Networking Introduction

ComS 252 — Iowa State University

Barry Britt and Andrew Miner

Standard Disclaimer

- Networking can be complex
  - There are semester–long courses for just "Networks"
- Most networking concepts are highly theoretical
  - But are also practical
- We will cover "just enough" in a few lectures
- We will start with a basic view
  and then work towards reality

What is a network?

What is a network?

A network is

What is a network?

A network is

- an interconnected collection
  - (we will discuss "how" they are interconnected)
- of computers and hardware
  - (hardware: switches, routers, etc.)
- for sharing and retrieving
- information and resources.

Why networks?

Why networks?

Networks facilitate communication

- Email / instant messaging
- Chat / UseNet
- Telephone calls / Skype
- Conferencing

Why networks?

Networks facilitate communication

- Email / instant messaging
- Chat / UseNet
- Telephone calls / Skype
- Conferencing

Networks facilitate information exchange

- File sharing
- Collaboration suites (often called groupware)
- Calendaring
- System updates

Why networks (2)

Why networks (2)

Networks facilitate resource sharing

- Printing (LPR/LPRNG)
- Files (NFS/CIFS)
- Network connections

Why networks (2)

Networks facilitate resource sharing

- Printing (LPR/LPRNG)
- Files (NFS/CIFS)
- Network connections

Networks let us to fun stuff

- WWW
- Online games

Network Scale

- Networks can be small
  - Home Network
  - Personal Network

Network Scale

- Networks can be small
  - Home Network
  - Personal Network
- Networks can be large
  - University network
  - Business network

Network Scale

- Networks can be small
  - Home Network
  - Personal Network
- Networks can be large
  - University network
  - Business network
- Networks can be REALLY large
  - Internet

Types of Networks

The original types:

- LAN (Local Area Network)
- WAN (Wide Area Network)

Types of Networks

The original types:

- LAN (Local Area Network)
- WAN (Wide Area Network)

"New" types:

- Metropolitan Area Network
- Personal Area Network
- Storage Area Network

LAN

LAN Properties:

- connects devices over a short distance

LAN

LAN Properties:

- connects devices over a short distance
- is usually implemented as one subnet
  - Typically fewer than 256 endpoints
  - We will discuss "subnets" later

LAN

LAN Properties:

- connects devices over a short distance
- is usually implemented as one subnet
  - Typically fewer than 256 endpoints
  - We will discuss "subnets" later
- all networking devices usually owned/operated by a single entity or individual

LAN

LAN Properties:

- connects devices over a short distance
- is usually implemented as one subnet
  - Typically fewer than 256 endpoints
  - We will discuss "subnets" later
- all networking devices usually owned/operated by a single entity or individual
- primarily use ethernet or wireless communication

WAN

WAN Properties:

- connects devices over a large physical distance

WAN

WAN Properties:

- connects devices over a large physical distance
- generally, a WAN interconnects multiple LANs

WAN

WAN Properties:

- connects devices over a large physical distance
- generally, a WAN interconnects multiple LANs
- will be many computers and may be any number of "subnets"

WAN

WAN Properties:

- connects devices over a large physical distance
- generally, a WAN interconnects multiple LANs
- will be many computers and may be any number of "subnets"
- devices will be owned/operated by many individuals

WAN

WAN Properties:

- connects devices over a large physical distance
- generally, a WAN interconnects multiple LANs
- will be many computers and may be any number of “subnets”
- devices will be owned/operated by many individuals
- may use any number of communcation methods
  - ATM (Asynchronous Transfer Mode)
  - ethernet
  - satellite
  - ...

Other types of networks

- Metropolitan Area Network
  - larger than a LAN, but smaller than a WAN.
  - some cities provide Internet access as a service to the local population
- Personal Area Network
  - Bluetooth connection between cell phone and headset
- Storage Area Network
  - CyFiles

Network Topology

Remember WANs?

- Typically interconnect multiple LANs
- Network structures can be complex
  - E.g., one machine can be connected to *multiple* LANs

Network Topology

Remember WANs?

- Typically interconnect multiple LANs
- Network structures can be complex
  - E.g., one machine can be connected to *multiple* LANs

Topology of a network

- Refers to "how everything is connected"
- Usually drawn abstractly, as graphs
  - Circles for devices
  - Lines between devices that can directly communicate
- We will discuss this more, later

The Client/Server Model

- A computing model
- Tasks are partitioned into two parts
  1. Clients: request "service"
  2. Servers: provide "service"
- Provides the foundation for networked computing
- Also used on single systems
  - Remember Microkernels?
  - `X` uses this
    - X server: provides "graphics service"
    - Clients: programs that want to use a GUI

The Client/Server Model

- A computing model
- Tasks are partitioned into two parts
  1. Clients: request "service"
  2. Servers: provide "service"
- Provides the foundation for networked computing
- Also used on single systems
  - Remember Microkernels?
  - `X` uses this
    - X server: provides "graphics service"
    - Clients: programs that want to use a GUI
- When setting up a network service, need to:
  1. Configure the Server
  2. Configure the Client
- Troubleshooting: not always clear which side has the problem

Server Programs

Life of a server program:

1. Listen for requests
2. When a request comes, handle it
   - Provide the requested service (if it is allowed)
   - Often, this means "start a process"...
3. Go to (1)

Server Programs

Life of a server program:

1. Listen for requests
2. When a request comes, handle it
   - Provide the requested service (if it is allowed)
   - Often, this means "start a process"...
3. Go to (1)

- Server programs are (usually) daemons
- Server programs can also be clients
  - For a different "service"
- Server programs may be local or remote to the client
  - "Local": on the same machine
  - "Remote": on a different machine

Server Machines

- Dedicated machines for running server program(s)
- Are often more powerful than typical machines
  - If server programs need to handle lots of clients
- Will normally have redundant components
  - E.g., redundant storage, redundant network interfaces
  - Redundancy can improve reliability
    - Can tolerate failures
  - Redundancy can improve speed
- May be part of a private network, as well as a public network
- Can be clustered with other servers

Clients

Client programs

- Make requests to server(s)
  - Content
  - Services
  - Data
- Receive responses from the server(s)
  - Web Pages
  - Email
  - etc. . .

Client machine

A machine running a client program

Example: web browsing

Client side

- Browser is the client process
  - Firefox, Safari, IE, Chrome, ...
- Browser is running on client machine

Server side

- HTTP server program is server process
- Service is "send web pages"
- Server program (usually) runs elsewhere
  - E.g., some machine in Google's server farm

Example: (secure) remote shell

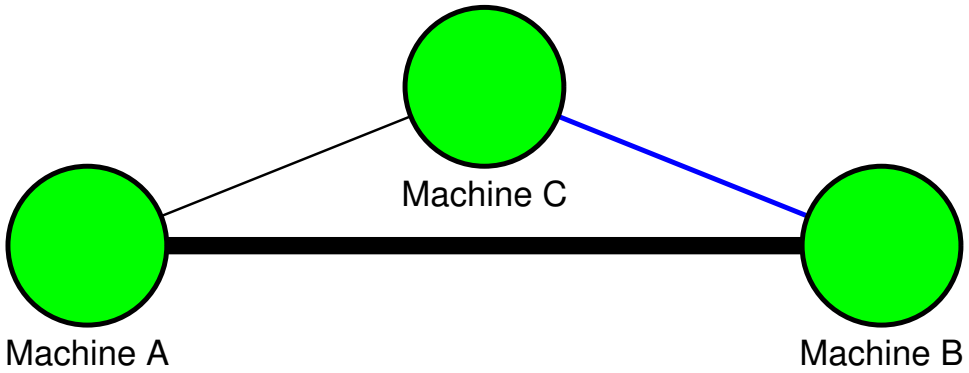
Server side

- `sshd` is the server program
- Service is "run shell comands"
- Suppose my office machine runs this
- Suppose my office machine is `gamera.cs.iastate.edu`

Client side

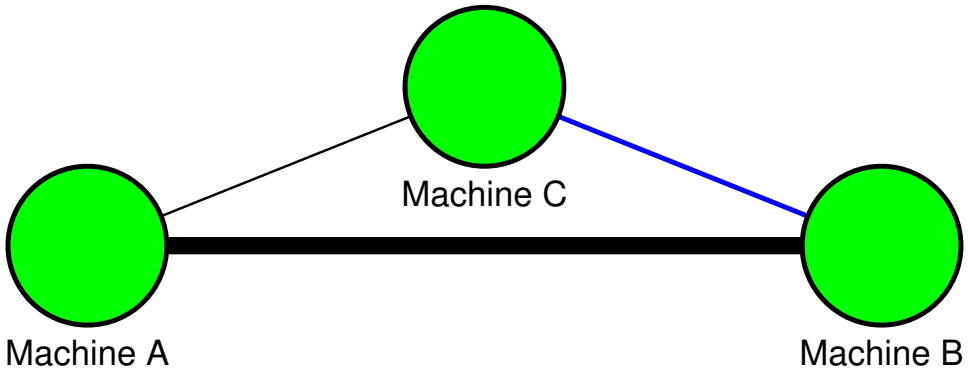
- `ssh` is the client program
- Running `ssh gamera.cs.iastate.edu` gives me
  - a shell running on my office machine
  - from anywhere in the world

Suppose we have a simple network:



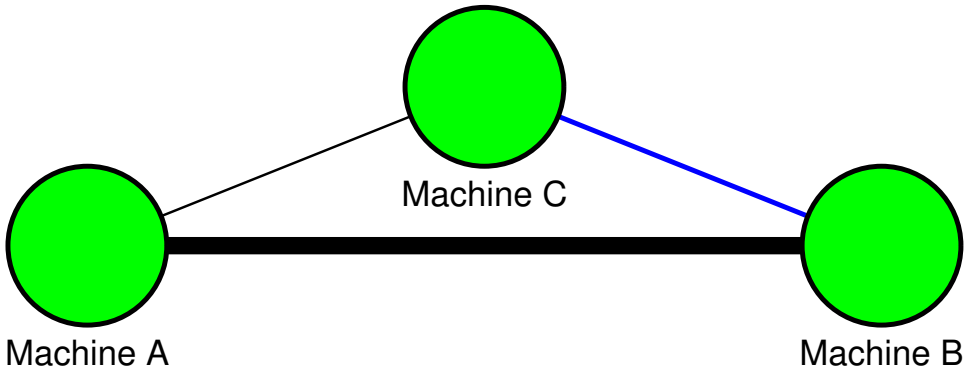
- Links between machines may be different types
- Want client `foo` on Machine A
- Want server `bard` on Machine B
- How do they communicate?

Suppose we have a simple network:



- Links between machines may be different types
- Want client `foo` on Machine A
- Want server `bard` on Machine B
- How do they communicate?
  - Need "names" for the machines
  - Need a way to connect the processes
  - Need a way to send and receive data

Suppose we have a simple network:



- Links between machines may be different types
- Want client `foo` on Machine A
- Want server `bard` on Machine B
- How do they communicate?
  - Need "names" for the machines
  - Need a way to connect the processes
  - Need a way to send and receive data
  - Protocols will handle all of this
    - Lots of protocols...

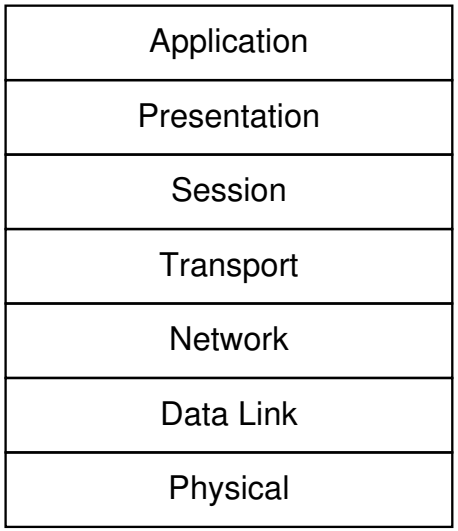
Communication model: idea

- Communication is partitioned into layers
- Each layer provides an abstract view of the "network"
- Layers deal only with the ones immediately above and below
- Each layer has its own protocols
  - The entire collection is the protocol stack or protocol suite
- Each layer will "encode" data for sending
- Each layer will "decode" data for receiving
- Conceptually it works like a pipeline:

```
# echo "Message" | topsend | midsend | botsend | wire |
botrecv | midrecv | toprecv
```

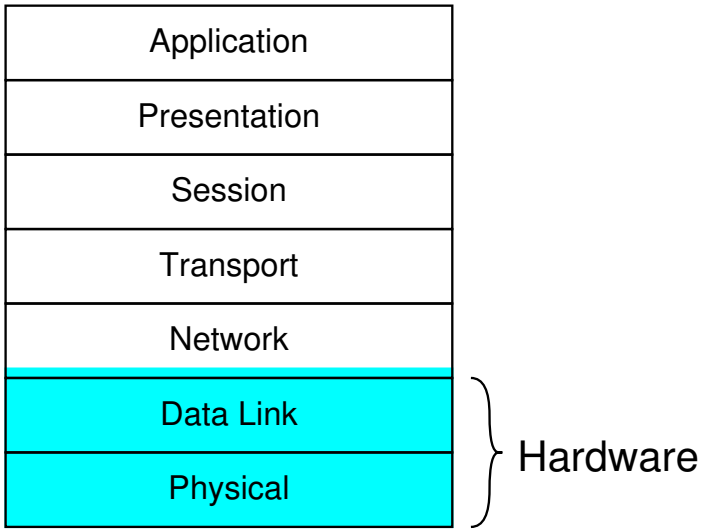
- Exact number and meaning of layers depends on the model

Open Systems Interconnection (OSI) model



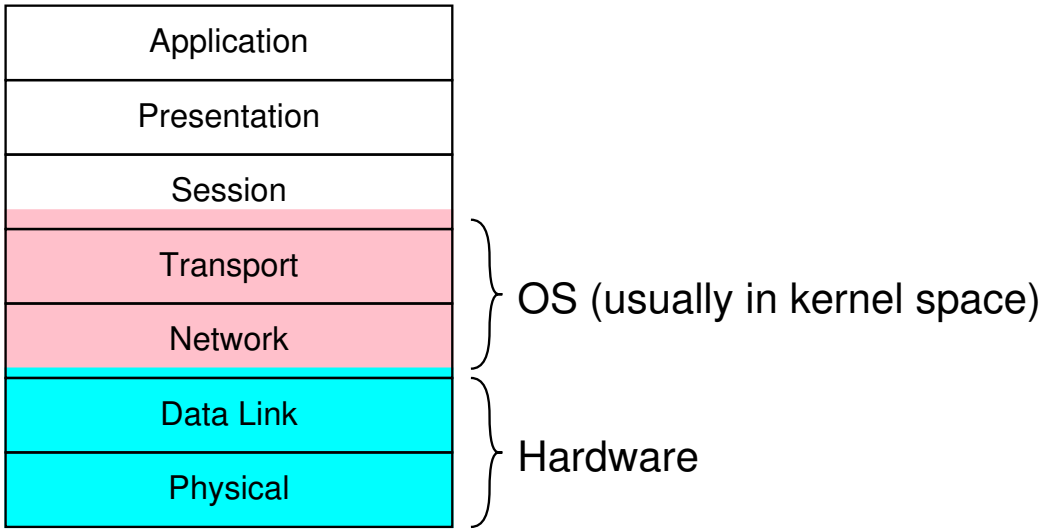
- Abstract model specified by ISO, with 7 layers
- Typically, on a modern computer

Open Systems Interconnection (OSI) model



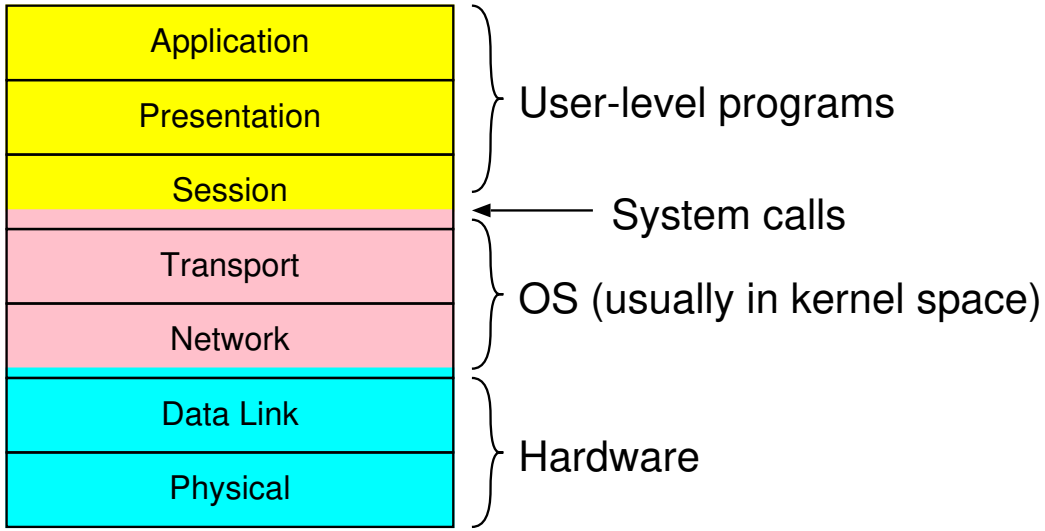
- Abstract model specified by ISO, with 7 layers
- Typically, on a modern computer
  - Physical and Data Link layers are implemented in hardware

Open Systems Interconnection (OSI) model



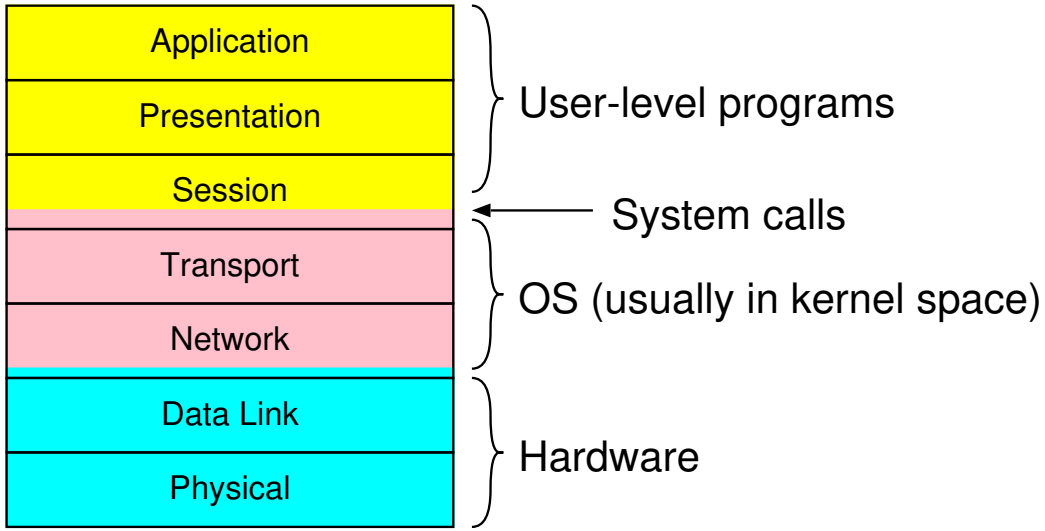
- Abstract model specified by ISO, with 7 layers
- Typically, on a modern computer
  - Physical and Data Link layers are implemented in hardware
  - Network and Transport layers are implemented in the OS

Open Systems Interconnection (OSI) model



- Abstract model specified by ISO, with 7 layers
- Typically, on a modern computer
  - Physical and Data Link layers are implemented in hardware
  - Network and Transport layers are implemented in the OS
  - Top layers are implemented in the client/server programs

Open Systems Interconnection (OSI) model



- Abstract model specified by ISO, with 7 layers
- Typically, on a modern computer
  - Physical and Data Link layers are implemented in hardware
  - Network and Transport layers are implemented in the OS
  - Top layers are implemented in the client/server programs
- We will discuss the layers in detail (from bottom to top)

Physical Layer

Idea: how do we physically send bits on a link between machines

- Defines transmission medium
  - Copper wire (including specs for cables)
  - Fiber optics
  - Radio frequencies (for wireless)
- Defines representation of data units (usually, bits)
- Defines signal timings
- Defines mechanisms to deal with contention
  - What if two machines start "talking" at the same time?

Physical Layer

Idea: how do we physically send bits on a link between machines

- Defines transmission medium
  - Copper wire (including specs for cables)
  - Fiber optics
  - Radio frequencies (for wireless)
- Defines representation of data units (usually, bits)
- Defines signal timings
- Defines mechanisms to deal with contention
  - What if two machines start "talking" at the same time?

Real world analogy — telegraph

- Specs for the electric circuit you need
- The telegraph key
- "Specs" for *dot* and *dash*

Data Link Layer

How do we send frames on a link between machines

- Frame: fixed–length chunk of data
  - Typically on the order of 1 Kb
- Includes error correction for the physical layer

Data Link Layer

How do we send frames on a link between machines

- Frame: fixed–length chunk of data
  - Typically on the order of 1 Kb
- Includes error correction for the physical layer

How does this layer work?

- Frames are broken into bits on sender side
  - Or the data unit for the physical layer
- Bits are collected into frames on receiver side
- Extra bits may be included for error checking
  - E.g., a parity bit
- Bits are passed to the physical layer
- If an error is detected, re-transmit

Data Link Layer (2)

Real world analogy — telegraph operator

- Messages are converted to Morse code
- The operator keys or listens to the message
- No need to know how the telegraph works, internally

Data Link Layer (2)

Real world analogy — telegraph operator

- Messages are converted to Morse code
- The operator keys or listens to the message
- No need to know how the telegraph works, internally

Real network examples

- Ethernet frames (1500 bytes of "payload")
- PPP frames (Point–to–point protocol)

Network Layer

Sends variable–sized packets between machines

- The machines might not have a direct link
- Routing is part of this layer
  - Find a path through the network
  - May have multiple paths — choose one

Network Layer

Sends variable–sized packets between machines

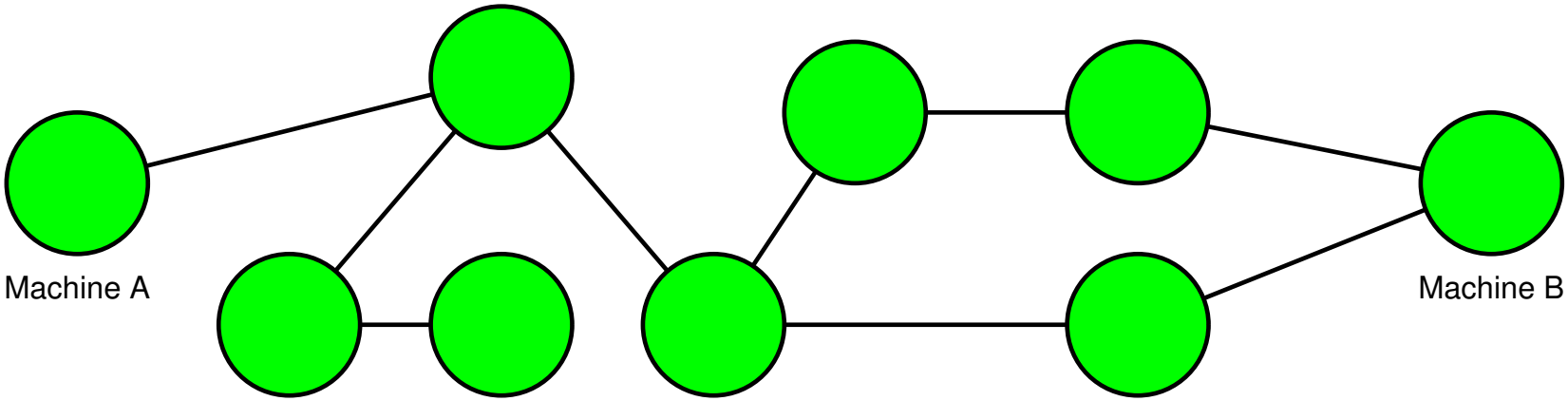
- The machines might not have a direct link
- Routing is part of this layer
  - Find a path through the network
  - May have multiple paths — choose one
- Addressing is part of this layer
  - How machines are "named"
  - Actually, each network device on a machine may get a name
  - Packets include sender and recipient addresses

Network Layer

Sends variable–sized packets between machines

- The machines might not have a direct link
- Routing is part of this layer
  - Find a path through the network
  - May have multiple paths — choose one
- Addressing is part of this layer
  - How machines are "named"
  - Actually, each network device on a machine may get a name
  - Packets include sender and recipient addresses
- May have different *types* of packets
- No guarantee a packet will be delivered
  - Packets may get lost
  - Packets may get delayed

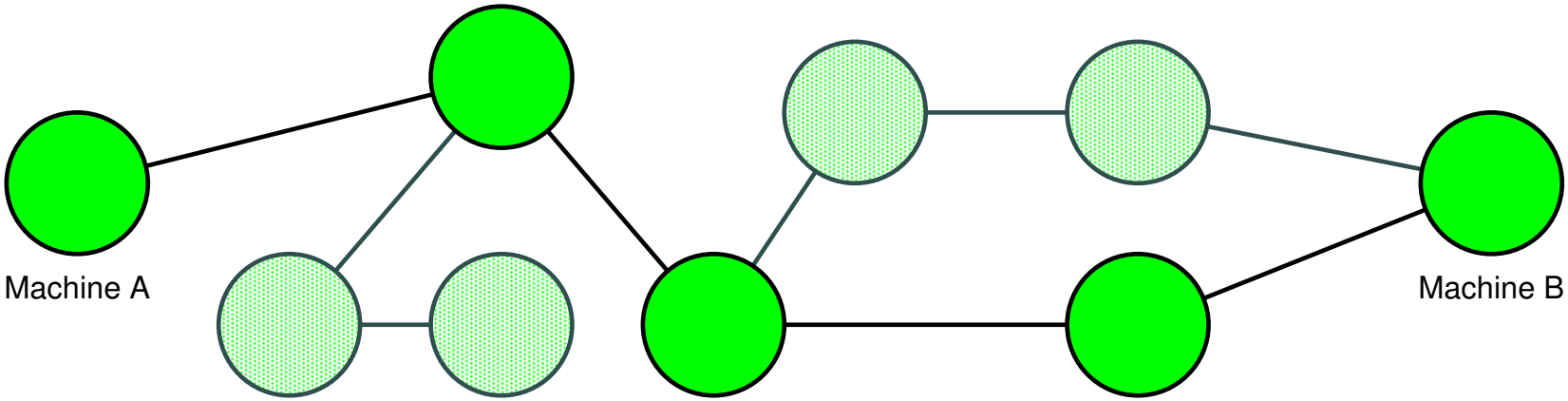
Network Layer: how it works



Want to send a packet from A to B

- Find a route from A to B

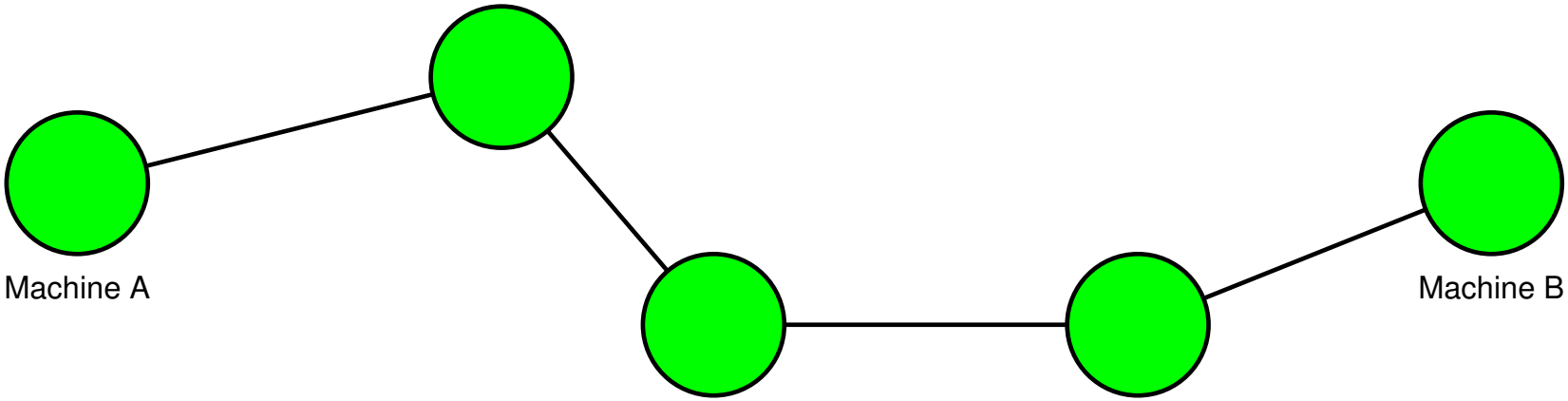
Network Layer: how it works



Want to send a packet from A to B

- Find a route from A to B
  - Magic for now

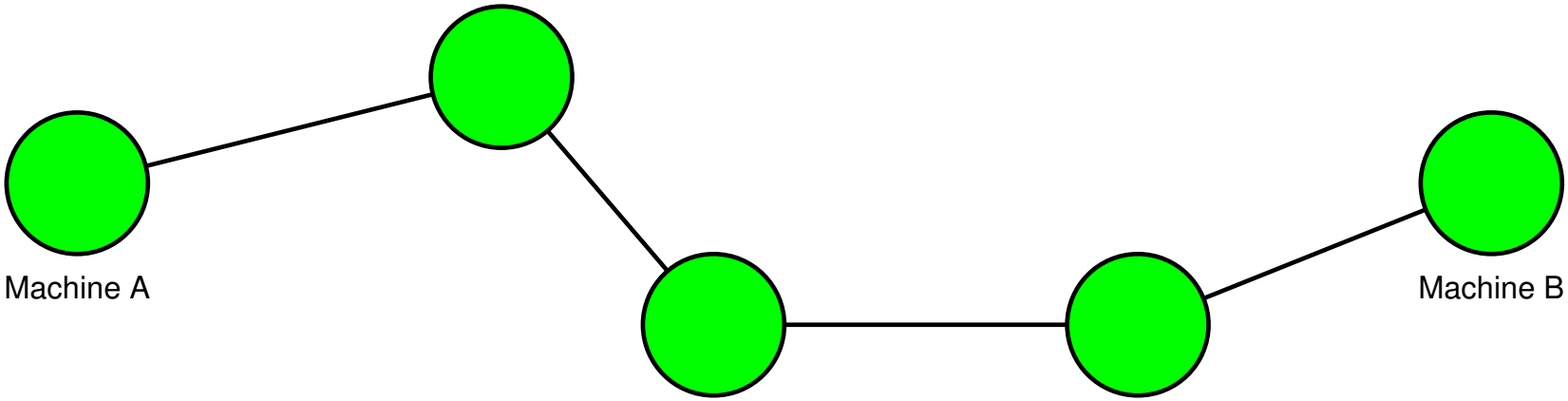
Network Layer: how it works



Want to send a packet from A to B

- Find a route from A to B
  - Magic for now
- For each link:
  1. Break the packet into frames (unless 1 frame is enough)
  2. Send each frame on the link (Data Link Layer handles this)
  3. Re-assemble packet from frames

Network Layer: how it works



Want to send a packet from A to B

- Find a route from A to B
  - Magic for now
- For each link:
  1. Break the packet into frames (unless 1 frame is enough)
  2. Send each frame on the link (Data Link Layer handles this)
  3. Re-assemble packet from frames
- Each link may be a different type
  - Layering abstraction makes this "easy"

Network Layer: real world analogy

Send a telegram to Mark Twain

1. By private courier to Ames Telegraph Office
2. By telegraph to Des Moines Telegraph Office
3. By telegraph to Davenport Telegraph Office
4. By telegraph to Peoria Telegraph Office
5. By telegraph to Springfield Telegraph Office
6. By carrier pigeons to Hannibal Pigeon Office
7. By private courier to Twain residence

Popular Network Layer: IP

- IP: Internet Protocol
- IP addresses are the "network layer addresses"
  - Names for network devices
  - Have the form `129.186.3.66`
  - We will discuss what the digits mean (later)
- But IP addresses are a pain to remember

Popular Network Layer: IP

- IP: Internet Protocol
- IP addresses are the "network layer addresses"
  - Names for network devices
  - Have the form `129.186.3.66`
  - We will discuss what the digits mean (later)
- But IP addresses are a pain to remember
- Ok, use the fully–qualified domain name instead
  - Have the form `popeye.cs.iastate.edu`
  - Must be unique on the Internet

Popular Network Layer: IP

- IP: Internet Protocol
- IP addresses are the "network layer addresses"
  - Names for network devices
  - Have the form `129.186.3.66`
  - We will discuss what the digits mean (later)
- But IP addresses are a pain to remember
- Ok, use the fully–qualified domain name instead
  - Have the form `popeye.cs.iastate.edu`
  - Must be unique on the Internet
- What happens when I use the FQDN?
  - It is converted into an IP address
  - Done by a DNS server
  - Details are magic for now

More with IP addresses

How does a machine (or network device) get its IP address?

More with IP addresses

How does a machine (or network device) get its IP address?

Statically :

- System administrator chooses an IP address
- It is listed in a configuration file somewhere
- The address is “fixed”

More with IP addresses

How does a machine (or network device) get its IP address?

Statically :

- System administrator chooses an IP address
- It is listed in a configuration file somewhere
- The address is "fixed"

Dynamically :

- Machine asks someone else for an IP address
- Follows the usual client/server model
  - Service is "give out addresses"
- DHCP: most common protocol used today
  - We will discuss this in depth (later)
- The address may change

IP utilities

`ping`

- Utility used to test the reachability of a network host.
- sends ICMP (Internet Control Message Protocol) packets, which are packets that are not intended to transfer data
- Was named after a sonar term, where a pulse of emitted sound was called a "ping".
- Typically sends one packet per second . . .
- . . . forever (or until Ctrl–C)
- Usage: `ping [switches] host`
  - "`host`" can be an IP address or a FQDN
  - `-c` Count (specify number of packets to send)
  - `-o` exit successfully after receiving one packet

Ping example

```
prompt$
```

Ping example

```
prompt$ ping popeye.cs.iastate.edu
```

Ping example

```
prompt$ ping popeye.cs.iastate.edu
PING popeye.cs.iastate.edu (129.186.3.66):  56 data bytes
64 bytes from 129.186.3.66:  icmp_seq=0 ttl=48 time=56.591 ms
```

Ping example

```
prompt$ ping popeye.cs.iastate.edu
PING popeye.cs.iastate.edu (129.186.3.66):  56 data bytes
64 bytes from 129.186.3.66:  icmp_seq=0 ttl=48 time=56.591 ms
64 bytes from 129.186.3.66:  icmp_seq=1 ttl=48 time=66.460 ms
```

Ping example

```
prompt$ ping popeye.cs.iastate.edu
PING popeye.cs.iastate.edu (129.186.3.66):  56 data bytes
64 bytes from 129.186.3.66:  icmp_seq=0 ttl=48 time=56.591 ms
64 bytes from 129.186.3.66:  icmp_seq=1 ttl=48 time=66.460 ms
64 bytes from 129.186.3.66:  icmp_seq=2 ttl=48 time=58.303 ms
```

Ping example

```
prompt$ ping popeye.cs.iastate.edu
PING popeye.cs.iastate.edu (129.186.3.66):  56 data bytes
64 bytes from 129.186.3.66:  icmp_seq=0 ttl=48 time=56.591 ms
64 bytes from 129.186.3.66:  icmp_seq=1 ttl=48 time=66.460 ms
64 bytes from 129.186.3.66:  icmp_seq=2 ttl=48 time=58.303 ms
64 bytes from 129.186.3.66:  icmp_seq=3 ttl=48 time=57.783 ms
```

Ping example

```
prompt$ ping popeye.cs.iastate.edu
PING popeye.cs.iastate.edu (129.186.3.66):  56 data bytes
64 bytes from 129.186.3.66:  icmp_seq=0 ttl=48 time=56.591 ms
64 bytes from 129.186.3.66:  icmp_seq=1 ttl=48 time=66.460 ms
64 bytes from 129.186.3.66:  icmp_seq=2 ttl=48 time=58.303 ms
64 bytes from 129.186.3.66:  icmp_seq=3 ttl=48 time=57.783 ms
64 bytes from 129.186.3.66:  icmp_seq=4 ttl=48 time=62.524 ms
```

Ping example

```
prompt$ ping popeye.cs.iastate.edu
PING popeye.cs.iastate.edu (129.186.3.66):  56 data bytes
64 bytes from 129.186.3.66:  icmp_seq=0 ttl=48 time=56.591 ms
64 bytes from 129.186.3.66:  icmp_seq=1 ttl=48 time=66.460 ms
64 bytes from 129.186.3.66:  icmp_seq=2 ttl=48 time=58.303 ms
64 bytes from 129.186.3.66:  icmp_seq=3 ttl=48 time=57.783 ms
64 bytes from 129.186.3.66:  icmp_seq=4 ttl=48 time=62.524 ms
64 bytes from 129.186.3.66:  icmp_seq=5 ttl=48 time=57.883 ms
```

Ping example

```
prompt$ ping popeye.cs.iastate.edu
PING popeye.cs.iastate.edu (129.186.3.66):  56 data bytes
64 bytes from 129.186.3.66:  icmp_seq=0 ttl=48 time=56.591 ms
64 bytes from 129.186.3.66:  icmp_seq=1 ttl=48 time=66.460 ms
64 bytes from 129.186.3.66:  icmp_seq=2 ttl=48 time=58.303 ms
64 bytes from 129.186.3.66:  icmp_seq=3 ttl=48 time=57.783 ms
64 bytes from 129.186.3.66:  icmp_seq=4 ttl=48 time=62.524 ms
64 bytes from 129.186.3.66:  icmp_seq=5 ttl=48 time=57.883 ms
^C
--- popeye.cs.iastate.edu ping statistics ---
6 packets transmitted, 6 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 56.591/59.924/66.460/3.463 ms
prompt$
```

IP utilities (2)

`traceroute`

- Utility used to trace one possible route to the destination address
- Can also be used to measure transit delays
- Traceroute sends a sequence of pings to hosts with varying "time-to-live" values.
- This continues until the endpoint is reached and a successful ICMP reply message is received.

Traceroute example

```
prompt$
```

Traceroute example

```
prompt$ traceroute -q 1 -w 1 129.186.3.66
```

Traceroute example

```
prompt$ traceroute -q 1 -w 1 129.186.3.66
traceroute to 129.186.3.66 (129.186.3.66), 64 hops max, 52 byte packets
1  192.168.1.1 (192.168.1.1)  3.754 ms
2  *
```

Traceroute example

```
prompt$ traceroute -q 1 -w 1 129.186.3.66
traceroute to 129.186.3.66 (129.186.3.66), 64 hops max, 52 byte packets
1  192.168.1.1 (192.168.1.1)  3.754 ms
2  *
3  172.30.6.21 (172.30.6.21)  50.538 ms
4  97-64-179-218.client.mchsi.com (97.64.179.218)  15.021 ms
5  isu.icn.state.ia.us (205.221.255.6)  11.839 ms
6  b31dmz1-438.tele.iastate.edu (192.245.179.49)  19.321 ms
7  m22sr1-vlan254.tele.iastate.edu (129.186.254.138)  13.479 ms
8  popeye.cs.iastate.edu (129.186.3.66)  12.522 ms
prompt$
```

Transport Layer

Idea: how do we transfer data between computers

- "Data" may be "messages"
- "Data" may be a stream of bytes
  - E.g., Skype, VOIP

This layer deals with:

- Breaking data into network packets
  - Network layer handles sending packets
- Flow control
- Handling errors (e.g., lost packets)

Transport Layer: real world analogy

Send me page 45 of "The Adventures of Huckleberry Finn"

1. Page 45 is broken into several telegrams
2. Telegrams include sequence numbers
3. Send telegrams
4. Reply with acknowledgements to detect lost telegrams
5. Any lost telegrams are re-sent
6. Collect telegrams and reconstruct page 45

Transport Layer: reality

Two protocols are usually included at this layer

Transport Layer: reality

Two protocols are usually included at this layer

UDP: User Datagram Protocol

- Data is sent in fixed–size "datagrams"
- "Best–effort" delivery only
  - Sometimes called "unreliable datagram protocol"
- A very thin wrapper around network packets

Transport Layer: reality

Two protocols are usually included at this layer

UDP: User Datagram Protocol

- Data is sent in fixed–size "datagrams"
- "Best–effort" delivery only
  - Sometimes called "unreliable datagram protocol"
- A very thin wrapper around network packets

TCP: Transmission Control Protocol

- Data is viewed as a "stream"
- "Guaranteed" delivery
- Conceptually: a giant, reliable, pipe of data

Transport Layer: reality (2)

- Wait, so what is TCP/IP?

Transport Layer: reality (2)

- Wait, so what is TCP/IP?
  - TCP running on top of IP
  - Remember: IP is a popular network layer protocol
  - . . . but not the only one

Transport Layer: reality (2)

- Wait, so what is TCP/IP?
  - TCP running on top of IP
  - Remember: IP is a popular network layer protocol
  - . . . but not the only one
- Some services use UDP
- Some services use TCP
- Some services can use either one
- Client and server processes have to agree
  - I.e., both use TCP or both use UDP

Session Layer

Idea: deal with communication sessions between computers

- Initializes connections as necessary
- Removes a connection when no longer necessary
- Manages the connection during transfer
- Error handling: recovering or closing failed sessions
- Connections may be

  Full duplex : may send and receive

  Half duplex : may send or receive, not both

Session Layer

Idea: deal with communication sessions between computers

- Initializes connections as necessary
- Removes a connection when no longer necessary
- Manages the connection during transfer
- Error handling: recovering or closing failed sessions
- Connections may be

  Full duplex : may send and receive

  Half duplex : may send or receive, not both

Continuing the analogy...

- Telegrams to set up a communication stream
- ...before "send me page 45" at the transport layer
- And telegrams to terminate the session

Session Layer: reality (1)

- Ordinary pipes may be viewed as a session–layer protocol
  - They are a "communication session" between processes
  - But usually there is no "networking"

Session Layer: reality (1)

- Ordinary pipes may be viewed as a session–layer protocol
  - They are a "communication session" between processes
  - But usually there is no "networking"
- TCP also includes session control
  - So partly, TCP is in this layer

Session Layer: reality (1)

- Ordinary pipes may be viewed as a session–layer protocol
  - They are a "communication session" between processes
  - But usually there is no "networking"
- TCP also includes session control
  - So partly, TCP is in this layer

Let's think about setting up a session between a client and server

Session Layer: reality (1)

- Ordinary pipes may be viewed as a session–layer protocol
  - They are a "communication session" between processes
  - But usually there is no "networking"
- TCP also includes session control
  - So partly, TCP is in this layer

Let's think about setting up a session between a client and server

- On different machines

Session Layer: reality (1)

- Ordinary pipes may be viewed as a session–layer protocol
  - They are a "communication session" between processes
  - But usually there is no "networking"
- TCP also includes session control
  - So partly, TCP is in this layer

Let's think about setting up a session between a client and server

- On different machines
- The server will "listen" for connections

Session Layer: reality (1)

- Ordinary pipes may be viewed as a session–layer protocol
  - They are a "communication session" between processes
  - But usually there is no "networking"
- TCP also includes session control
  - So partly, TCP is in this layer

Let's think about setting up a session between a client and server

- On different machines
- The server will "listen" for connections
- How does the client "find" the server process?

Session Layer: reality (2)

Port Numbers

- Port numbers are a way to plug communication into a process
- Each port number can have at most 1 process listening
  - Technical term and system call is `bind`
- Specific port numbers are reserved for specific services
  - E.g., port 22 is reserved for `ssh`
  - Nothing magic here, just convention
  - `http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`
- Port numbers below 1024 are reserved
  - Only a process running as `root` can `bind` to them

Firewalls

- A firewall is a mechanism to ignore certain network packets
- Firewalls may be configured at the previous 3 levels

Firewalls

- A firewall is a mechanism to ignore certain network packets
- Firewalls may be configured at the previous 3 levels

  session layer : drop packets by port number

Firewalls

- A firewall is a mechanism to ignore certain network packets
- Firewalls may be configured at the previous 3 levels

  session layer : drop packets by port number
  transport layer : drop packets based on
  - TCP or UDP
  - State of a TCP connection

Firewalls

- A firewall is a mechanism to ignore certain network packets
- Firewalls may be configured at the previous 3 levels

  session layer : drop packets by port number
  transport layer : drop packets based on
    - TCP or UDP
    - State of a TCP connection

  network layer : drop packets based on the sender's address

Firewalls

- A firewall is a mechanism to ignore certain network packets
- Firewalls may be configured at the previous 3 levels

  session layer : drop packets by port number
  transport layer : drop packets based on
    - TCP or UDP
    - State of a TCP connection

  network layer : drop packets based on the sender's address
- We will discuss this more, later

Presentation Layer

Idea: get data ready for final presentation

- Collects / aggregates data
- Converts data
  - "Network" format may differ from "application" format
  - E.g., how are integers transmitted?
  - E.g., transmitted data is ASCII, application needs PETSCII

Application Layer

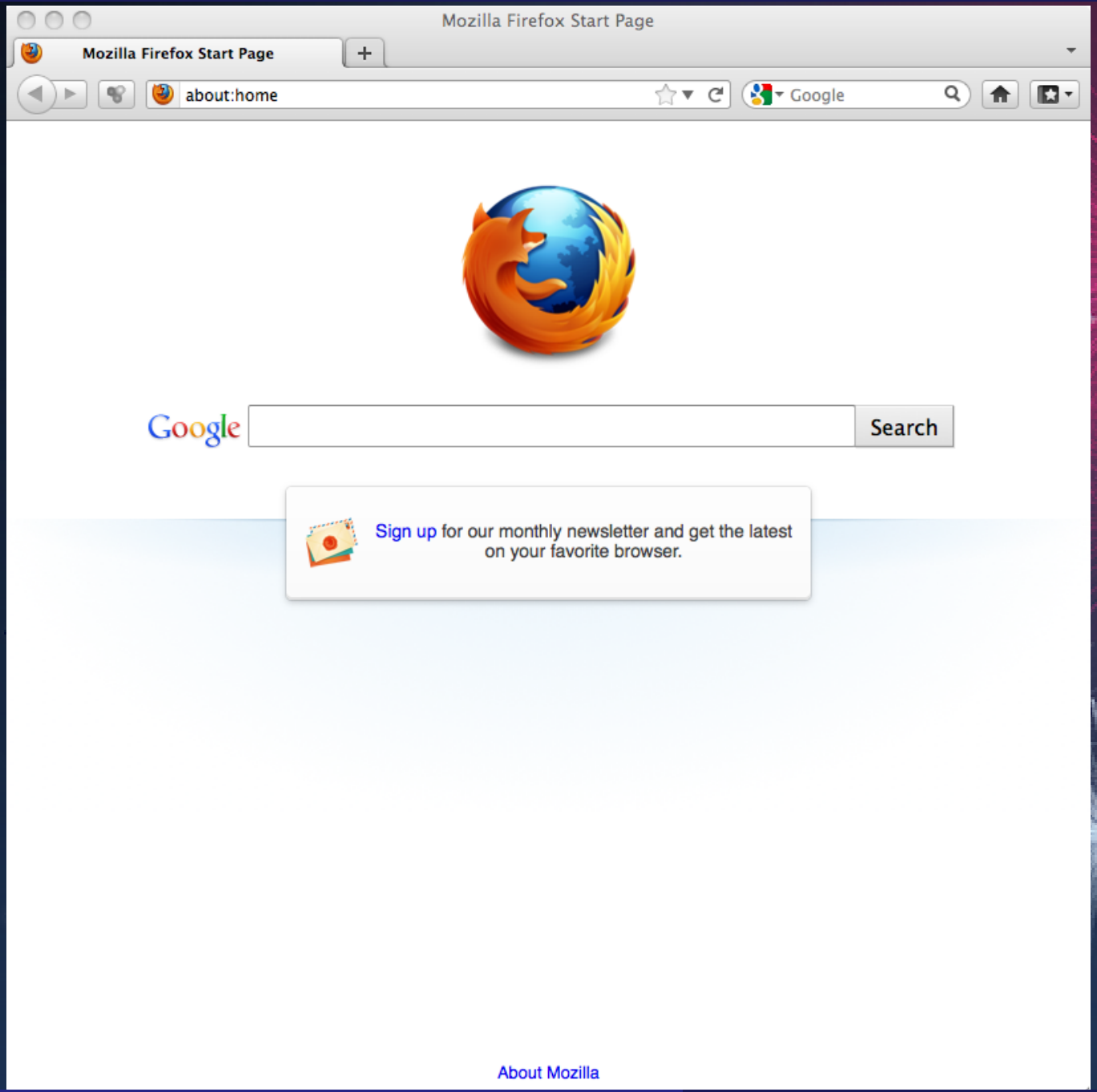
- Layer that interacts with the end user
- Handles top–layer client/server protocol
  - Including synchronization of communication

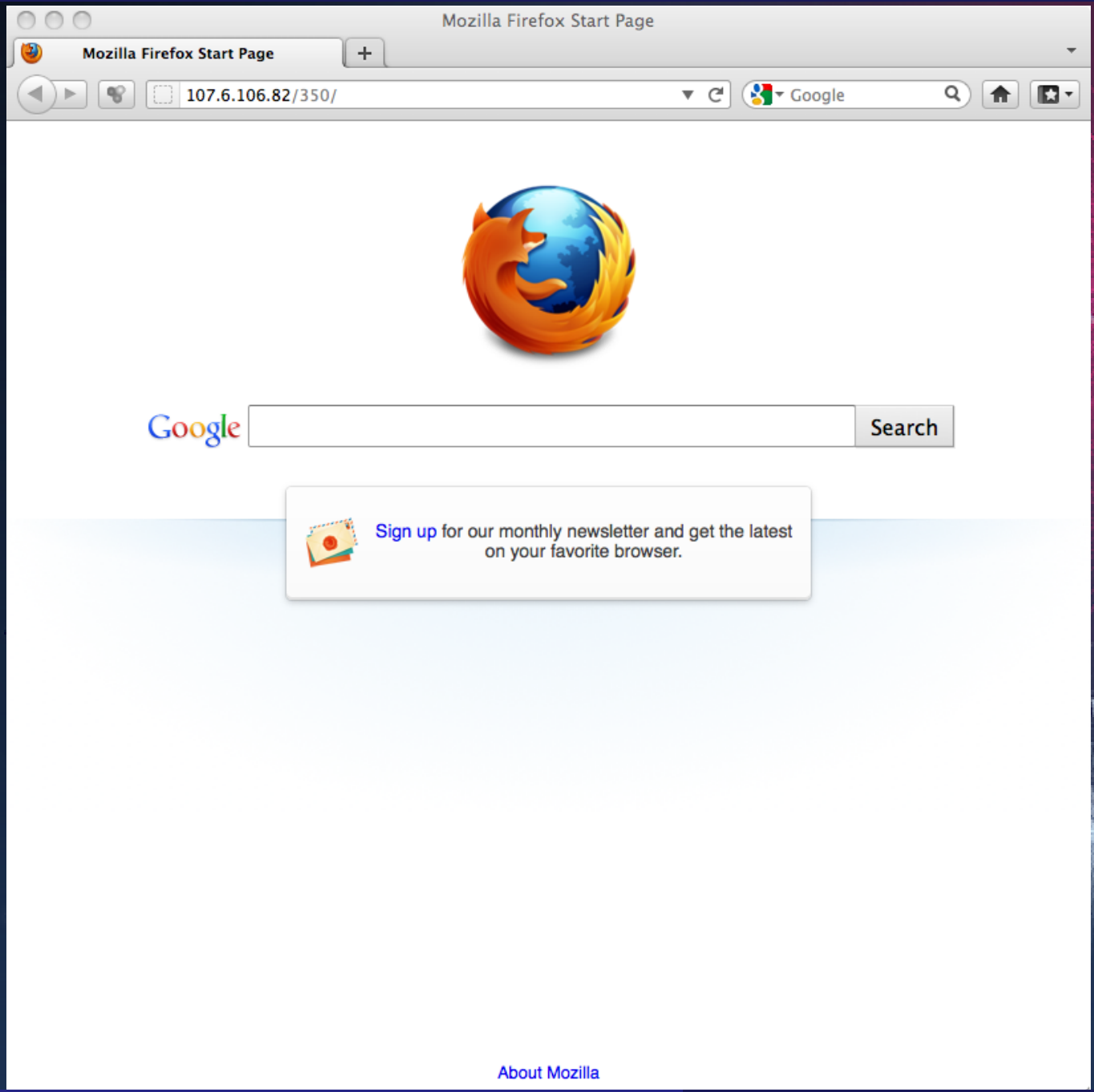
Application Layer

- Layer that interacts with the end user
- Handles top–layer client/server protocol
  - Including synchronization of communication

Application layer: reality

- HTTP/HTTPS (web browsing)
- SMTP/IMAP/POP3 (email)
- DNS/DHCP (internet)
- SSH/RDP (remote sessions)
- FTP/SFTP/SCP (file transfer)
- SSL (security)
- LDAP/AD (directory services)
- SNMP (network management)


Mozilla Firefox Start Page
Mozilla Firefox Start Page
about:home
Google
Google
Search
Sign up for our monthly newsletter and get the latest on your favorite browser.
About Mozilla


Mozilla Firefox Start Page
Mozilla Firefox Start Page
107.6.106.82/350/
Google
Google
Search
Sign up for our monthly newsletter and get the latest on your favorite browser.
About Mozilla

xkcd: Network

xkcd: Network

107.6.106.82/350/

Google

ARCHIVE
WHAT IF?
BLAG
STORE
ABOUT

xkcd

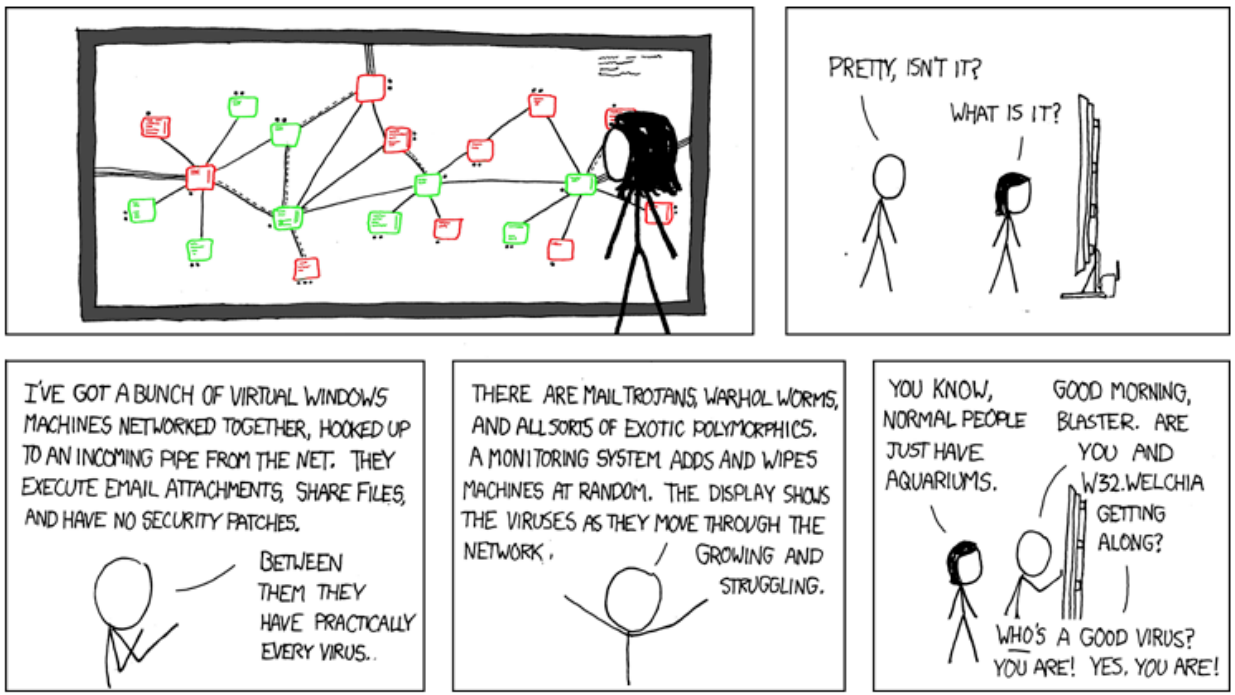
A WEBCOMIC OF ROMANCE, SARCASM, MATH, AND LANGUAGE.

NEW WHAT-IF ARTICLE:

WHAT IF YOU SET OFF A NUCLEAR BOMB IN THE MARIANA TRENCH?.

NETWORK

I< < PREV RANDOM NEXT > >I



I< < PREV RANDOM NEXT > >I

What just happened?

1. Application layer: get url `107.6.106.82/350`

What just happened?

1. Application layer: get url `107.6.106.82/350`
2. Presentation layer: get url `107.6.106.82/350`

What just happened?

1. Application layer: get url `107.6.106.82/350`
2. Presentation layer: get url `107.6.106.82/350`
3. Session layer:
   - Open a TCP connection with `107.6.106.82` on port 80
   - Send `GET 350`
   - Read reply
   - Close connection

What just happened?

1. Application layer: get url `107.6.106.82/350`
2. Presentation layer: get url `107.6.106.82/350`
3. Session layer:
   - Open a TCP connection with `107.6.106.82` on port 80
   - Send `GET 350`
   - Read reply
   - Close connection
4. Transport layer (TCP):
   - 3–way TCP handshake to establish connection
   - Send packet(s) for `GET 350`
   - Receive reply packets

What just happened?

1. Application layer: get url `107.6.106.82/350`
2. Presentation layer: get url `107.6.106.82/350`
3. Session layer:
   - Open a TCP connection with `107.6.106.82` on port 80
   - Send `GET 350`
   - Read reply
   - Close connection
4. Transport layer (TCP):
   - 3–way TCP handshake to establish connection
   - Send packet(s) for `GET 350`
   - Receive reply packets
5. Network layer (IP):
   - Route packets between us and `107.6.106.82`

What just happened?

1. Application layer: get url `107.6.106.82/350`
2. Presentation layer: get url `107.6.106.82/350`
3. Session layer:
   - Open a TCP connection with `107.6.106.82` on port 80
   - Send `GET 350`
   - Read reply
   - Close connection
4. Transport layer (TCP):
   - 3–way TCP handshake to establish connection
   - Send packet(s) for `GET 350`
   - Receive reply packets
5. Network layer (IP):
   - Route packets between us and `107.6.106.82`
6. Data link layer (ethernet):
   - Break packets into frames, transmit, reassemble

What just happened?

1. Application layer: get url `107.6.106.82/350`
2. Presentation layer: get url `107.6.106.82/350`
3. Session layer:
   - Open a TCP connection with `107.6.106.82` on port 80
   - Send `GET 350`
   - Read reply
   - Close connection
4. Transport layer (TCP):
   - 3–way TCP handshake to establish connection
   - Send packet(s) for `GET 350`
   - Receive reply packets
5. Network layer (IP):
   - Route packets between us and `107.6.106.82`
6. Data link layer (ethernet):
   - Break packets into frames, transmit, reassemble
7. Physical:
   - Send bits on CAT5 cable

What just happened? (2)

That was all, right?

What just happened? (2)

That was all, right?
Not quite:

What just happened? (2)

That was all, right?

Not quite:

8. Presentation layer:
   - The web page has images to load

What just happened? (2)

That was all, right?

Not quite:

8. Presentation layer:
   - The web page has images to load
   - For each image URL . . .

What just happened? (2)

That was all, right?

Not quite:

8. Presentation layer:
   - The web page has images to load
   - For each image URL ...
9. Session layer:
   - Open a TCP connection with `107.6.106.82` on port 80
   - Send `GET <url>` for the image
   - Read reply
   - Close connection

⋮

And now for something completely different. . .

Apache Software Foundation

- Home: `http://www.apache.org`
- A group of open–source developers, not unlike GNU
- Non–profit corporation
- Projects are released under ASL (Apache Software License)
  - A "Free" software license (i.e., open–source)
  - Similar to GPL except for issues with Patents

Apache Software Foundation

- Home: `http://www.apache.org`
- A group of open–source developers, not unlike GNU
- Non–profit corporation
- Projects are released under ASL (Apache Software License)
  - A "Free" software license (i.e., open–source)
  - Similar to GPL except for issues with Patents
- And why are we talking about this?

Apache Software Foundation

- Home: `http://www.apache.org`
- A group of open–source developers, not unlike GNU
- Non–profit corporation
- Projects are released under ASL (Apache Software License)
  - A "Free" software license (i.e., open–source)
  - Similar to GPL except for issues with Patents
- And why are we talking about this?
- Apache makes open–source web server software. . .

Apache HTTP Server

- Apache 1.0 released in 1995
- One year later — #1 server on Internet
- Remained #1 for 20 years
- Top servers are now Apache, Microsoft, and nginx
  - Depending how you count
  - Some companies now use custom servers (e.g., Google)
  - See `http://news.netcraft.com`
- Runs on some of Internet's busiest sites
- Runs on Windows, Linux, and other Unix platforms
- Uses modules to customize features
  - Can be loaded at compile or run time

Life of a web server

- Listen for requests
- Reply to requests
  - Usually, serve "web pages"
  - . . . or images, audio files, etc.
  - Call this "content"
- Content may be:

Life of a web server

- Listen for requests
- Reply to requests
  - Usually, serve "web pages"
  - . . . or images, audio files, etc.
  - Call this "content"
- Content may be:

  Static

  - Does not change
  - Taken from files on the filesystem

Life of a web server

- Listen for requests
- Reply to requests
  - Usually, serve "web pages"
  - . . . or images, audio files, etc.
  - Call this "content"
- Content may be:

  Static
  - Does not change
  - Taken from files on the filesystem

  Dynamic
  - May change
  - Built on demand by running a program

Setting up a web server and client

- Configure client
  - That's your browser
  - Not much to configure
- Configure server
  - Install Apache `httpd`
  - Set `httpd` service to start at boot time
  - Edit the configuration file
    - A text file
    - In Fedora: `/etc/httpd/conf/httpd.conf`
    - Has structure and comments
    - Not too difficult to navigate

Configuring Apache HTTP server

- Good news:
  - You do not need to know the HTTP server protocol

Configuring Apache HTTP server

- Good news:
  - You do not need to know the HTTP server protocol
- Bad news:
  - You may need to read documentation
  - Apache has some nice HOWTOs online
  - See `http://httpd.apache.org/`

Configuring Apache HTTP server

- Good news:
  - You do not need to know the HTTP server protocol
- Bad news:
  - You may need to read documentation
  - Apache has some nice HOWTOs online
  - See `http://httpd.apache.org/`
- Troubleshooting:
  - Check the log files!
  - In directory `/var/log/httpd`
  - Entries have date and time information

Handy utility: `curl`

- "Copy URL"
- Command–line utility to transfer data from or to a server
- Supports many protocols, including HTTP
- Useful for debugging web servers and dynamic content
- Check your `man` pages for details

Generic network troubleshooting

- Can packets get from the client to server and back (use `ping`)
- Are packets being dropped (check firewall on client and server)
- Is the server running?
- What happened to the request (check server logs)
  - Note: servers may ignore requests based on IP address
  - As a general rule, for security:
    - Grant access to smallest set of client machines necessary
    - Deny access by default
- On VMs: can see if there is network activity

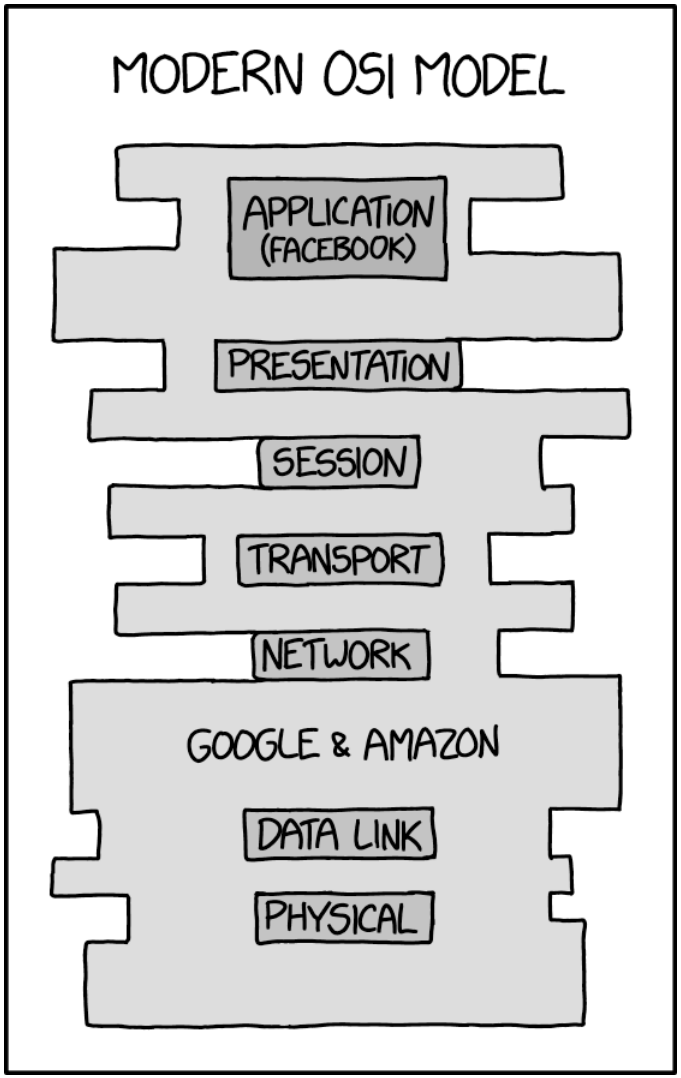
`curl` : copy a URL

`ping` : send IP packets to a machine

- Useful to check connectivity between machines

`traceroute` : determine a route to a machine

An appropriate xkcd comic: http://xkcd.com/2105



End of lecture