - switched LANS
  - WiFi
- algorithms
  - Binary Exponential Backoff
  - Spanning Tree