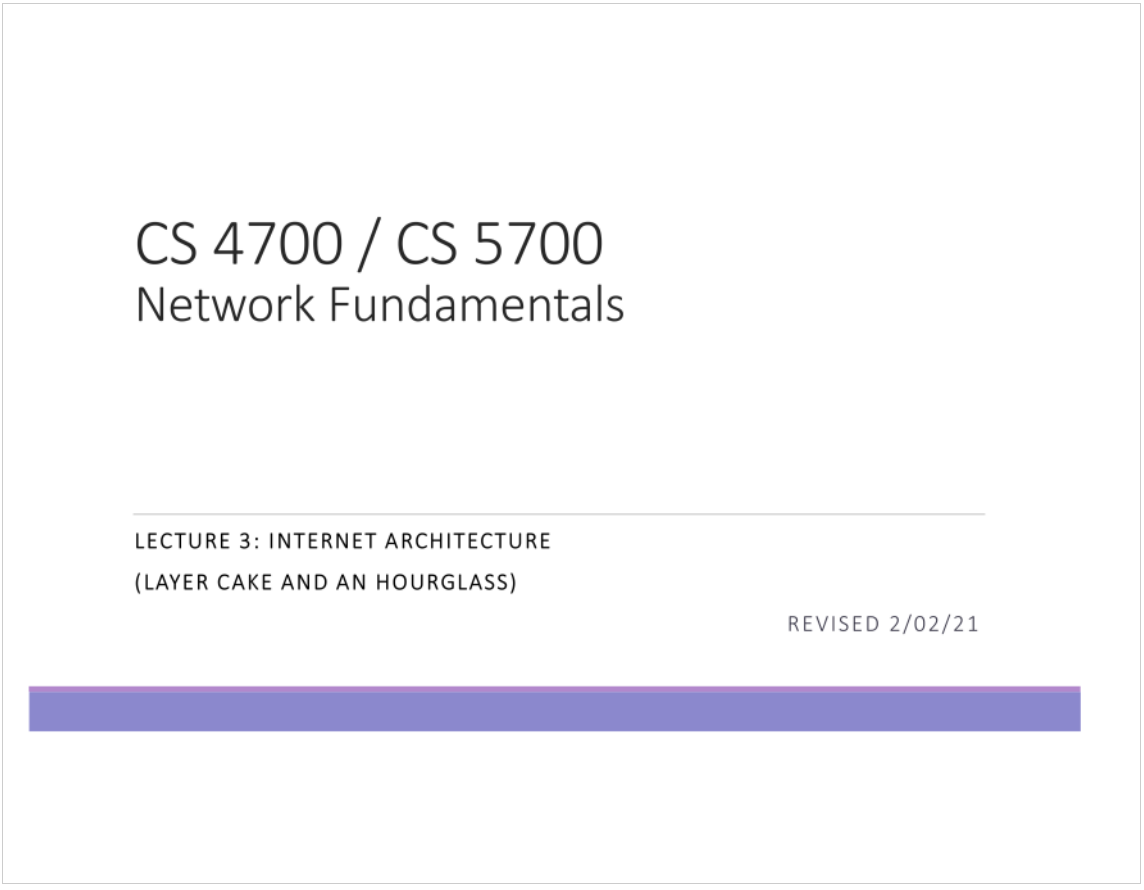
1/11/24 Lecture



Logistics

Project 1 posted, available
- Sorry for the hiccups

Office hours for me and TAs
- Will be worked out this week

HotCRP not up yet.
- Links to the Papers are on the Schedule on the course website
- I have not assigned a paper to review, but I expect you to have read the papers by the deadline

We may have an In-class exercise this week

Organizing Network Functionality

Networks are built from many components
- Networking technologies
  - Ethernet, Wifi, Bluetooth, Fiber Optic, Cable Modem, DSL
- Network styles
  - Circuit switch, packet switch
  - Wired, Wireless, Optical, Satellite
- Applications
  - Email, Web (HTTP), FTP, BitTorrent, VoIP

How do we make all this stuff work together?!

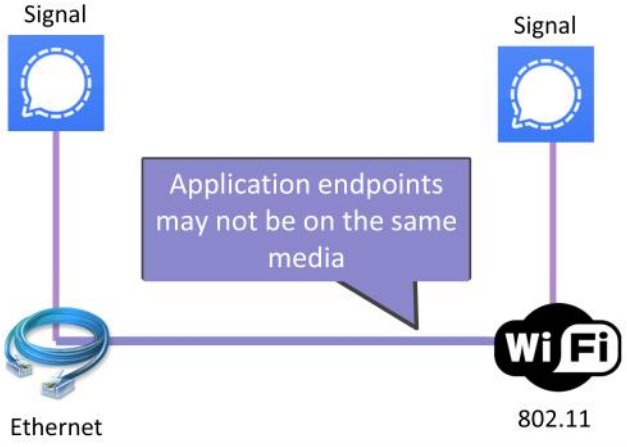
Problem Scenario

Chromium service needs to write code for each form of the different networking services of Ethernet, Wifi, Bluetooth and Cellular

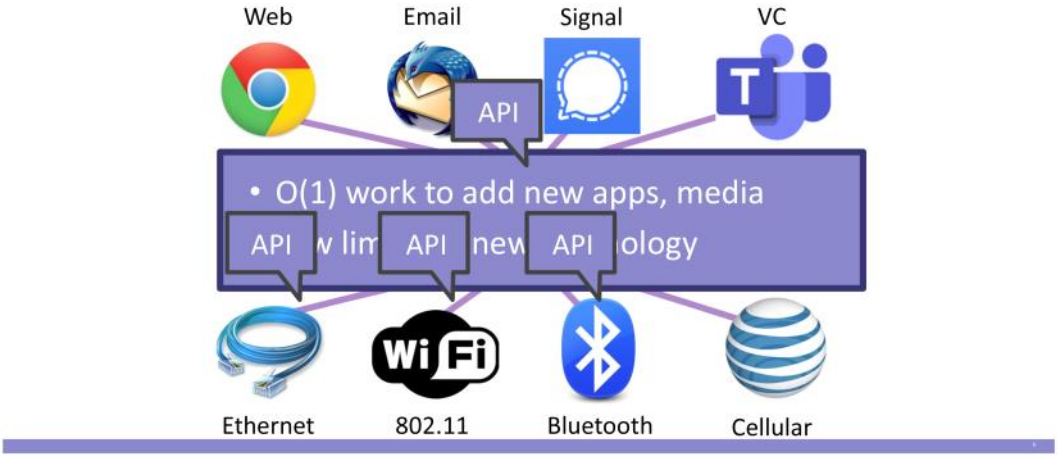
So does Email, Signal

This is not a good thing since it's not adaptable, if there is new networking application, then you have to write code for this new service then

More Problems



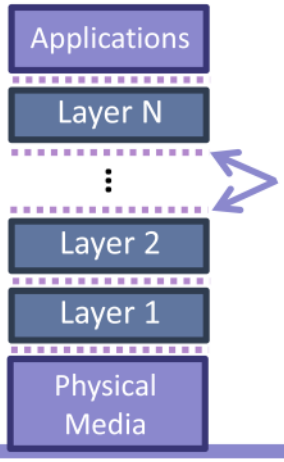
Solution: Use Indirection



Magical Networ Abstraction Layer is added : This is an itnerface that is able to break up amount of binnding when it comes to the applications and the kind of networks

API: Application Progarmming Langauge

Layered Network Stack



Modularity
- Does not specify an implementation
- Instead, tells us how to organize functionality

Encapsulation
- Interfaces define cross-layer interaction
- Layers only rely on those below them

Flexibility
- Reuse of code across the network
- Module implementations may change

Unfortunately, there are tradeoffs
- Interfaces hide information
- As we will see, may hurt performance…

There are stack of layers when it comes to networking

Each layer have distinct API, and there is no tight bindings between the applications and physical media

This is showing the good parts of this interface approach

Tradeoffs: Interfaces hide information --

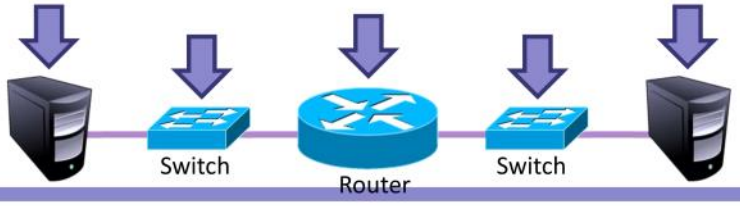
Key Questions

How do we divide functionality into layers?

- Routing
- Congestion control
- Error checking
- Security
- Fairness
- And many more…

How do we distribute functionality across devices? Or in which devices should the functionality be bound?

- Example: who is responsible for security?



Outline

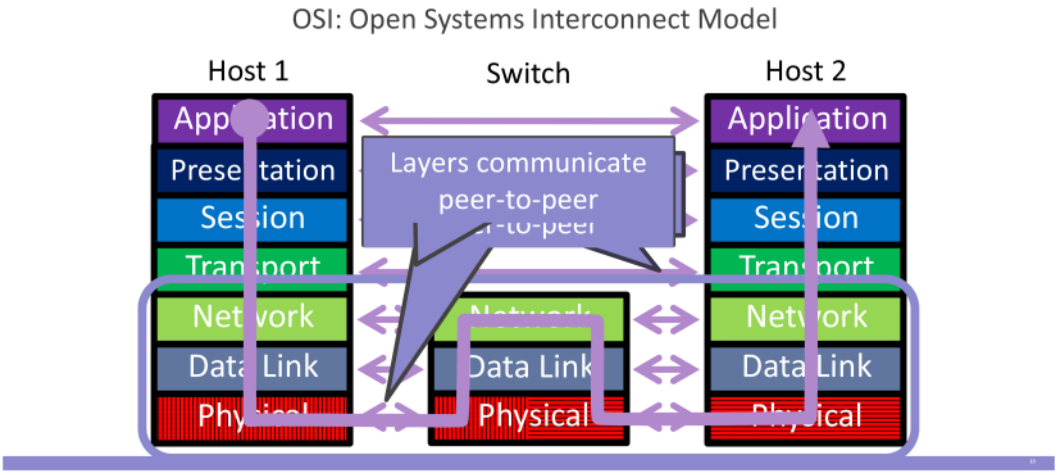
Layering

- The OSI Model

Communicating

- The End-to-End Argument

The ISO OSI Model



It defines seven layers which form the stack for the internet

Each layer and between the different systems, they only talk to the same peer layers --

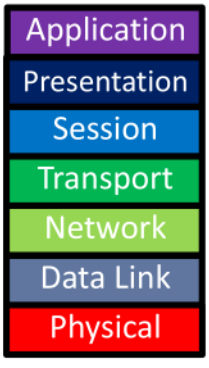
- Physical talks to physical layer of the switch
- Etc
- Application is talking to application on the other machine

Transport is UDP and TCP

How does data flow through the stack?

- Application from home machine goes down the stack, reaches physical layer, then talk to physical layer of the switch component and goes up to networking, down to physical layer, and then talks to the other machine you want to talk to

Layer Features

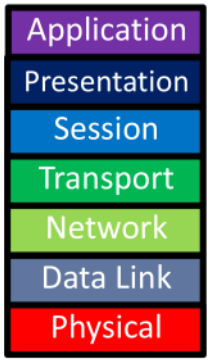


Service
- What does this layer do?

Interface
- How do you access this layer?

Protocol
- How is this layer implemented?

Physical Layer



Service
- Move information between two systems connected by a physical link

Interface
- Specifies how to send one bit

Protocol
- Encoding scheme for one bit
- Voltage levels
- Timing of signals

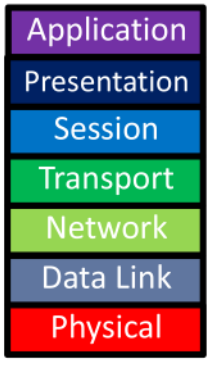
Examples: coaxial cable, fiber optics, radio frequency transmitters

It recieves bits (o and 1) and make it into a physical medium to then give to the world

Transmit bits

How to encode bits into physical media-- look at voltage levels, and timing of signals

Data Link Layer



Service
- Data framing: boundaries between packets
- Media access control (MAC)
- Per-hop reliability and flow-control

Interface
- Send one packet between two hosts connected to the same media

Protocol
- Physical addressing (e.g. MAC address)

Examples: Ethernet, Wifi, DOCSIS

MAC:
- Instances of physical connection:: Not a shared space
- Wireless aspect is a shared aspect, so this entails more complexity
  - Only one transmitter at a time so that each other data doesn't consume each other

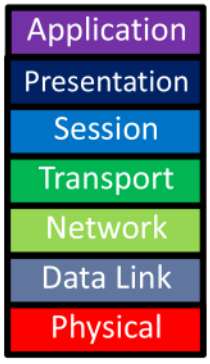
Packet: Discrete bunch of data that needs to go together, sequence of bits

Data link will frame it, and give some medium data maybe and give it to the physical layer

My computer to wifi acess point, data link between point to point on the netwrok

Ex:
DOCSIS: Data link proticol for cable motums.
Bluetooth
5G, 4G, 3G

Network Layer



Service
- Deliver packets across the network
- Handle fragmentation/reassembly
- Packet scheduling
- Buffer management

Interface
- Send one packet to a specific destination

Protocol
- Define globally unique addresses
- Maintain routing tables

Example: Internet Protocol (IP), IPv6

Network layer is the routing aspect of things:
End to end with many intermediate aspects

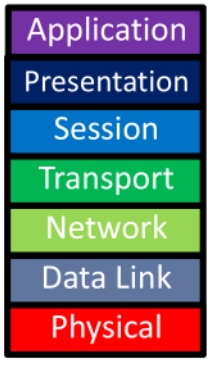
Tells your data where to go with many hops

Handle Fragmentation: Be able to take many fragments of packets and compose them together to get general idea of the packet
There can be some fragmentation in regards hop to hop in routing

Packets getting broken up to many aspect -- it is able to reassembly the packets

Reassembly needs to be able to take in to aspect the order of the packets broken up (Networking layer of your system)

Transport Layer



Service
- Multiplexing/demultiplexing
- Congestion control
- Reliable, in-order delivery

Interface
- Send message to a destination

Protocol
- Port numbers
- Reliability/error correction
- Flow-control information

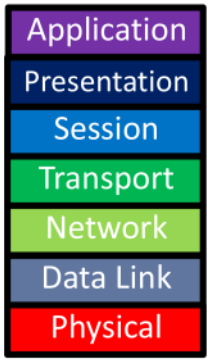
Examples: UDP, TCP

Goal: Ports // Delivered something to machine, and now it needs to be delivered to an application--

Port numbers

Congestion Control: Detecting slowdowns in the network and reacting them -- and you yourself don't create your own congestion

Session Layer



Service
- Access management
- Synchronization

Interface
- It depends…

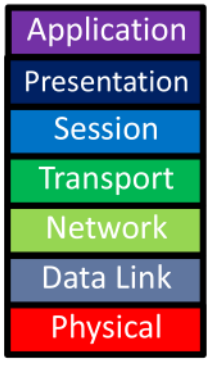
Protocol
- Token management
- Insert checkpoints

Examples: none

In the abstract model of the internet, there is no concrete examples of this

There doesn't exist on the actual outside world in the layer aspect - you as yourself need to be able to implement this

Presentation Layer



Service
- Convert data between different representations
- E.g. big endian to little endian
- E.g. Ascii to Unicode

Interface
- It depends…

Protocol
- Define data formats
- Apply transformation rules

Examples: none

It is able to convert data between different representations:
Big endian vs little endian
- CPU order bits differently
- High to low: Big endian on how the bits are ordered
- Low to high: Little endian on how the bits are ordered

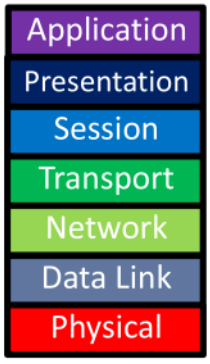
Encoding text also varies

____

It is able to take any format of the data(shown in notes above), it is able to convert to whatebver format you are wanting on your system
- Can convert between little endian and big endian

No example of this, this isnt implemented
You as coder need to be able to hve clear communication between the clients on what data type if expected

Application Layer



Service
- Whatever you want :)

Interface
- Whatever you want :D

Protocol
- Whatever you want ;)

Examples: turn on your smartphone and look at the list of apps

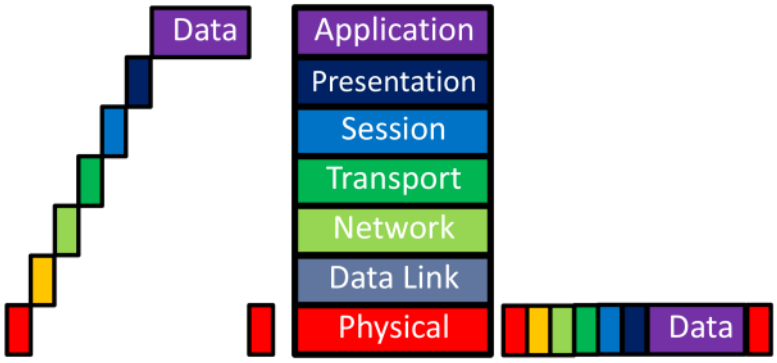
Logistics

Next week's exercise (Spanning Tree paper) posted
- Starting with this paper, exercise will be due **Thursday**, not Monday

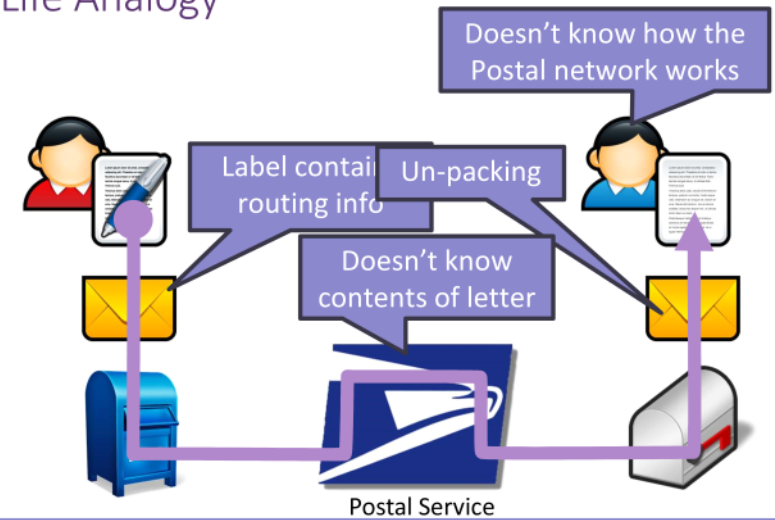
Project 1 due Tuesday
- If you haven't started yet, start now!

Encapsulation

How does data move through the layers?

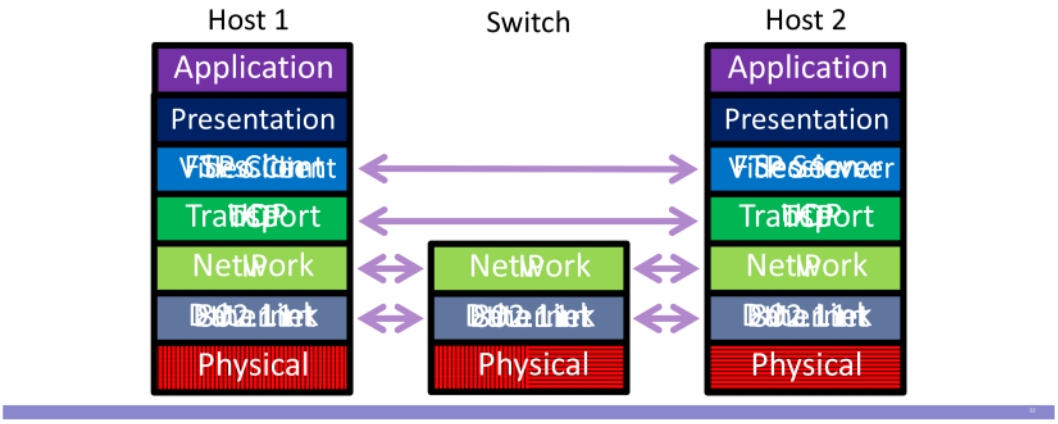


Real Life Analogy



Mail anaology helps see the visual aspect of the network

Network Stack in Practice



Remove Session and Presentation, so remove them

Don't really talk about the physical layer, also remove them

We are left with:

Application	< >	Application
Transport	< >	Transport
Network	Network	Network
Data Link	Data Link	Data Link

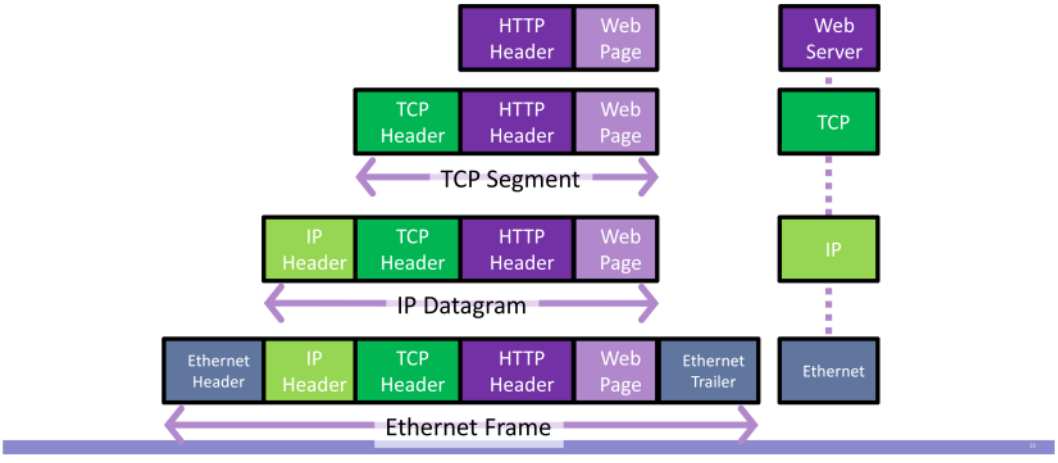
Remove abstract naming:
Ex of what is being used at each layer::

FTP Client	< >	FTP Server
TCP	< >	TCP
IP	Ip	IP
Ethernet	Ethernet	Ethernet

Another example:

Video Client	< >	Video Server
UDP	< >	UDP
IP	Ip	IP
802.11n	802.11n	802.11n

Encapsulation, Revisited



Web server talk in regards to HHTp, why there is designated into the header

Moves to socket to the TCP protocol
-- Turns into TCP segment--

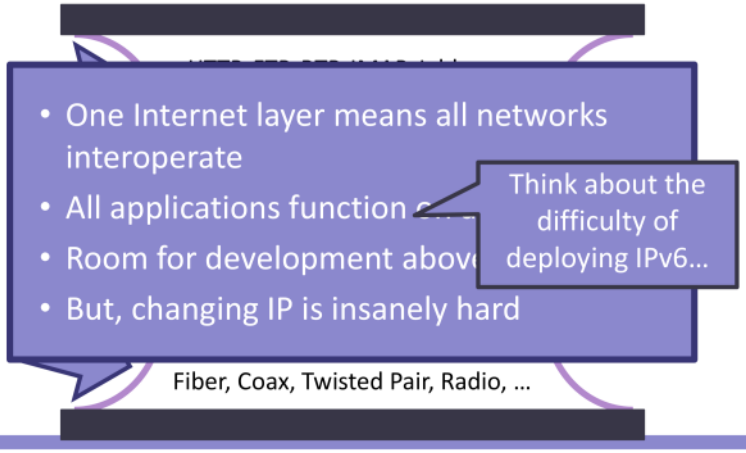
IP Layer
IP header is added here
-- Turns into IP dataGram--

Ethernet will add header and a footer for framing aspect,
-- Turns into ethernet frame--

Random Note: Ethernet is faster due to the physical aspect because there is minimal interference, and it is solely isolated -- could potentially not be shared

Need headers -- if corrupted, this will stop whatever networking is going on

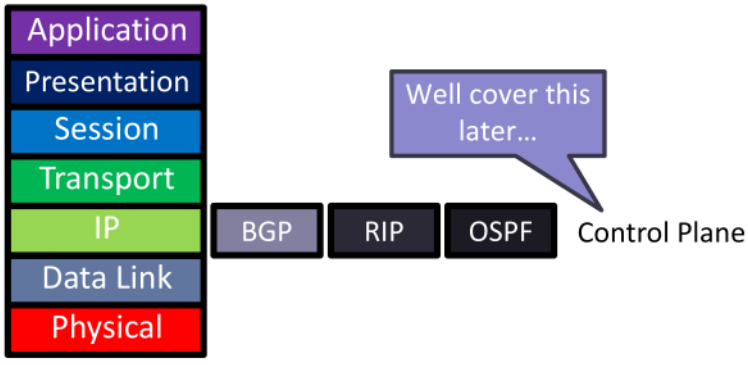
The Hourglass



Dual stacking when there is a devie with IPv6 and IPv4

Orthogonal Planes

Control plane: How **Internet paths** are established

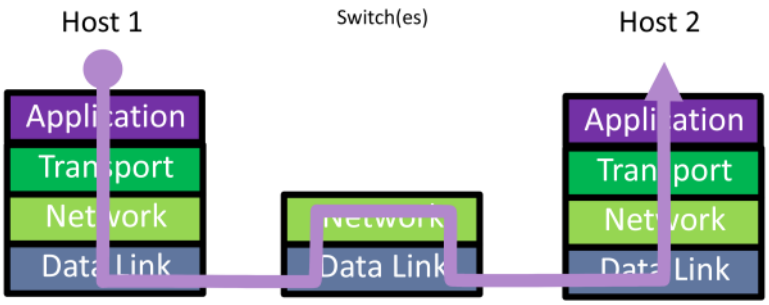


Orthogonal Plane/ Control Plane;;

It is about routing

Orthogonal Planes

Data plane: How data is **forwarded** over Internet paths



Reality Check

The layered abstraction is very nice

Does it hold in reality?

No.

Firewalls

- Analyze application layer headers

Transparent Proxies

- Simulate application endpoints within the network

NATs

- Break end-to-end network reachability

Firewall is able to look at all the layers, even though this wasn't initially described that si allowed

NATS: are able to alter Ip and port stuff, should not be able to happen

Outline

Layering

- The OSI Model

Communicating

- The End-to-End Argument