# CPSC 416 Distributed Systems

Winter 2022 Term 2 (January 12, 2023)

**Tony Mason (fsgeek@cs.ubc.ca), Lecturer**

# Class Logistics

*If you are on the waitlist and are added to the class 48 hours or less before the Lab 1 deadline, you may request an extension of time to submit and it will be granted as a matter of policy. Project 1 is due Monday January 16, 2023 @ 11:59 PM PT*

*TAs will be posting office hours.*

*I will have informal office hours today from 4-5 pm PT on Discord*.

*Note that I'll continue to post the slides and links to the video (assuming it works right) on my class website (*Lectures – A File Systems Geek (fsgeek.ca)) I will try to post copies of lecture slides in advance. I reserve the right to update them!

# Fair Warning

These slides **are not** intended to be a *textbook*.

The words are mnemonic triggers about **what I will talk about**.

The slides **are not** a replacement for the lectures.

# Today's Failure

October 4, 2021

[Facebook Outage]

> All of this happened very fast. And as our engineers worked to figure out what was happening and why, they faced two large obstacles: first, it was not possible to access our data centers through our normal means because their networks were down, and second, the total loss of DNS broke many of the internal tools we'd normally use to investigate and resolve outages like this.

*Root cause:* Incorrect command was issued

*Secondary cause:* Audit tool that should have blocked the incorrect command was flawed.

*Result:* Facebook disconnected from the Internet

> Our primary and out-of-band network access was down, so we sent engineers onsite to the data centers to have them debug the issue and restart the systems. But this took time, because these facilities are designed with high levels of physical and system security in mind. They're hard to get into, and once you're inside, the hardware and routers are designed to be difficult to modify even when you have physical access to them. So it took extra time to activate the secure access protocols needed to get people onsite and able to work on the servers. Only then could we confirm the issue and bring our backbone back online.

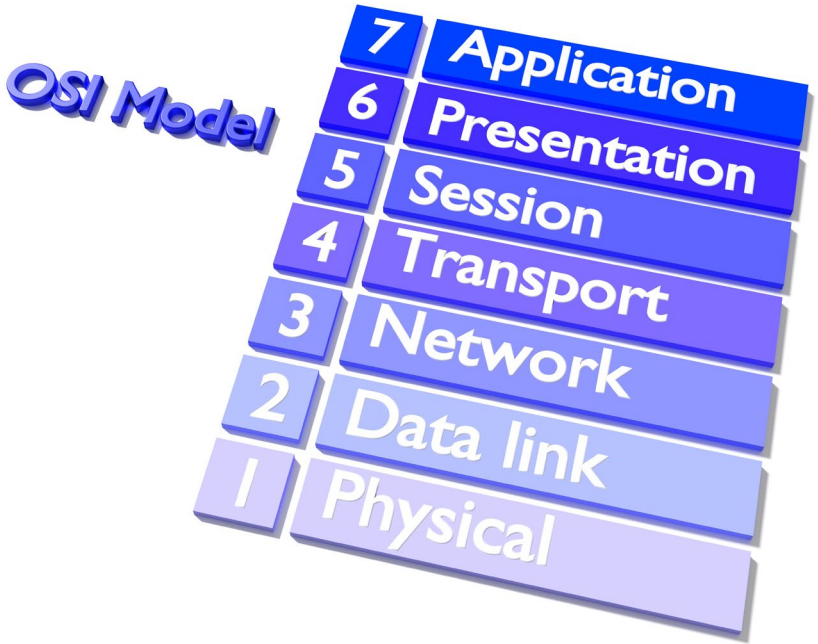
# Networks

Focus: Communications

Limited services @ each layer

Message Agnostic

End-to-end idea



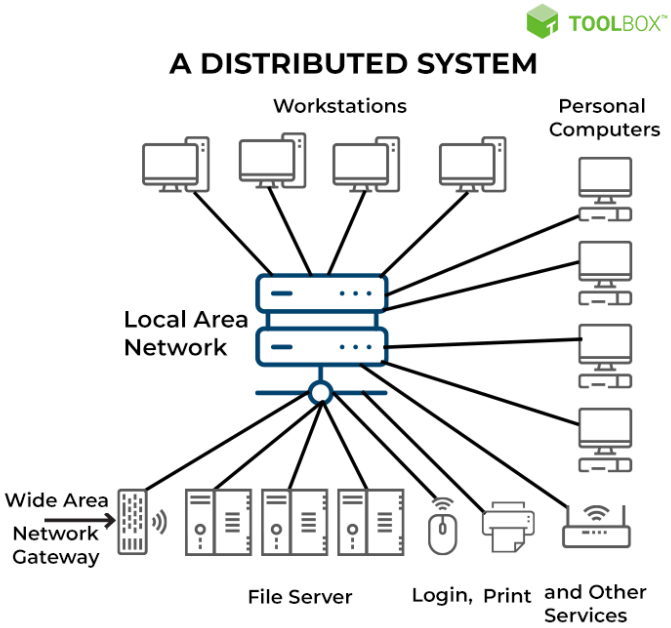
# Networks ≠ Distributed Systems

Network Aspects of Distributed Systems:

- Session Layer (and higher)
- Application-level interfaces

Additionally:

- Long-lived
- Service Interfaces
- Guarantees



# Common Techniques

Modularity, Layering, and Decomposition:

- Simplify systems building efforts
- Encapsulate complexity
- Implementation ≠ Interface
- Deferred binding
- Isolation *v.* sharing

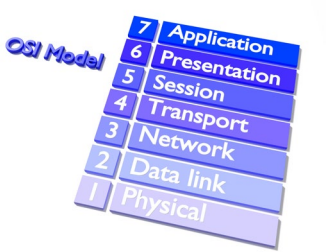
Define:

- Models
- Constraints
- Assumptions

# Networks: Review

Physical Layer:

- Moving data between endpoints
- Examples:
  - Coax
  - Copper
  - Fiber
  - EM signals



# Networks: Review

Physical Layer Characteristics

- Latency
- Jitter (latency variation)
- Bandwidth (capacity)
- Error rates

Errors:

- Collisions (shared media)
- Data errors
  - Neutrino storms!
  - Competing signals
  - Imperfections in physical components

# Networks: Review

Data Link

- Messages
- *What* kind of message
- *How big* of message
- *Data integrity* (checksum)


Examples:

- Host-Host Control Message Formats (RFC 22)
- Ethernet
- Avian Carriers (RFC 1149)
- Asynchronous Transfer Mode (ATM)


Network Protocol

- Connects distinct networks together
- Support *relay*
- Permits *routing*
- Supports one-to-many delivery
  - Broadcast
  - Multicast

# Networks: Review

Transport Layer

- End-to-end network delivery
- Slice and dice (segmentation)
- Gluing together (reassembly)
- Defined *ordering*
- *May* include connection state (not required)


Examples:

- Network Control Protocol (NCP, RFC 60)
- Transmission Control Protocol (TCP, RFC 793, RFC 9293)
- User Datagram Protocol (UDP, RFC 768)

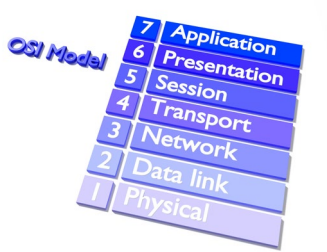
August 2022!

# Networks: Review

Session: Connection state/information

Presentation: Data pack/unpack

Application: Everything *else* (Web browser, FTP, NFS, etc.)



# Internet

Two or more connected networks

Challenges: Routing, Billing, Security, Performance

# Internet Challenges

Heterogeneity/Interoperability

Locating Resources

Routing

Reliability

Guarantees

# Heterogeneity/Interoperability

Addressing/Routing

Performance (Bandwidth/Latency/Jitter)

Packet Size

Data loss

Dissimilar network technology

Maintaining delivery order

# Naming

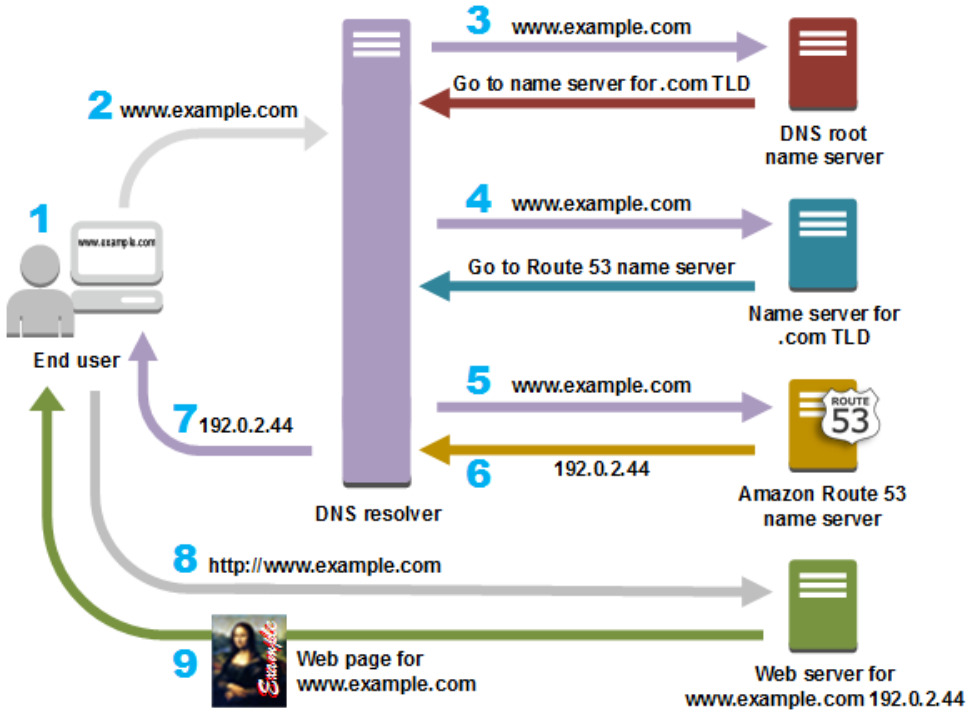
IP

- IPv4 – 4 byte
- IPv6 – 16 byte

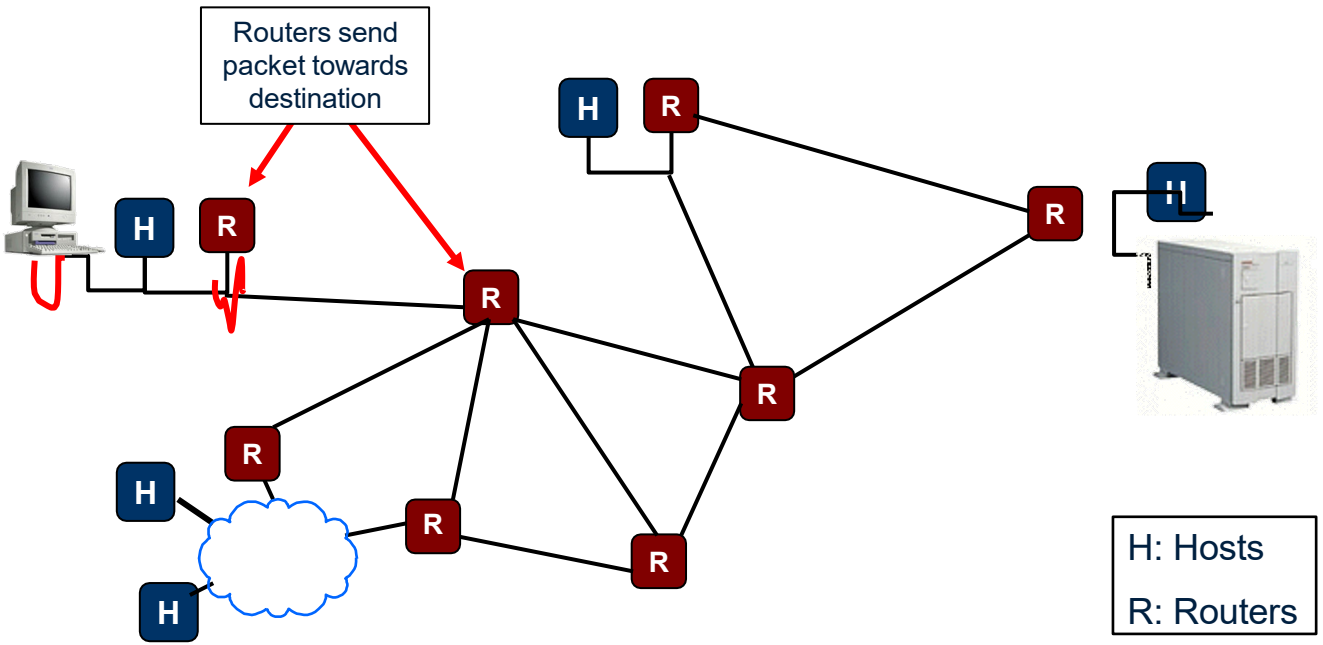
TCP/UDP: 2 byte “ports”

DCE/RPC: UUIDs (16 bytes)

Logical to Physical Naming



# Routing


Routers send packet towards destination
H
R
H
R
R
H
R
R
R
H
R
R
H
H
R
H: Hosts
R: Routers

# Network Service Model

Ethernet/Internet best-effort

- Packets lost/damaged
- Out-of-order delivery

Enhanced Services

- Quality of Service
- Reliability
- Detect corruption
- Ordered delivery
- Fairness (flow/congestion control)

# Failure models

Fail-stop

- Something bad? STOP

Fail-slow

- Taking too long
- Difficult to detect (Fail-slow at Scale) but real

Byzantine

- Undetected errors
- Bad actors

# TCP

TCP provides:

- Reliability
  - Timeouts
  - Retries
  - Checksums
- Flow/Congestion Control
- Segmentation/Reassembly
- Reordered packets (sequence numbers)

Disadvantage?  Latency sensitive, slow, complex

# Network Functionality

Link

Multiplexing

Routing

Addressing/naming (locating peers)

Reliability

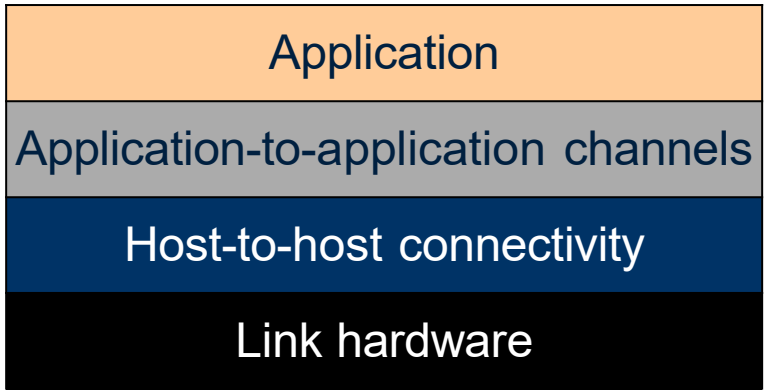
Flow control

Fragmentation

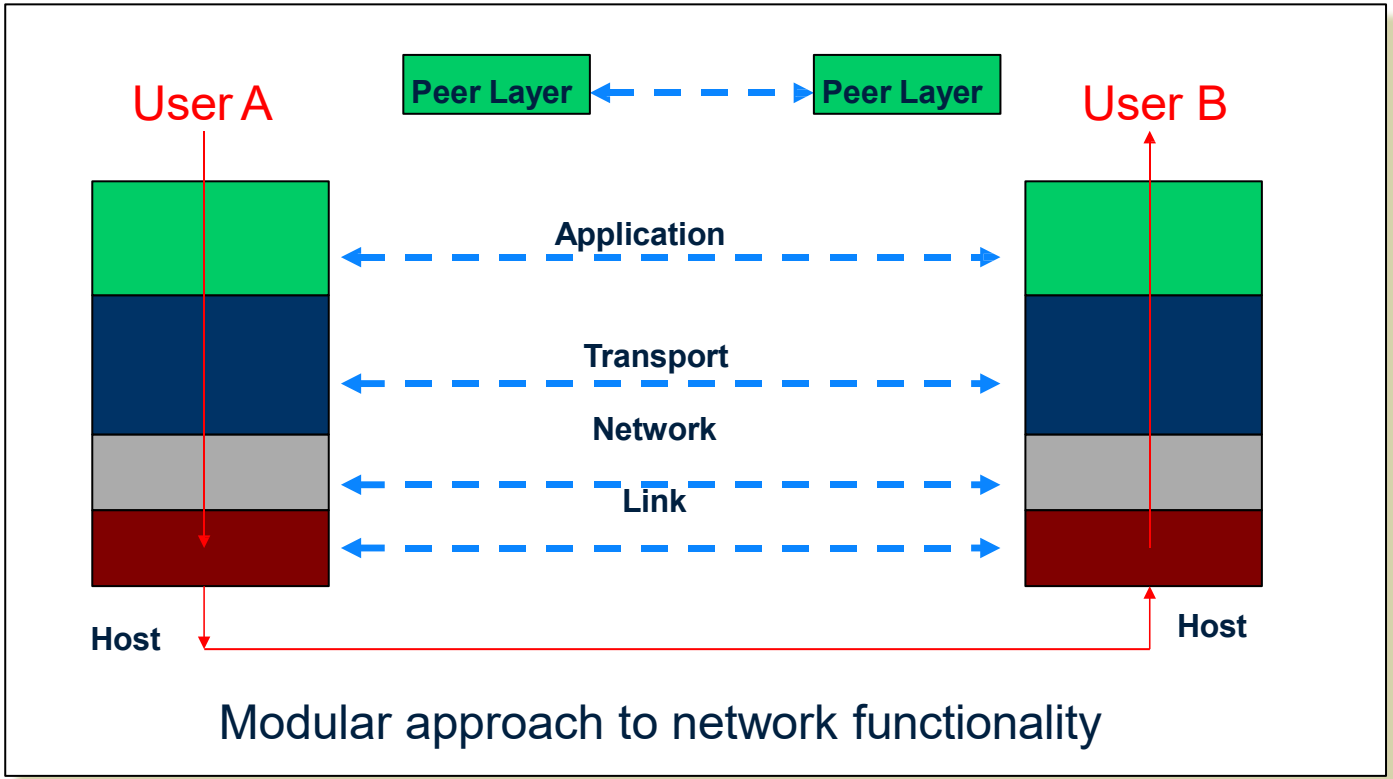
Etc….

# Layering

Modular approach to network functionality



# Layering


User A
Peer Layer
Peer Layer
User B
Application
Transport
Network
Link
Host
Host
Modular approach to network functionality

# Layering

Each layer

- Relies on lower level services
- Exports services to upper levels

Interface

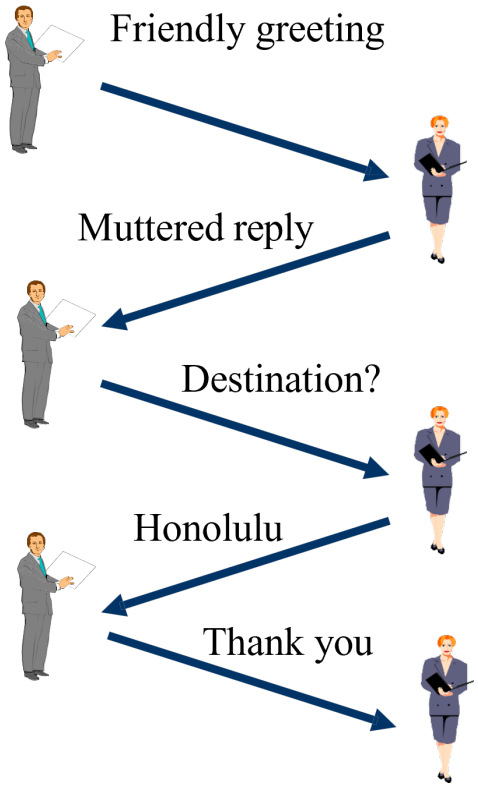
- Defines interactions
- Abstracts away implementation

# Protocols

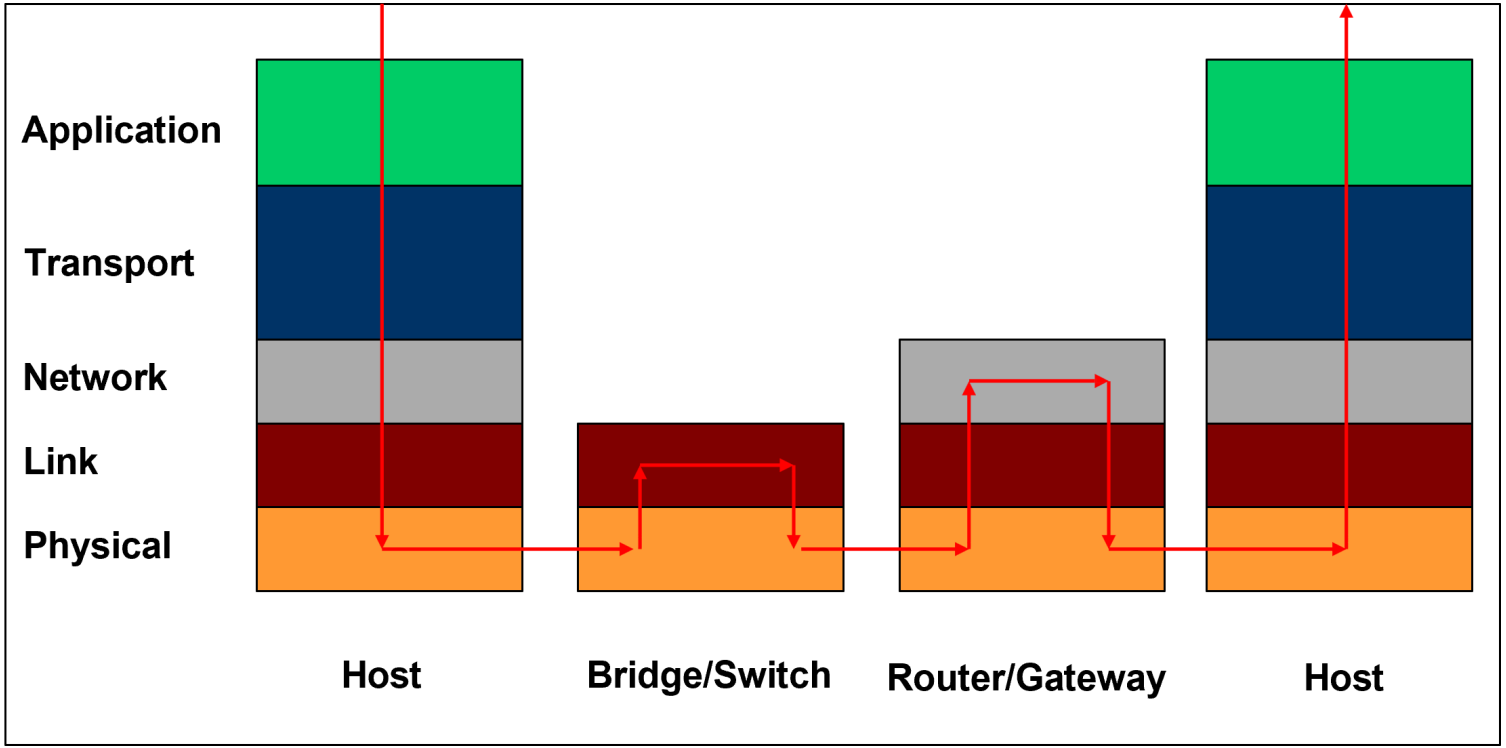
Essential for communications

Define

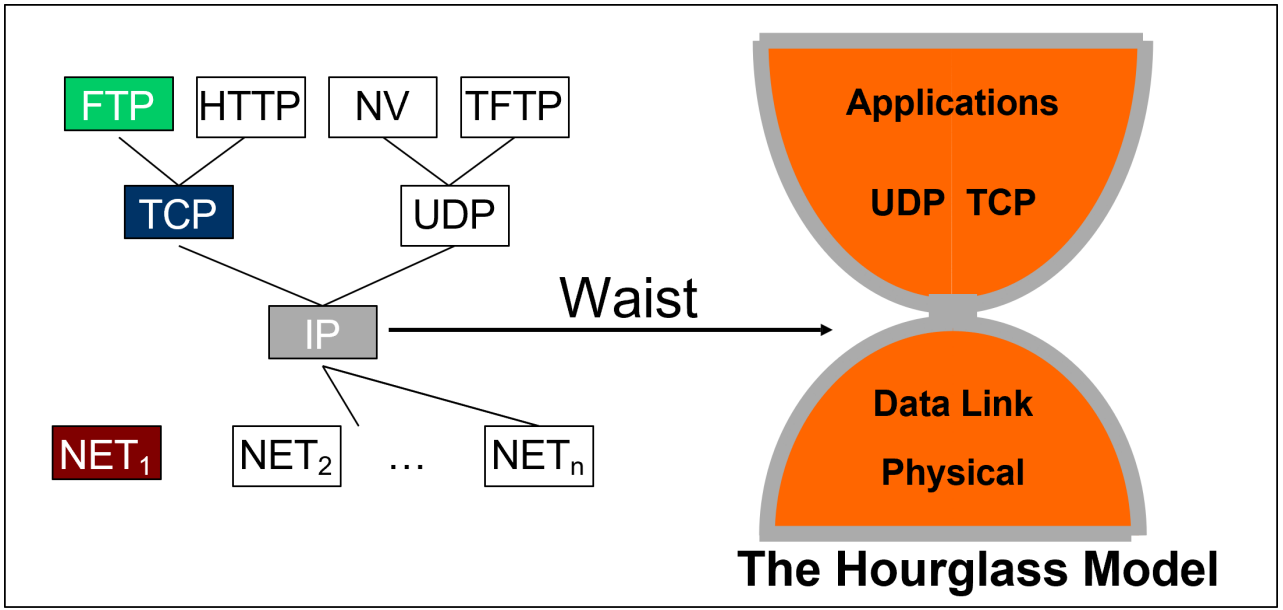
- Interface to higher layers (API)
- Interface to peer layers (syntax/semantics)
  - Initiation
  - Data format
  - Message ordering
  - Message to action mapping
  - Error handling
  - Termination



# IP Layering

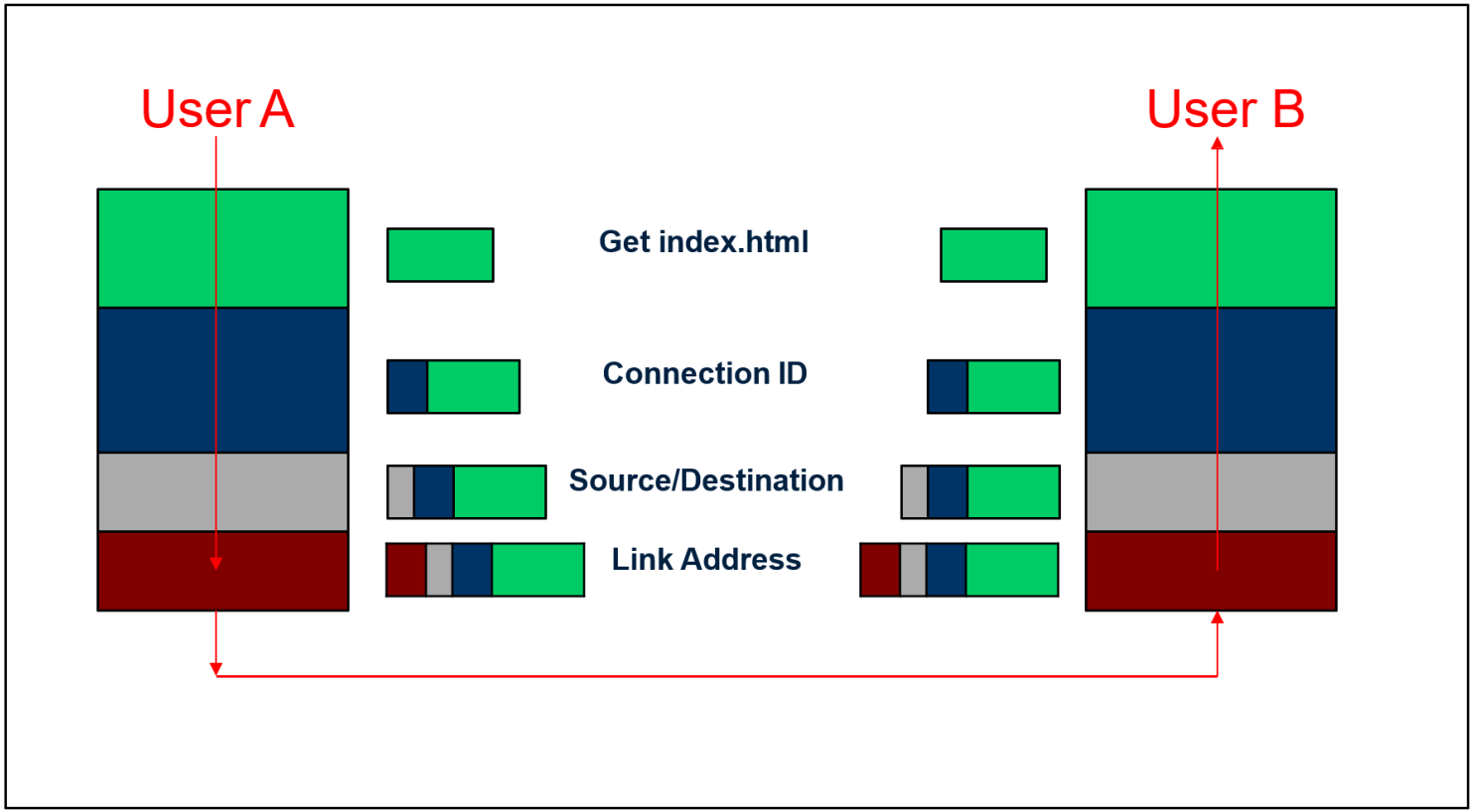

Application
Transport
Network
Link
Physical
Host
Bridge/Switch
Router/Gateway
Host

# Internet Protocol Suite (IP)




Facilitates Interoperability

# Layer Encapsulation



# Multiplexing and Demultiplexing

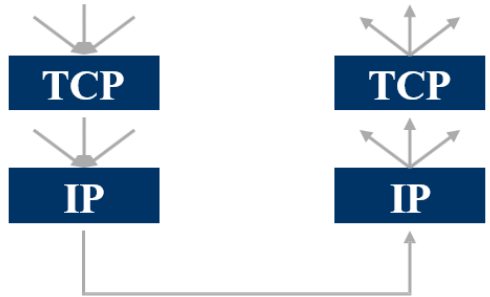
Multiple implementations @ each layer

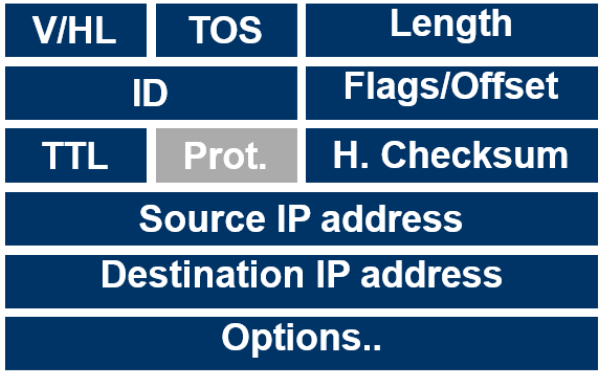
- Need to determine which version to use

Header includes layer ID field

- Set by sender
- Used by receiver

Each layer can multiplex



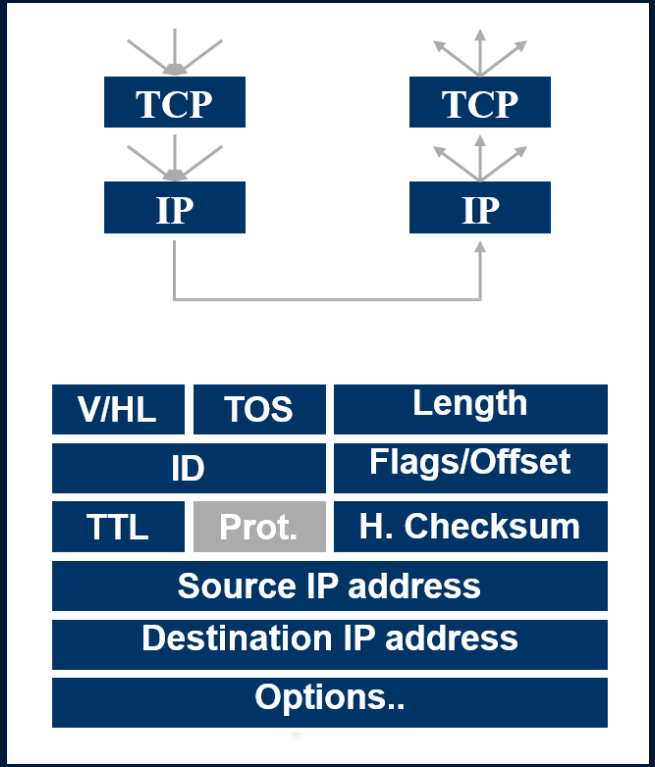
# Multiplexing and Demultiplexing

## List of IP protocol numbers

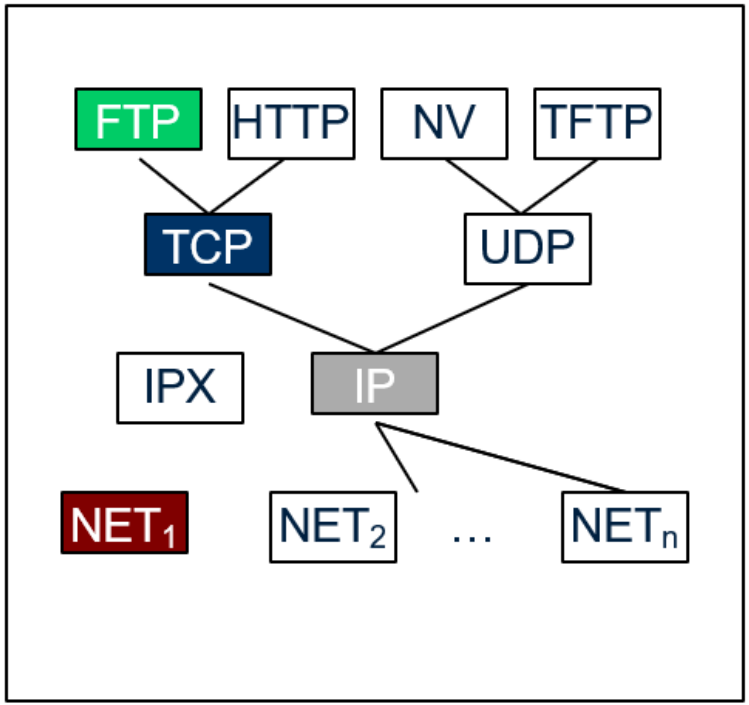
From Wikipedia, the free encyclopedia

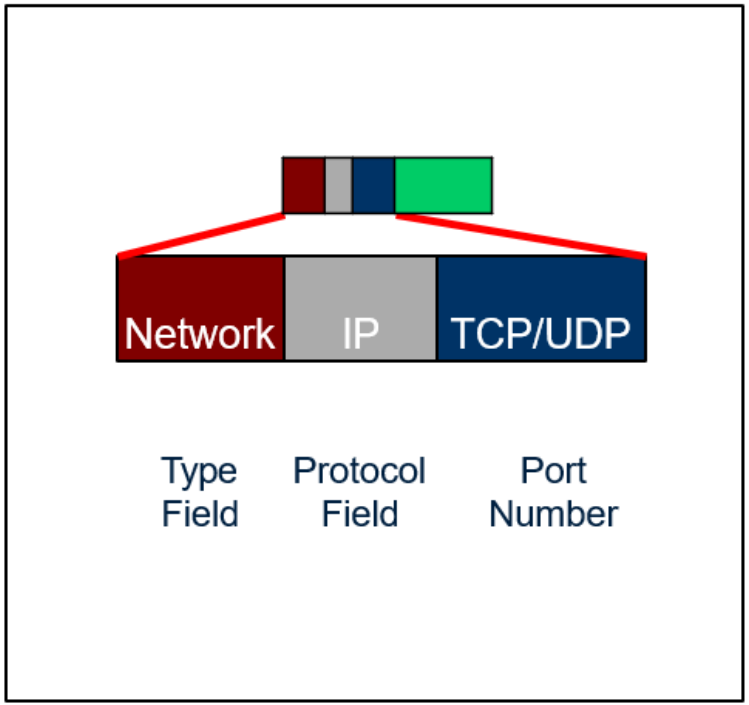
**This is a** list of IP numbers **used in the *Protocol* field of the** IPv4 hea

| Decimal | Hex | Keyword | |
|---|---|---|---|
| 0 | 0x00 | HOPOPT | IPv6 Hop-by-Hop Option |
| 1 | 0x01 | ICMP | Internet Control Message Prot |
| 2 | 0x02 | IGMP | Internet Group Management P |
| 3 | 0x03 | GGP | Gateway-to-Gateway Protocol |
| 4 | 0x04 | IP-in-IP | IP in IP (encapsulation) |
| 5 | 0x05 | ST | Internet Stream Protocol |
| 6 | 0x06 | TCP | Transmission Control Protocol |
| 7 | 0x07 | CBT | Core-based trees |
| 8 | 0x08 | EGP | Exterior Gateway Protocol |
| 9 | 0x09 | IGP | Interior Gateway Protocol (any their IGRP)) |
| 10 | 0x0A | BBN-RCC-MON | BBN RCC Monitoring |
| 11 | 0x0B | NVP-II | Network Voice Protocol |
| 12 | 0x0C | PUP | Xerox PUP |
| 13 | 0x0D | ARGUS | ARGUS |



# Protocol Demultiplexing



# Internet Design Goals

Connect Existing Networks

Survivability (failure *resistant*)

Heterogeneous services

Distributed Control (“cooperation”)

Easy Attachment

Cheap (“cost effective”)

Economic accountability

Recommended Reading:

- End-to-end arguments in system design (Saltzer, et. Al.)

# Survivability

Network disruption/reconfiguration

- Endpoints don’t care
- No “higher state” reconfiguration

How?

| | State in Network | State in Host |
|---|---|---|
| Failure handing | Replication | “Fate sharing” |
| Net Engineering | Tough | Simple |
| Routing state | Maintain state | Stateless |
| Host trust | Less | More |

# Survivability (2)

Network disruption/reconfiguration

- Endpoints don't care
- No "higher state" reconfiguration

How?

| | State in Network | State in Host |
|---|---|---|
| Failure handing | Replication | "Fate sharing" |
| Net Engineering | Tough | Simple |
| Routing state | Pkts on same path: complex | Pkts on indep. paths: simple |
| Host trust | Less | More |

# Fate Sharing


Lose state iff entity is lost

Examples:

- TCP state lost if a host crashes
- TCP state *not* lost if relay crashes

Trade-offs

- Network knowledge limited
- Trust Endpoints > Network

# Reality Check

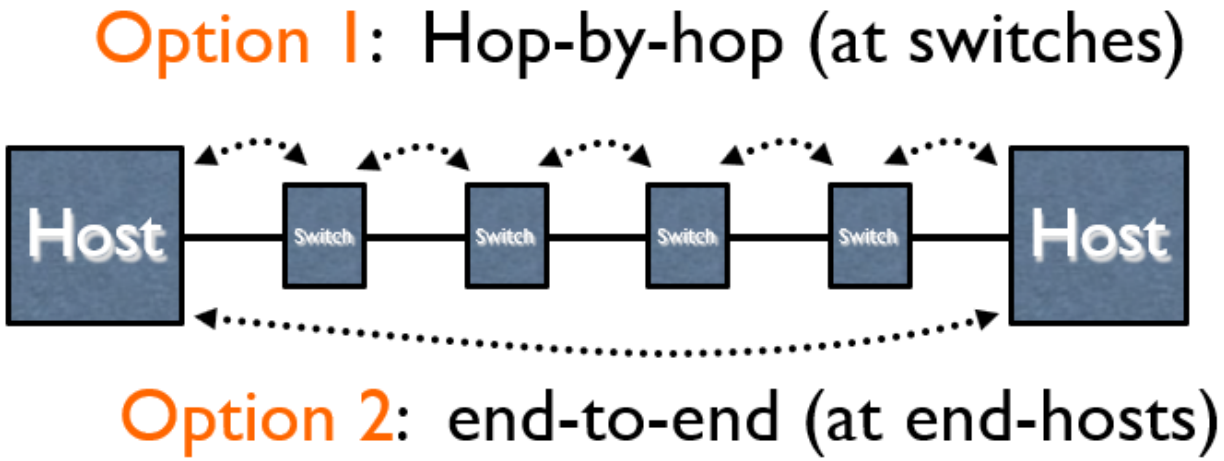
Real systems blend knowledge/control

- Link
- Multiplexing/bonding/balancing
- Routing
- Addressing/naming
- Reliabitily
- Security/Encryption
- Flow control
- Fragmentation

# Design Question

Where to implement reliability?

- In the network
- On the endpoints



# Reliability Options

Hop-by-hop: relay guarantees message delivery to next hop

End-to-end: Endpoint guarantees delivery

Question: what issues to consider?

# Hop-by-hop Option

How to guarantee delivery

- Relay reboots
- Relay *loses* traffic

Queuing theory bites you.

Think Circuit versus Packet switched

Ever used a wired phone?

# End-to-end Option

***Where*** to place functionality

- In network
- Network edges

Principle, not requirement

- If you have to implement a function end-to-end anyway (e.g., because it requires the knowledge and help of the end-point host or application), **don't implement it inside the communication system**
- Unless there's a compelling performance enhancement

# Case: send file over Internet

Break it into packets

- *Segmentation and reassembly*
- *Acknowledge receipt*
  - *Can do multiple at once*
  - *Optimize for the common case!*

Efficient (real-world) case

- Track portions of file received
- Acknowledge received portions *or*
- Request missing portions

FTP versus Bittorrent

# Service Types

Network layer: *datagram delivery* (“best effort”)

Simple network elements

Fast/efficient message transmission

Higher-level services add guarantees

Scalable/Flexible

No quality of service (QoS) required

- Physical networks guarantee delivery!
- Reality? QoS requires network support

# User Datagram Protocol Analogy


| UDP | Postal Mail |
| --- | --- |
| • Single socket to receive messages | • Single mailbox to receive letters |
| • No guarantee of delivery | • Unreliable |
| • Not necessarily in-order delivery | • Not necessarily in-order delivery |
| • Datagram – independent packets | • Letters sent independently |
| • Must address each packet | • Must address each letter |

# Transmission Control Protocol (TCP) Analogy

## TCP

- Reliable – guarantee delivery
- Byte stream – in-order delivery
- Connection-oriented – single socket per connection
- Setup connection followed by data transfer

## Telephone Call

- Guaranteed delivery
- In-order delivery
- Connection-oriented
- Setup connection followed by conversation

# Just use TCP?

TCP Guarantees > UDP Guarantees

*All magic comes with a price.*

Connection set-up: three messages, one round-trip

Lost packet: one extra round-trip

Delivery ordering: buffering, tracking

*Guarantee may be incorrect* (e.g., “what time is it?”)

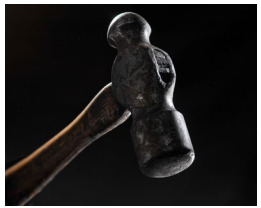
# Reality: Pick the *right* tool

Right tool *for the task*

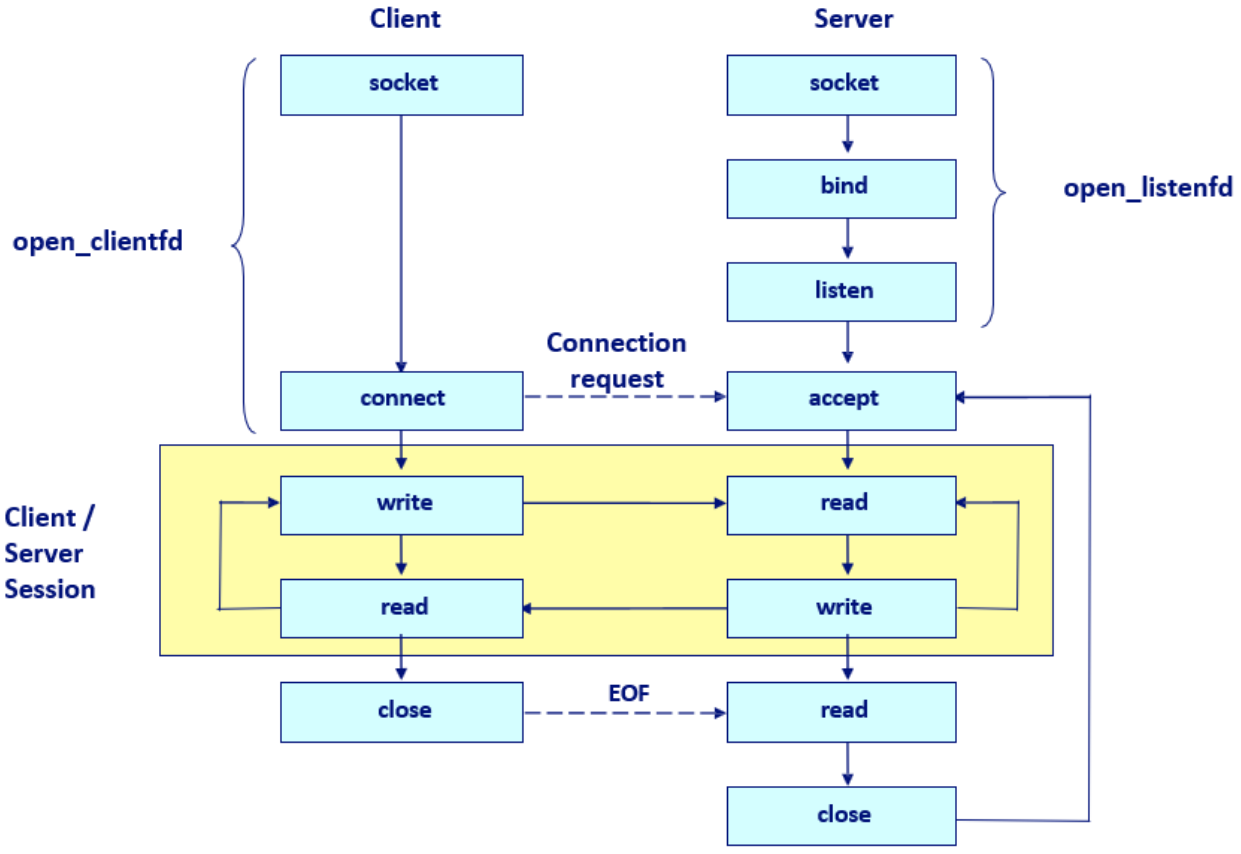
Clog + butter knife ≠ *ball peen hammer + chisel*

*TCP ≠ UDP*

*Look around:* **lots of options**



# Socket API Overview



# Blocking Sockets

What happens if an application write()s to a socket waaaaay faster than the network can send the data?

- TCP: controls send speed
- Fills kernel socket buffers
  - Once full: blocks send operation
- Blocking
  - Thread execution suspended
  - Thread resumes when space allows

# Datagrams

No blocking

No buffers/space?

- Drop on the floor
- Send *might* return an error (not likely)

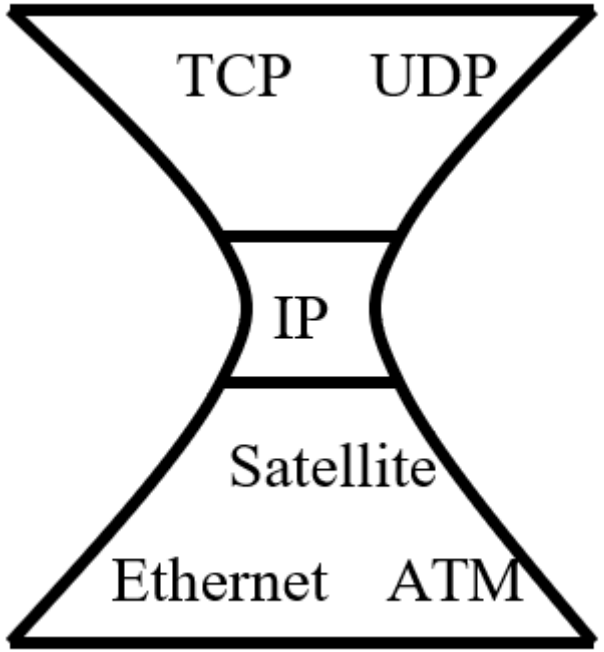
*Best effort*

# Internet Architecture Summary

Packet-switched datagram network

IP is the *most* common network protocol

No network level state, only end-to-end



# KISS: Keep it simple, stupid

Dumb network

- IP provides minimal functionalities
- *Pay no attention to Ipsec*
- *Ignore IP Multicast*

Smart endpoint

- Transport layer (TCP): {flow,error,congestion} control
- RPC (TCP or UDP): mixed services
  - Flow control
  - Authentication/security
  - Session management
  - Reliability

# Questions?


UBC
THE UNIVERSITY OF BRITISH COLUMBIA