Introduction to Layering and Network Layering

COS 316: Principles of Computer System Design

Lecture 8

Amit Levy & Ravi Netravali

Barbara Liskov

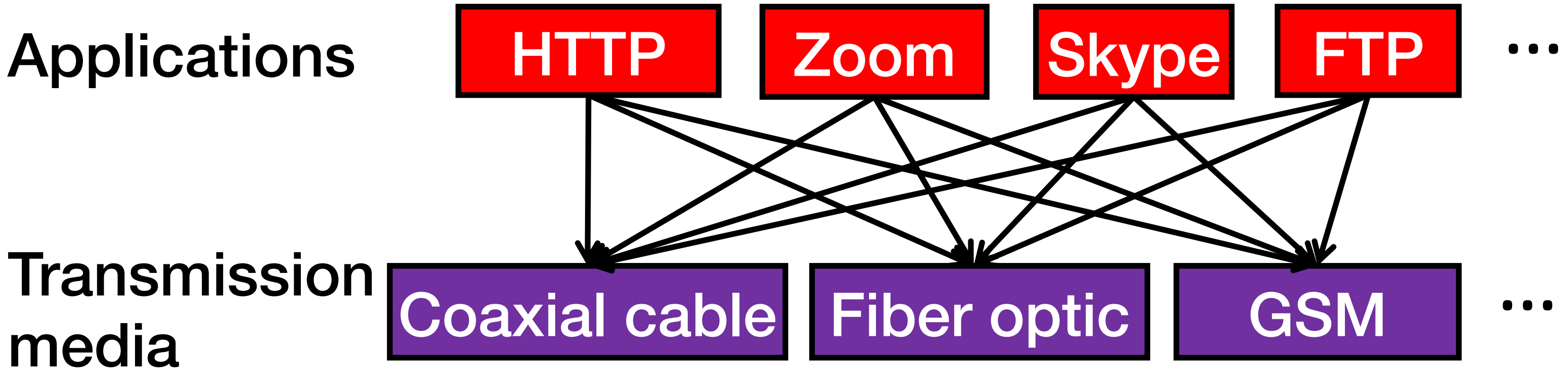
“Modularity based on abstraction is the way things get done”

2009 Turing Award Lecture

Modularity Through Layering

- Systems on systems on systems though layering
- Each layer hides complexity with abstraction
- Network layers today!

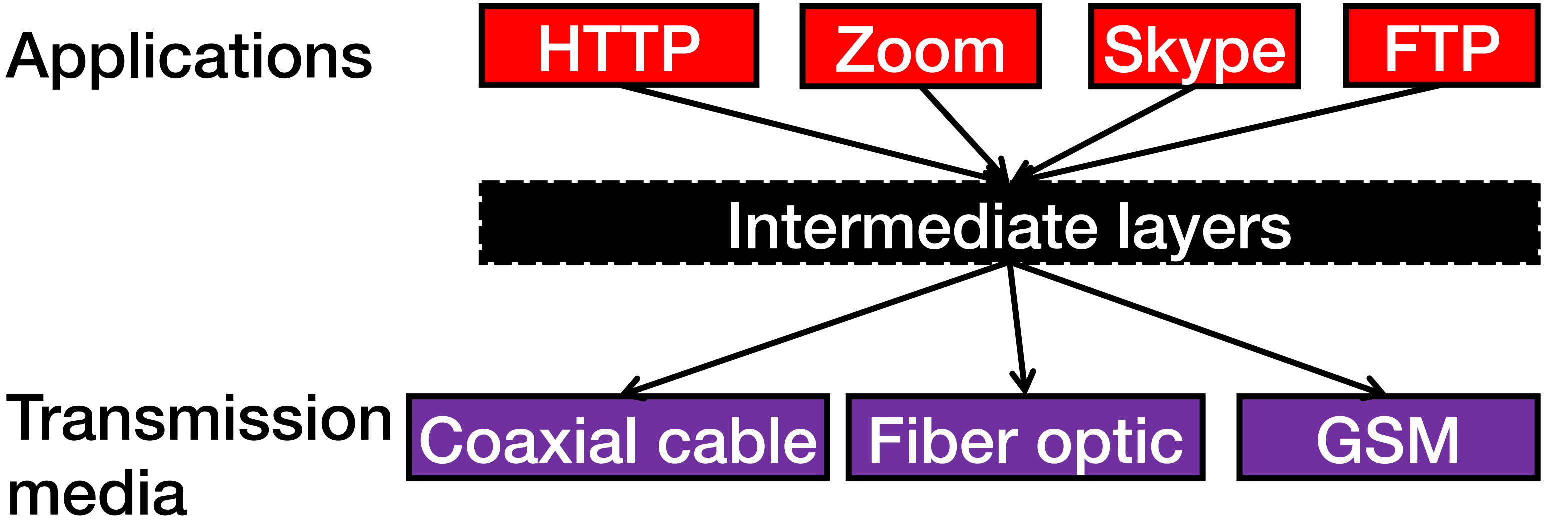
The Problem of Communication



- Re-implement every application for every new transmission medium?
- Change every application on any change to a transmission medium?

- No! But how does the Internet design avoid this?

Solution: Layering



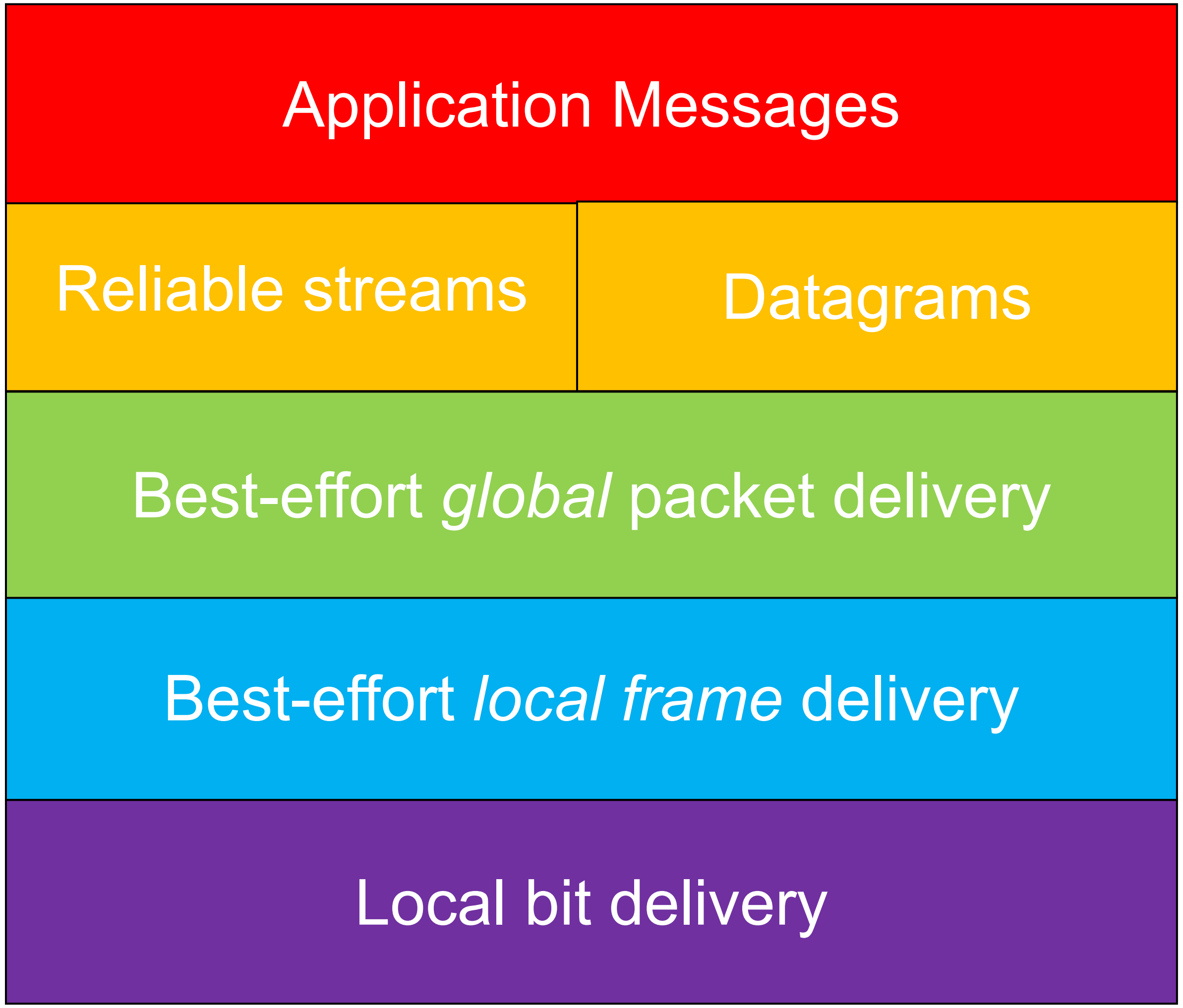
- Intermediate **layers** provide a set of abstractions for applications and media

- New applications or media need only implement for intermediate layer’s interface

The Art of Layering

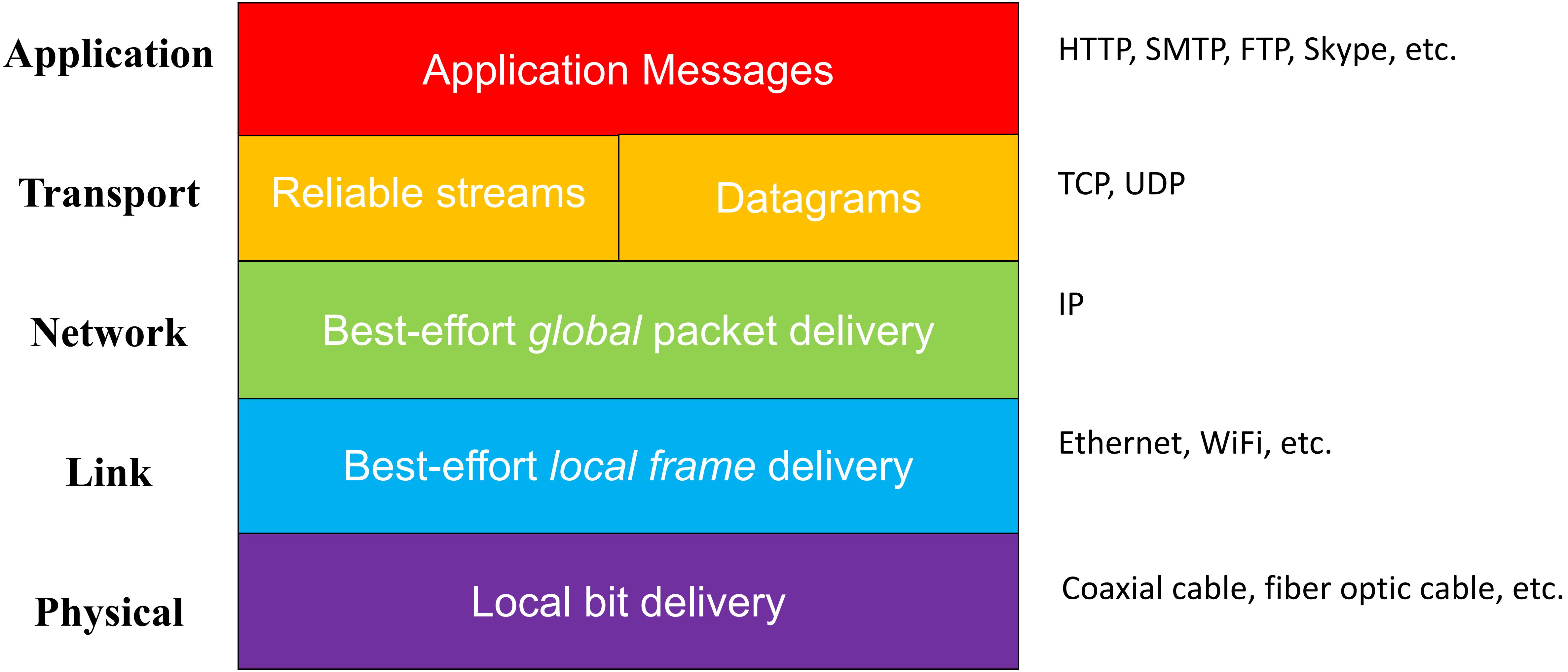
- How many layers?
- What goes in each layer?
- What abstraction (interface) does each layer provide?

Internet Protocol Layers



Application	Application Messages		HTTP, SMTP, FTP, Skype, etc.
Transport	Reliable streams	Datagrams	TCP, UDP
Network	Best-effort *global* packet delivery		IP
Link	Best-effort *local frame* delivery		
Physical	Local bit delivery		

Internet Protocol Layers



Application

Application Messages

HTTP, SMTP, FTP, Skype, etc.

Transport

Reliable streams

Datagrams

TCP, UDP

Network

Best-effort *global* packet delivery

IP

Link

Best-effort *local frame* delivery

Ethernet, WiFi, etc.

Physical

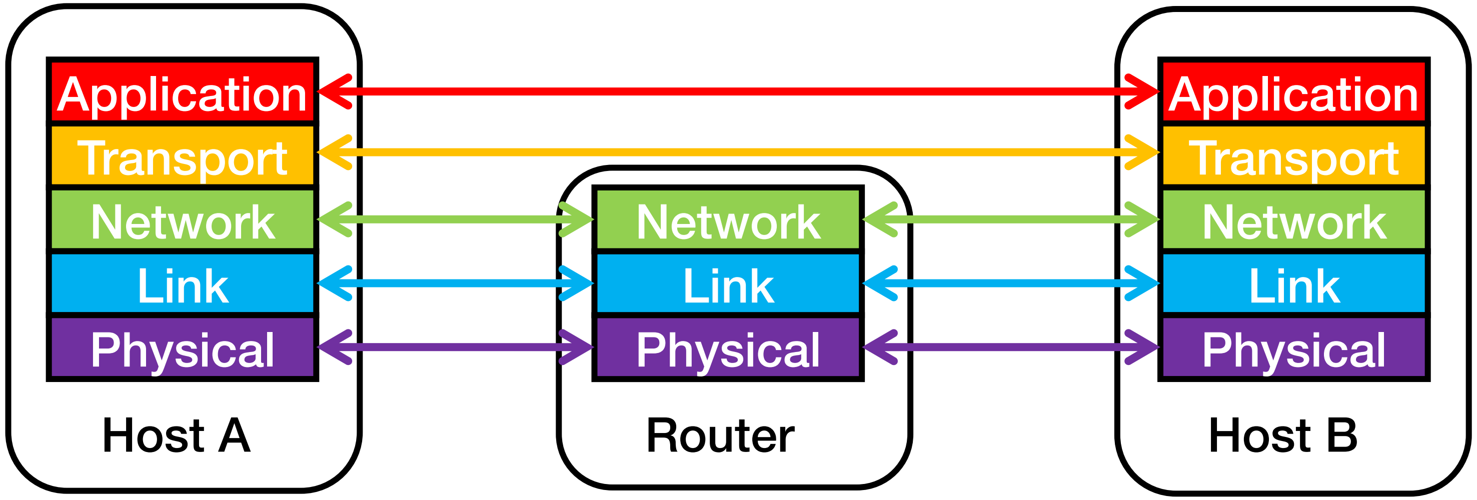
Local bit delivery

Coaxial cable, fiber optic cable, etc.

Logical Communication Between Layers

- How to forge agreement on meaning of bits exchanged between two hosts?

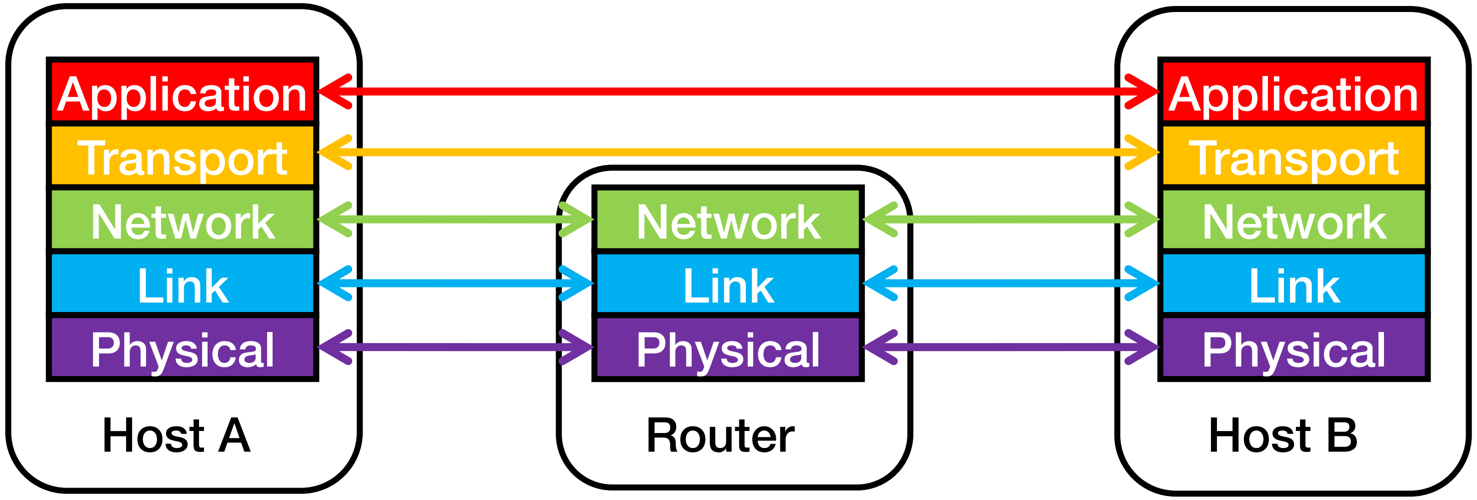
- **Protocol:** Rules that govern the format, contents, and meaning of messages
  - Each layer on a host interacts with peer host's corresponding layer via **protocol interface**



Logical Communication Between Layers

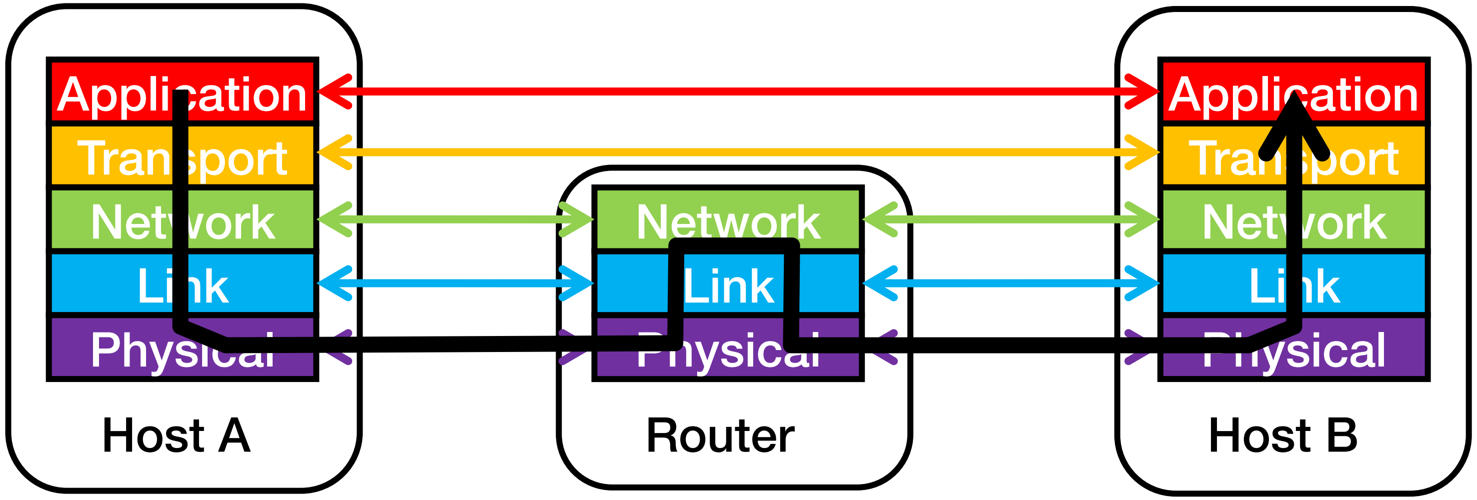
- How to forge agreement on meaning of bits exchanged between two hosts?

- **Protocol:** Rules that govern the format, contents, and meaning of messages
  - Each layer on a host interacts with peer host's corresponding layer via **protocol interface**



Physical communication

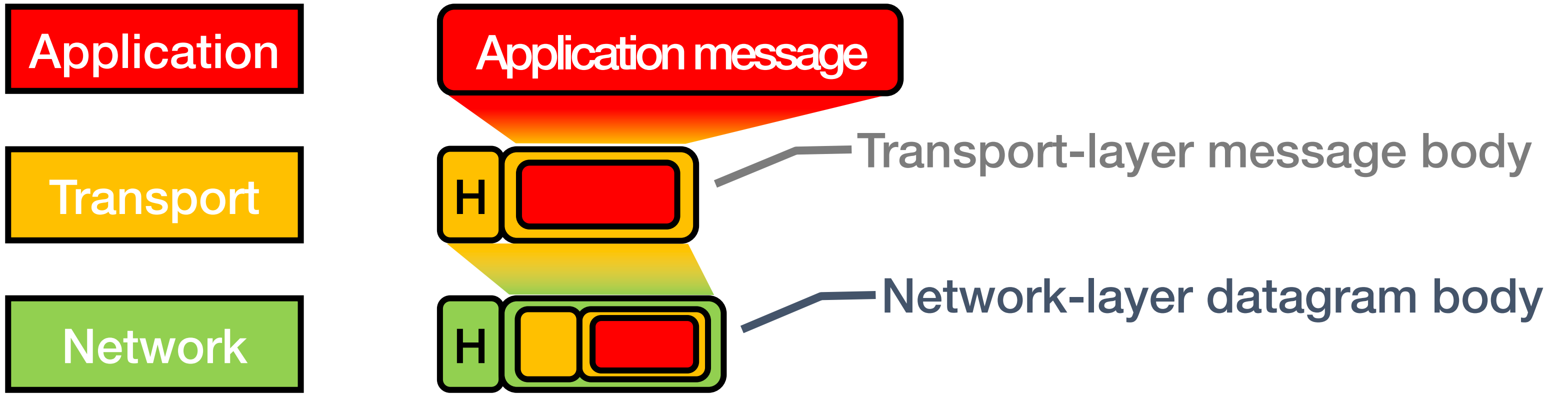
- Communication goes down to the **physical network**
- Then from **network** peer to peer
- Then up to the **relevant application**



Communication Between Peers

- How do peer protocols coordinate with each other?

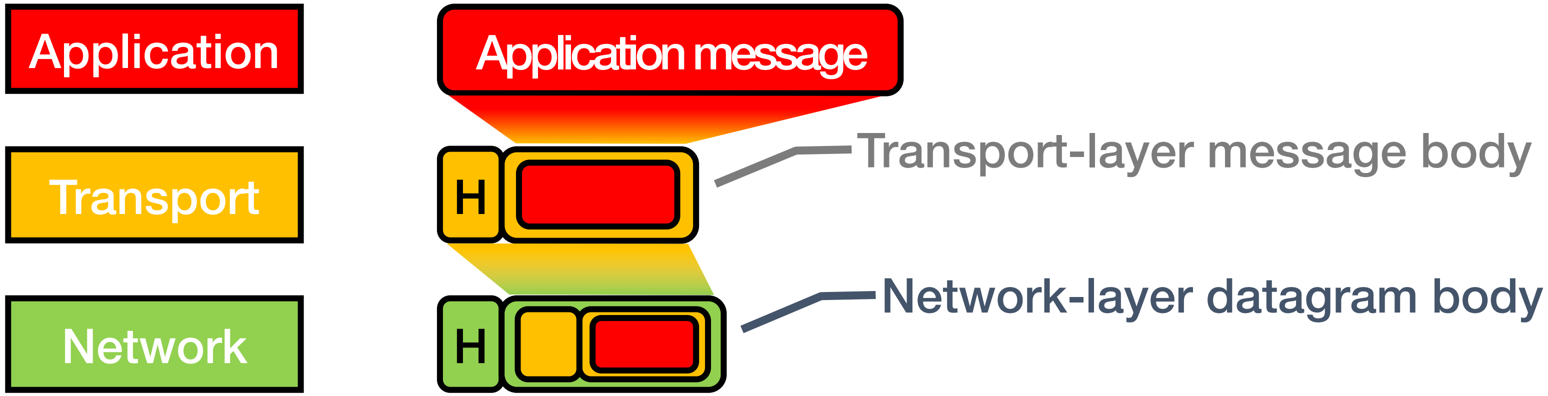
- Layer attaches its own header (H) to communicate with peer
  - Higher layers' headers, data encapsulated inside message
  - Lower layers generally do not inspect higher layers' headers



Communication Between Peers

- How do peer protocols coordinate with each other?

- Layer attaches its own header (H) to communicate with peer
    - Higher layers' headers, data encapsulated inside message
    - Lower layers generally do not inspect higher layers' headers



IP is the “Narrow Waist” of the Internet

- **The** network-layer protocol
  - Enables portability above and below
- Lots of link layer protocols underneath
- Several transport protocols on top
  - TCP, UDP, QUIC

IP: **Best-Effort** Global Packet Delivery

- Never having to say you’re sorry
  - Don’t have to reserve bandwidth and memory
  - Don’t have to do error detection and correction
  - Don’t have to remember anything across packets

- Easier to survive failures
  - Transient disruptions are okay during failure recovery

IP: **Best-Effort** Global Packet Delivery

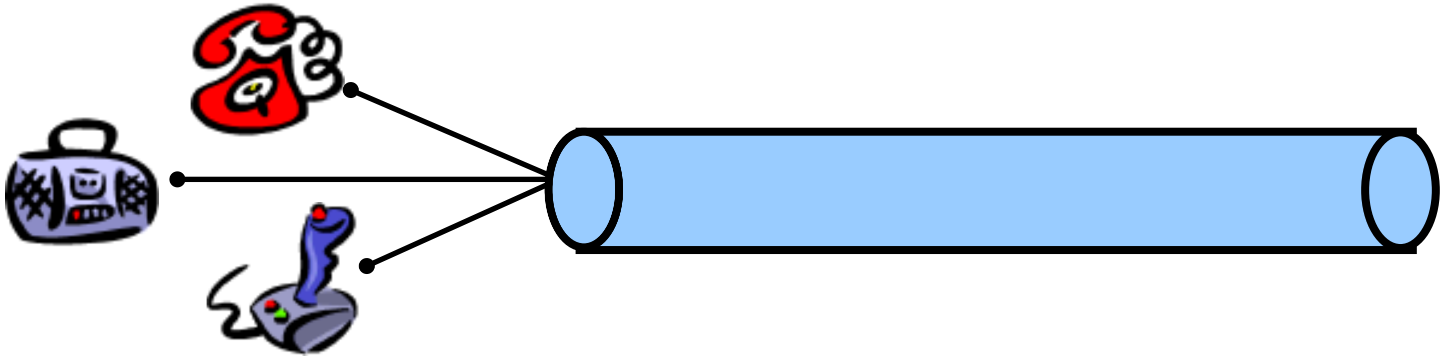
- Never having to say you’re sorry
  - Don’t have to reserve bandwidth and memory
  - Don’t have to do error detection and correction
  - Don’t have to remember anything across packets

- Easier to survive failures
  - Transient disruptions are okay during failure recovery

- Can run on nearly any link technology
  - Greater interoperability and evolution
  - RFC 1149: IP Datagrams Over Avian Carriers

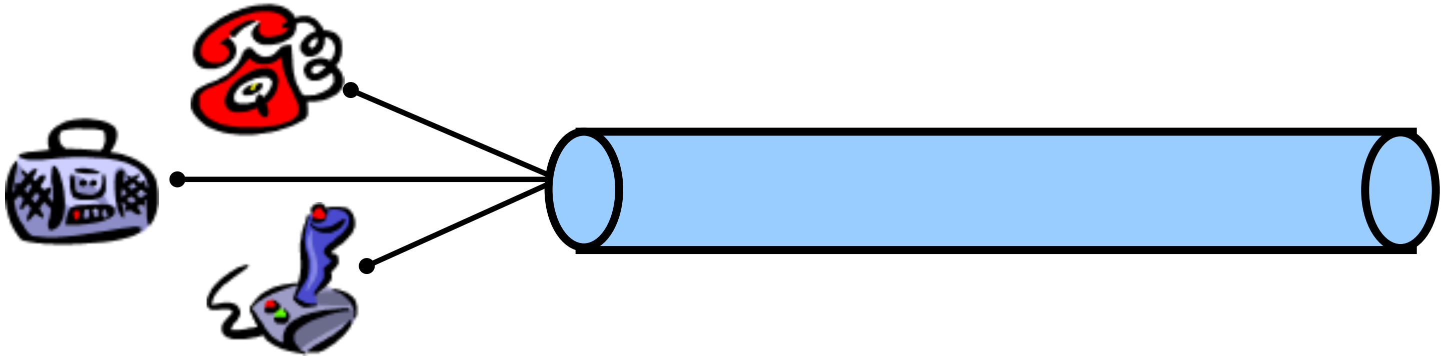
IP: Statistical Multiplexing

- Data traffic is bursty
  - Logging in to remote machines
  - Exchanging e-mail messages
- Don’t waste bandwidth
  - No traffic exchanged during idle periods
- Better to allow multiplexing
  - Different transfers share access to same links



IP: Statistical Multiplexing

- Data traffic is bursty
  - Logging in to remote machines
  - Exchanging e-mail messages
- Don’t waste bandwidth
  - No traffic exchanged during idle periods
- Better to allow multiplexing
  - Different transfers share access to same links



Transport: Application to Application

- Network layer is host-to-host

- Transport layer is port-on-host-to-port-on-host
  - think application to application
  - demultiplexing
  - e.g., port 80 is HTTP, port 443 is HTTPS, port 22 is SSH

- *Why transport and not network layer?*

Transport: Application to Application

- Network doesn’t have error detection

- Transport layer does have error detection

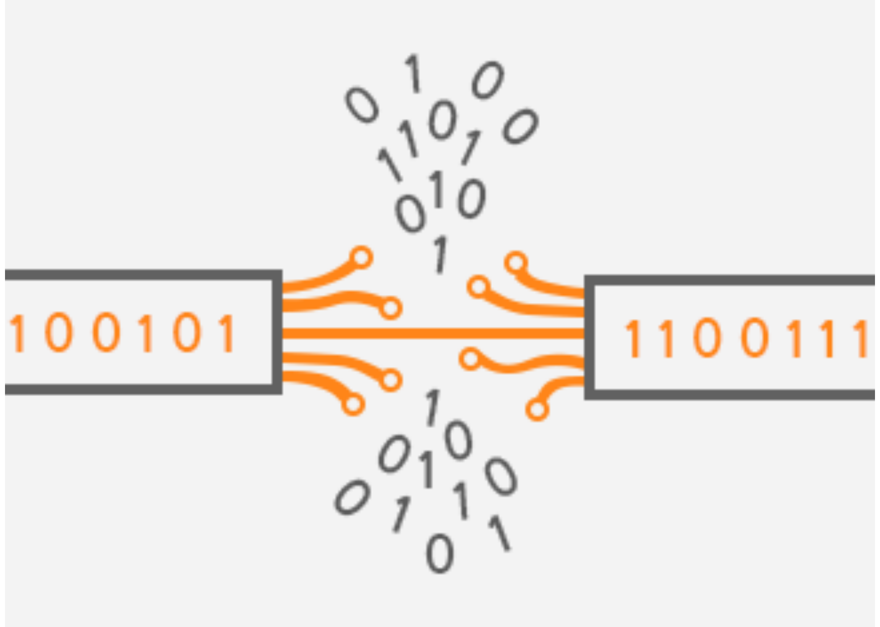
- *Why transport and not network layer?*

- *Why not both?*

Transport: Transmission Control Protocol (TCP)

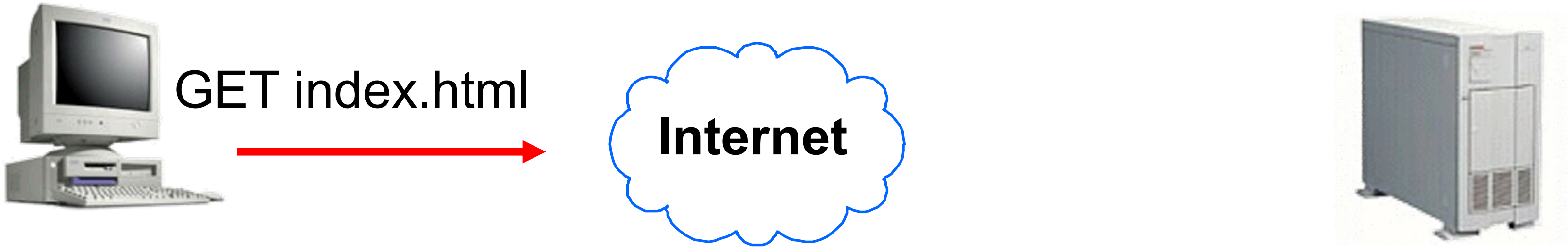
- Ordered, reliable stream of bytes
  - Built on top of best-effort packet delivery at the network layer

- Challenges with IP
  - Lost or delayed packets
  - Corrupted packets
  - Out-of-order packet arrivals
  - Receiver runs out of space
  - Network cannot handle current load

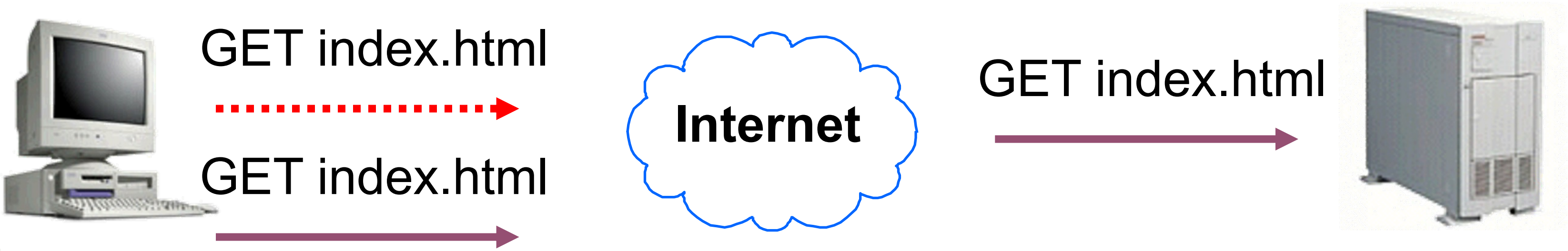


TCP: Lost or Delayed Packets

Problem: Lost or Delayed Data

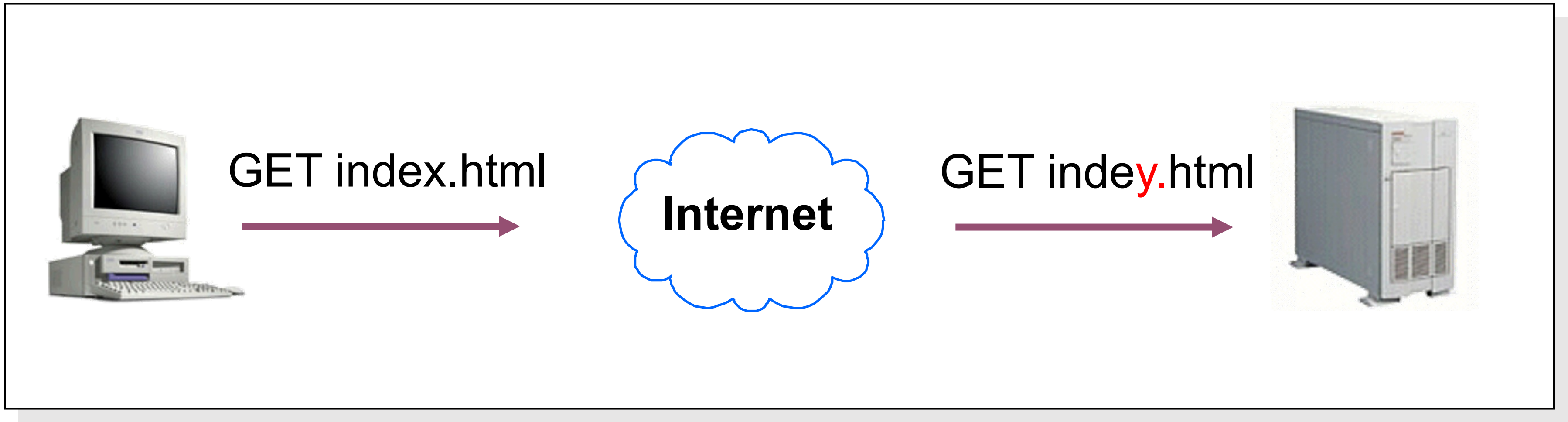


Solution: Timeout and Retransmit



Waiting for an acknowledgment…

TCP: Corrupted Data



GET index.html

Internet

GET indey.html

- Sender computes a checksum
  - Sender sums up all bytes in the payload
  - And sends the sum to the receiver

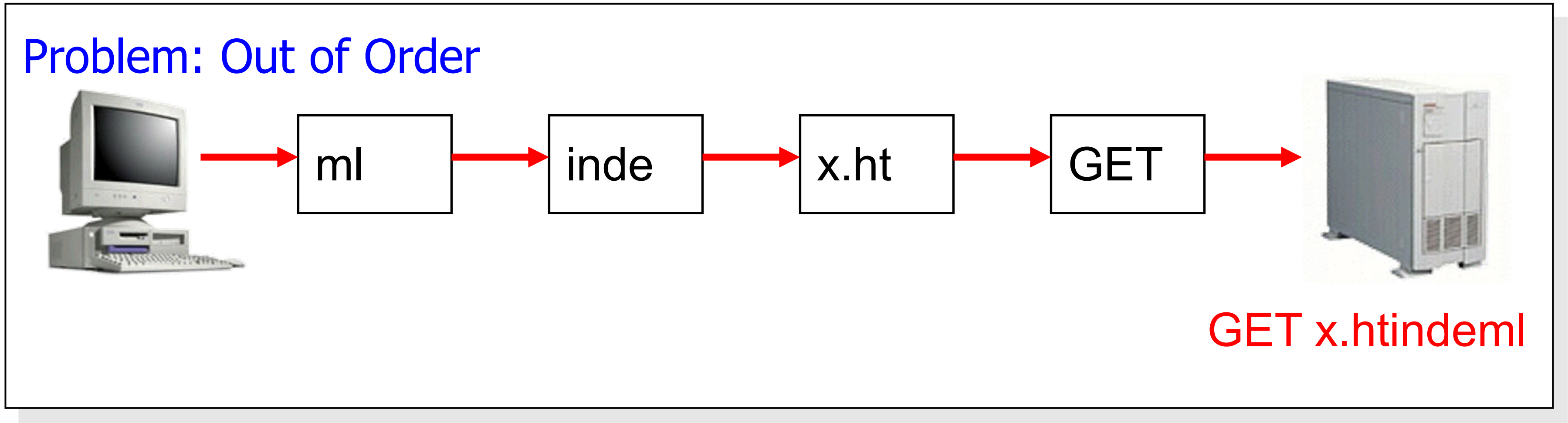
```
  134
+ 212
= 346
```

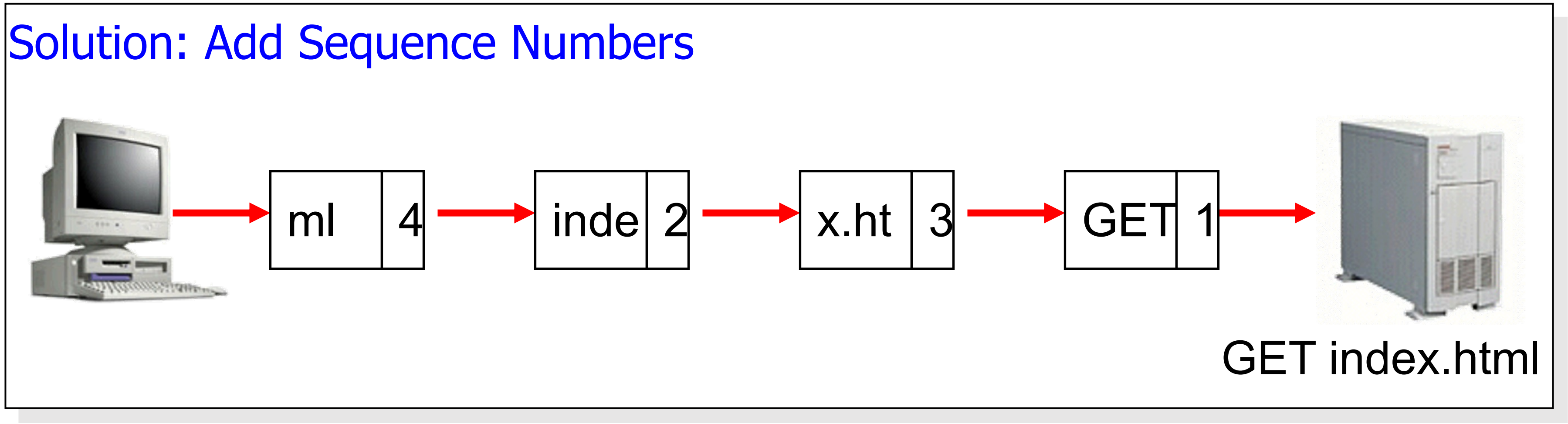
- Receiver checks a checksum
  - Recevier sums up all bytes in the payload
  - And compares against the checksum

```
  134
+ 216
= 350
```

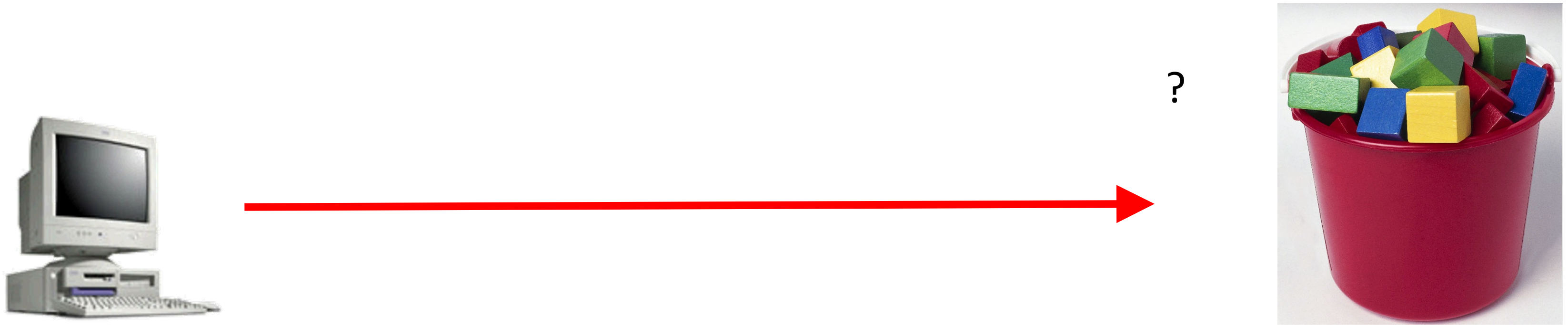
Then what?

TCP: Out-of-Order Packet Arrivals





TCP: Receiver that Runs Out of Space



- Receiver maintains a *window size*
  - Amount of data it can buffer
- Advertises window to the sender
  - Amount sender can send without acknowledgment
- Ensures that sender does not send too much
  - While still sending as much as possible

Flow control!

TCP: Receiver that Runs Out of Space

- Receiver maintains a *window size*
  - Amount of data it can buffer
- Advertises window to the sender
  - Amount sender can send without acknowledgment
- Ensures that sender does not send too much
  - While still sending as much as possible

Flow control!

TCP: Network that Cannot Handle the Load

- Problem: Too many packets at once
- Solution: Congestion control
  - Future lecture!

Transport: User Datagram Protocol (UDP)

- Datagram of bytes
  - A message

UDP does less than TCP,
why do we want UDP too?

- Challenges with IP
  - Lost or delayed packets X
  - Corrupted packets √
  - Out-of-order packet arrivals X
  - Receiver runs out of space X
  - Network cannot handle current load X

Layering & Network Layers Conclusion

- The art of layering
- Network layers
    - Protocol, headers, encapsulation
- IP layer: best-effort global packet delivery between host
- TCP layer: ordered, reliable byte stream between applications