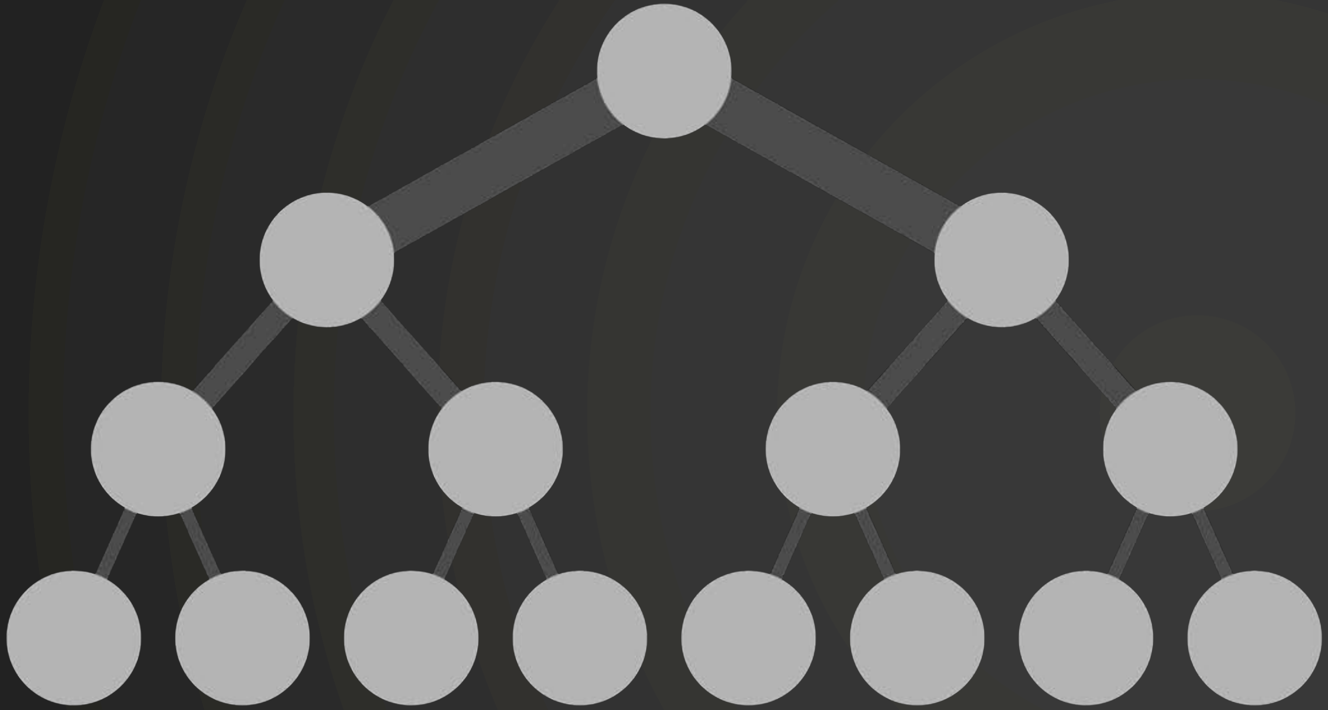
Los Alamos

NATIONAL LABORATORY

EST. 1943

NNSA

National Nuclear Security Administration

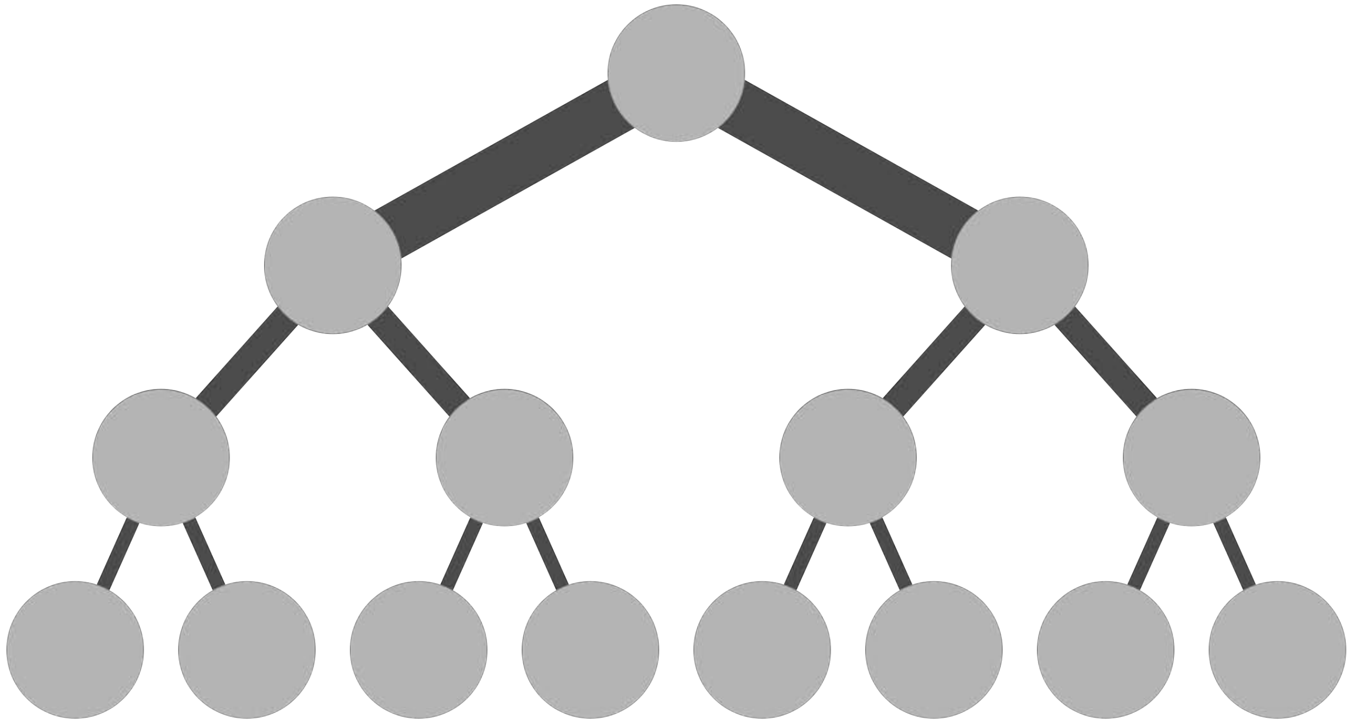


Los Alamos
NATIONAL LABORATORY
EST. 1943

Delivering science and technology
to protect our nation
and promote world stability

HPC Networking and Services

Presented by CSCNSI



Agenda

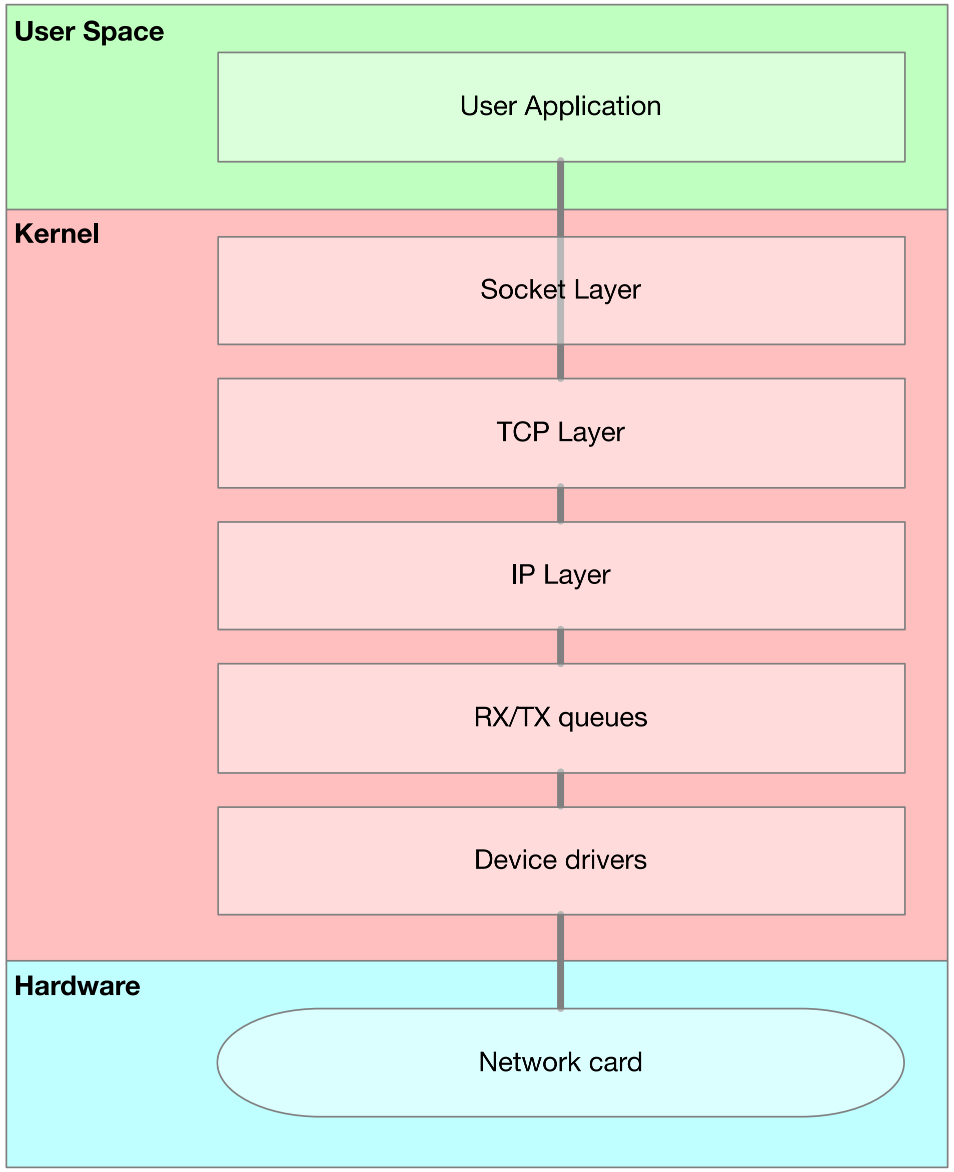
- Networking Overview
- Services

Networking Overview

- Computers need to talk to each other. They do this over networks made up of communication hardware
- HPC clusters frequently have multiple networks
  - High Speed Network (HSN): special low-latency, high speed network for job communication
  - BMC network: Standard management network for baseboard management controllers (BMCs)
  - Cluster network: Standard network used for general node-to-node communication (ssh between nodes; communication with scheduler; etc.)
- This section looks at the standard networks; HSNs will be covered later
- Your cluster uses 1000Mbit ethernet as its cluster network

HPC Networking

- Networks are complex
- They may consist of…
  - Multiple hardware types
    - (wireless, wired, optical, …)
  - Multiple vendors
    - (Linksys, Arista, Broadcom, Unifi, …)
  - Multiple protocols
    - (HTTP, Bittorrent, SSH, …)
  - Multiple speeds
    - (54Mbit, 100Mbit, 1000Mbit, …)
- These details are hidden beneath libraries, standards, and layers of abstraction



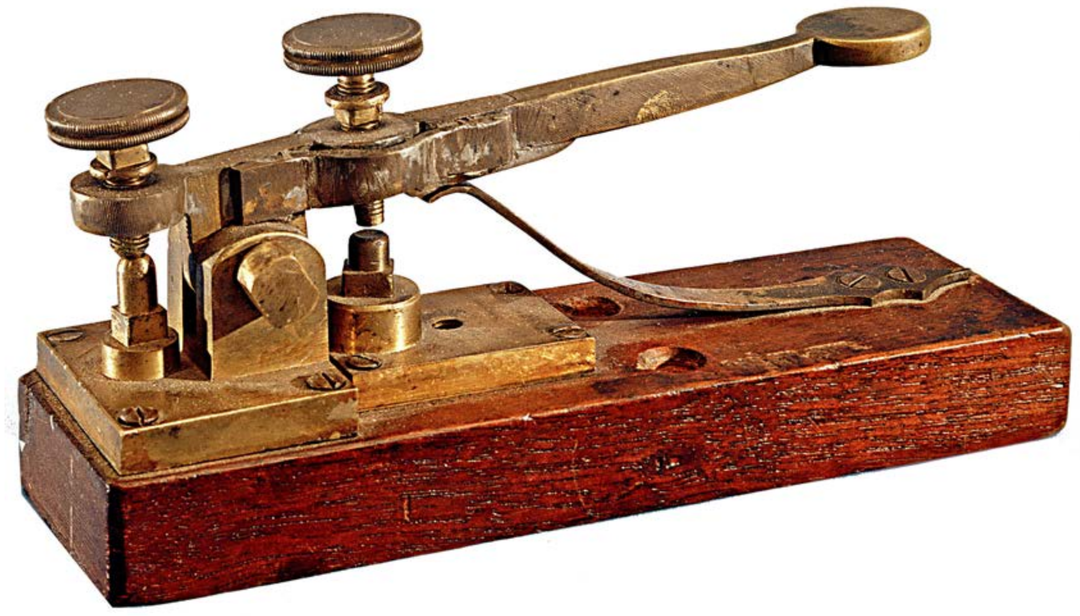
Networking Theory

- The seven-layer Open Systems Interconnection (OSI) model
  - Logically splits an application's view of a network into interoperable pieces
  - In theory, any layer can be replaced without affecting the others
  - In reality, some layers are combined and some have complex interactions depending on the network implementation
- *For now, we're concerned about layers 1 through 4.*

7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

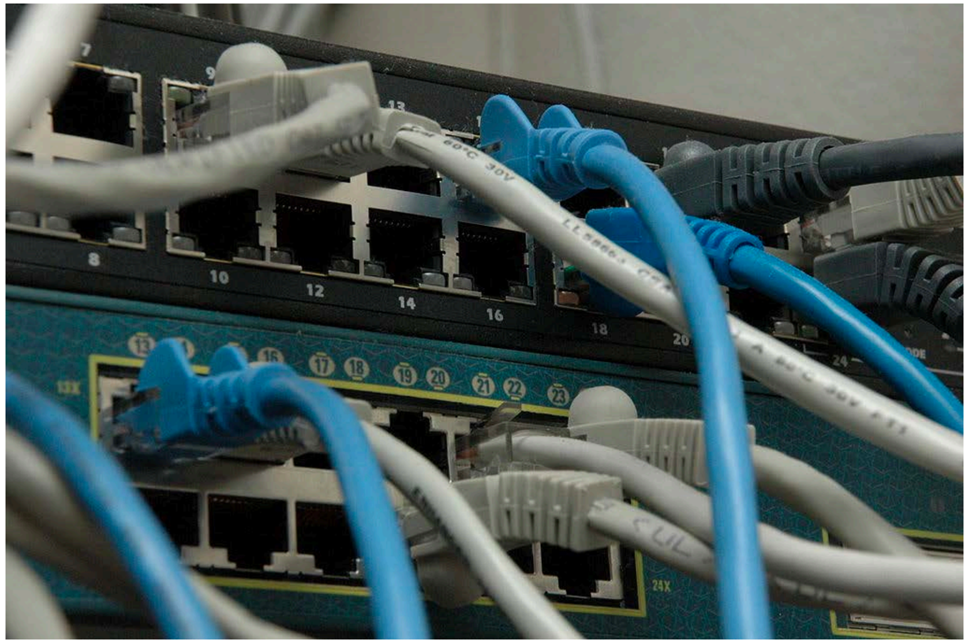
Layer 1: The Physical Layer

- The most basic communication layer
- Enough for one device to send a signal to another
  - Electrical signals flowing over copper wires
  - Light signals transmitted via fiberoptic cables
  - Wireless signals transmitted via radio waves
- Just a series of ones and zeros
- Bits are merely passed from one device to the next. No interpretation of the bits is performed at this level.

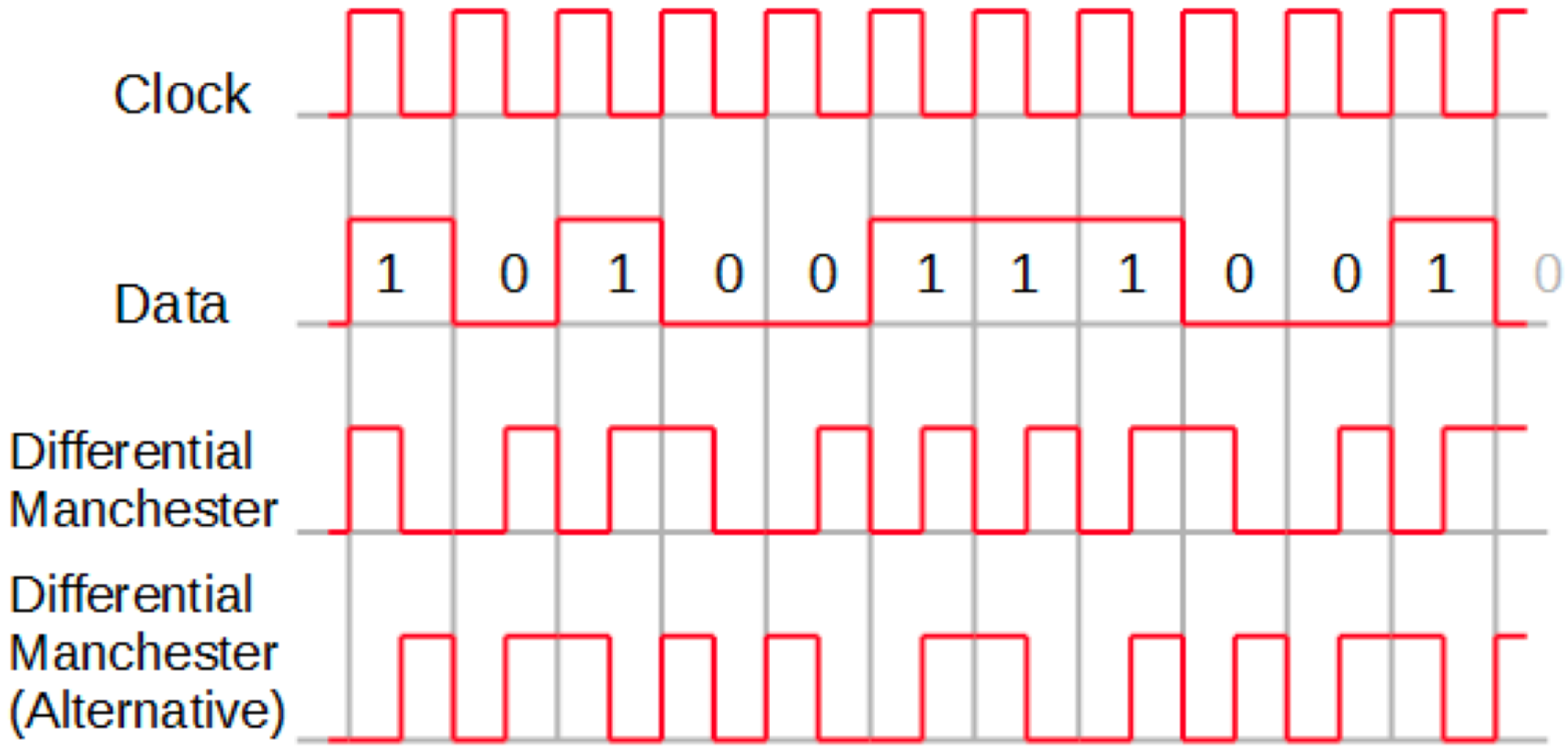


A Layer 1 Interface: Twisted Pair

- An extremely common physical network
- Cables consist of four pairs of individual copper wires that are twisted together
- Ends consist of RJ45 connectors, which look like big phone connectors
- Cables plug into hardware ports on each end: one on the computer, one on the switch
- Basic unit of transfer is a single bit, indicated by high or low voltage on the wire

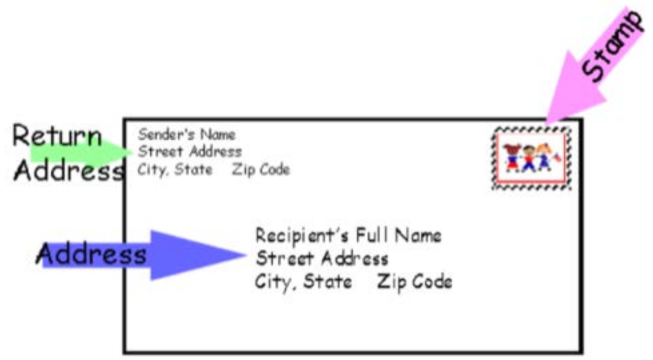


Layer 1 Details: Encoding



https://commons.wikimedia.org/wiki/File:Differential_Manchester_encoding_alternatives.png

Layer 2: The Data Link Layer



- The physical layer is too low-level to use directly
  - It’s just high and low voltage on a wire
  - It only works on one wire
- The Data Link layer builds a protocol on top of the Physical Layer for moving information between endpoints
- At this level, the bits can be examined by intermediary hardware to do things like check for errors, look at destination addresses, and provide quality of service (i.e. choose who gets priority when things are busy)

A Layer 2 Protocol: Ethernet

- An extremely common Layer 2 local-area network standard
- Allows one computer to communicate to another, potentially passing through one or more switches on the way
- Basic unit of transfer is a *frame*, which consists of a set of bits
- Every endpoint has a globally unique Medium Access Control (MAC) address
  - 48-bit address in the firmware of every ethernet card
  - Example: EC:F4:BB:C6:23:E4
- Communication involves broadcasting to find what switch hardware port a particular MAC address is on, and then forwarding frames that way

- Efficient for local connections; very inefficient for wide-area networks
  - You can’t broadcast to the entire internet to find a particular MAC address

Layer 2: The Ethernet Frame

	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Preamble																															
4	32	Preamble, continued																								Start of Frame Delimiter							
8	64	Destination MAC Address																															
12	96	Destination MAC Address, continued																Source MAC Address															
16	128	Source MAC Address, continued																															
20	160	802.1Q Tag																															
24	192	Length																Payload															
…	…	Payload, continued																															
X	X	Frame Sequence Check																															

Layer 3: The Network Layer

- The Data Link layer is still too low-level to use directly
  - It would be essentially impossible to find a MAC address on the other side of the world (or even on the other side of town)
- The Network layer introduces network segments that can be routed between
  - MAC addresses can still be found on small local broadcast domains
  - Anything that can’t be reached locally gets sent out of the local network

A Layer 3 Protocol: IPv4

- Internet Protocol: A “best effort” protocol built on top of the Data Link layer
  - No guaranteed data delivery; no guaranteed data ordering; possible duplicate data delivery
- Basic unit of transfer is a *packet*, which is encapsulated within a frame
- Packets are sent between *IP addresses*
  - Four 8-bit octets: 0.0.0.0 to 255.255.255.255
  - Unlike MAC addresses, IP addresses can be used to route packets over long distances
- IP Addresses are hierarchical, which helps the routing process
  - 192.x.x.x
  - 192.168.x.x
  - 192.168.1.x
- For scalability, addresses are grouped into *subnets*

Layer 3 Details: IPv4 Subnets

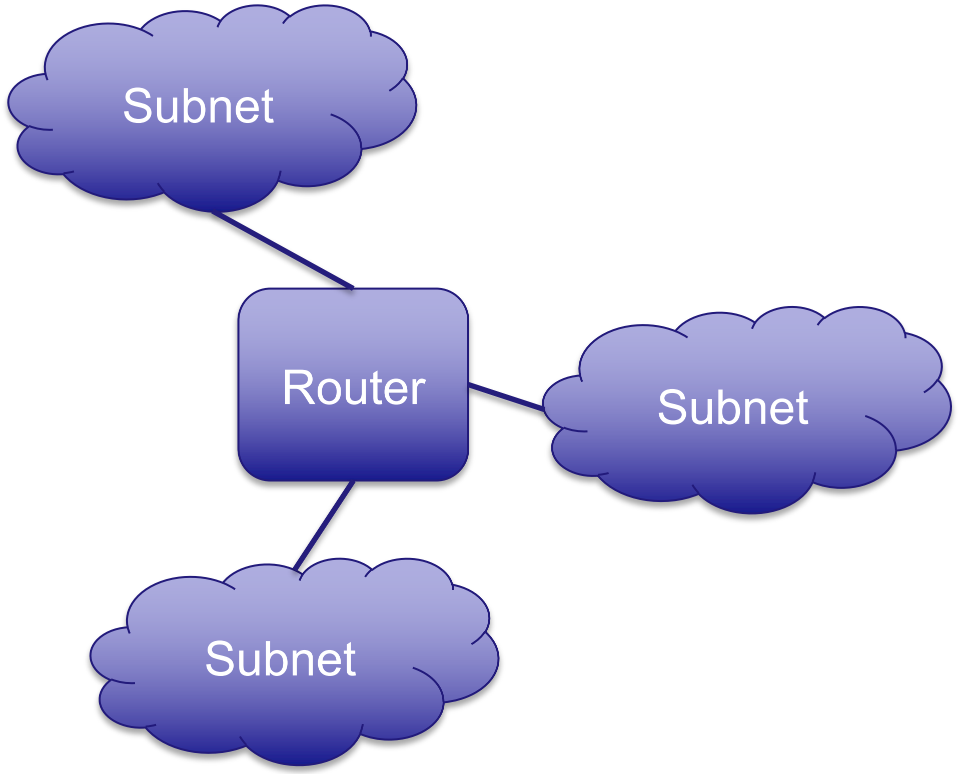
- Logical groupings of addresses that make networking more efficient
  - Within a subnet, local hardware and software handles packet delivery
  - Between subnets, routing needs to happen
- Defined by an address (192.168.0.0) and a mask (255.255.0.0)
  - Alternate format: 192.168.0.0/16
- All addresses on a subnet have the same routing prefix
  - 192.168.1.1, 192.168.1.2, 192.168.2.10 are all on 192.168.0.0/16
  - 10.0.0.1, 192.168.1.1, 172.16.2.1 are all on different subnets
- All subnets have a *broadcast address*
  - Packets sent to this special address get sent to all other addresses on the subnet
  - Implemented as the last address in a subnet
    - Examples: 10.255.255.255, 192.168.255.255

Layer 3: The IPv4 Header

	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset												
8	64	Time to Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options																															
24	192																																
28	226																																
32	256																																

IPv4 Routing

- IPv4 traffic can address the whole world by introducing *routers*
- Routers keep track tables of how to get to different subnets
  - A special entry in the table, the *default,* or *0.0.0.0/0*, entry becomes a bucket for any address that isn't specified by the table
- Building and maintaining routing tables can be complicated
  - Another example of a Layer 3 protocol is OSPF, a protocol used to build these tables dynamically



Layer 4: The Transport Layer

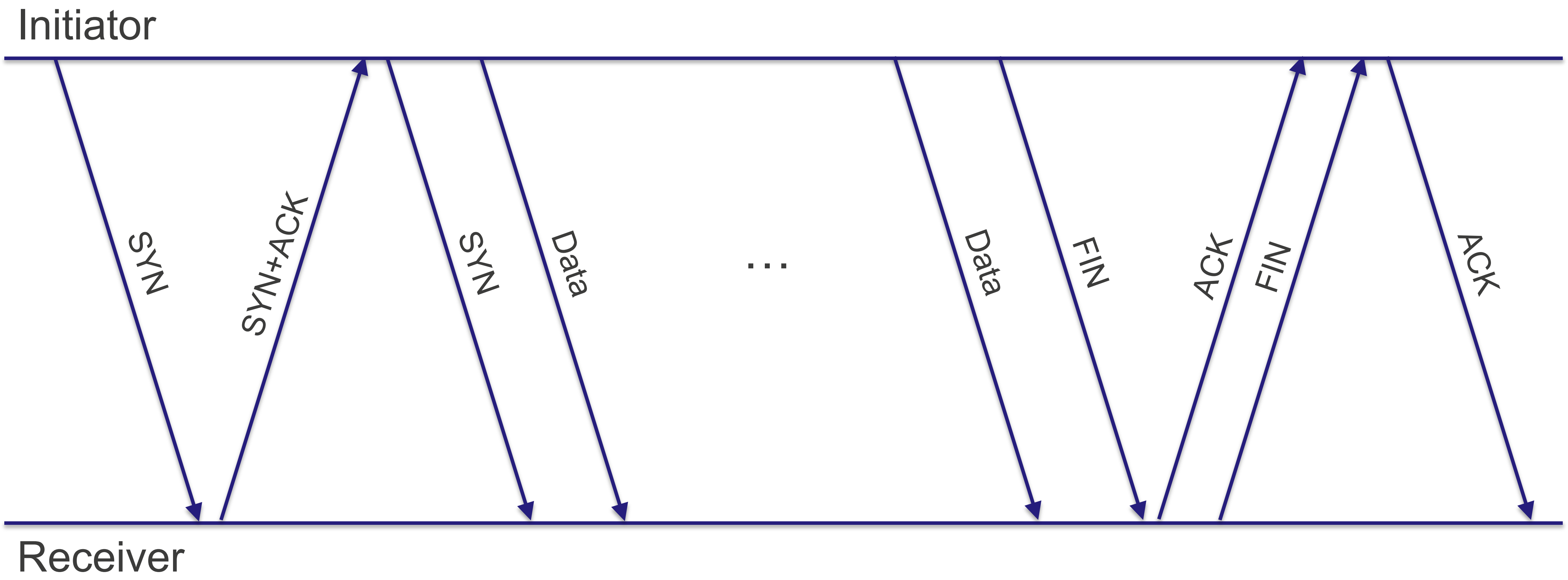
United States Postal Service®
CERTIFIED MAIL®
FOR USE ONLY WITH IMpb SHIPPING LABEL
Label 3800-N, January 2014
PSN 7690-17-000-0906

- The Network layer is *still* too low-level to use directly
  - Only one application can use it at a time
    - …how would we keep track of more than one address on one machine?
  - It is only best-effort
    - …we have no guarantee that data is delivered.
- The Transport layer adds the remaining functionality that applications need
  - Persistent connections
  - Multiplexing
  - … and others

A Layer 4 Protocol: TCP

- Transmission Control Protocol: a robust protocol built on top of the Network layer
  - Guarantees packet delivery, guarantees packets are received in order, guarantees at-most-once delivery of packets
  - This adds overhead, but also provides very useful functionality
- Basic unit of transfer is a *segment*, which is encapsulated within a packet
- TCP introduces *software ports*
  - Numbered between 1 and 65535
  - A port can be open by one application at a time
- TCP is connection oriented:
  - One computer connects to another via a unique IP address and port combination
  - This connection stays open until explicitly closed down (or until it times out)

Layer 4 Details: TCP Connection Lifecycle



Layer 4: The TCP Header

	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source Port																Destination Port															
4	32	Sequence Number																															
8	64	Acknowledgement Number																															
12	96	Data Offset				Reserved			Flags									Window Size															
16	128	Checksum																Urgent Pointer															
20	160	Options																															
…	…																																

Another Layer 4 Protocol: UDP

- User Datagram Protocol: a less robust protocol built on top of the Network layer
  - No guaranteed data delivery; no guaranteed data ordering; possible duplicate data delivery
- But! Provides ports and checksums, just like TCP
- Provides some of the benefits of TCP without all of the overhead
- Good for:
  - Applications that can tolerate missing packets
  - Communication that would be dramatically slowed down by the full TCP connection process
  - Protocols that use simple, singleton messages

Layer 4: The UDP Header

	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source Port																Destination Port															
4	32	Length																Checksum															

Layers 1 Through 4 Working Together

- Each of these protocols encapsulates the data above it
  - An IP packet’s data section is an entire TCP segment: header plus data
  - An Ethernet frame’s data section is an entire IP packet
- As the data moves through the stack, a layer either adds its header and passes the data down or removes its header and passes the data up

TCP Segment			Header	Data	
IP Packet		Header	Data		
Ethernet Frame	Header	Data			Footer

Practical Networking

- On a real computer, the kernel maps physical *hardware ports* to virtual *interfaces*
  - Example: The leftmost hardware port is named `em1`
- Interfaces are associated with IP addresses
  - Example: Interface `em1` has address 192.168.1.1

- Interfaces can be manipulated using the `ip` command
  - `ip addr show`
- Interface connectivity can be tested using the `ping` command
  - `ping 192.168.1.2`

What is a cluster network useful for?

- Network booting: transferring an initial image from a master node to a compute node
- Remote login: using SSH to open a command line on another system
- Scheduling: submitting a compute job to a queue that will be run on part or all of the cluster
- Remote logging: sending logs from compute nodes to a master node
- And many other tasks

- All of these tasks are enabled by *services*

Services

Services

- Services are long-running processes that (usually) listen on a TCP or UDP port and perform actions based on incoming connections from other systems
  - These processes are referred to as “daemons”
  - Unlike most processes that you start in your shell, daemons are run in the background and are not associated with a login process
- Examples
  - `sshd`: provides remote login capabilities for a system
  - `httpd`: serves up web pages from a system
  - `mysqld`: provides a database on a system

Services

- Service processes listen for connections on an IP address and a well-defined software port
  - HTTP: port 80
  - HTTPS: port 443
  - SSH: port 22
- Ports assigned to standard services are listed in /etc/services
- Service protocols are generally built on top of TCP or UDP
  - The choice often depends on and latency and resiliency requirements
- Services on networks are sometimes referred to by their *address:port* combination
  - 192.168.1.1:80

Services

- Services you will likely see on an HPC cluster include:
  - SSH – Port 22, TCP; enables remote access
  - DNS – Port 53, UDP; Provides host name lookup
  - DHCP – Port 67, UDP; hands out IP addresses
  - NTP – Port 123, UDP; synchronizes clocks
  - Syslog – Port 514, UDP; collects system logs
  - Slurm – Port 6817, TCP; provides scheduling services

systemd

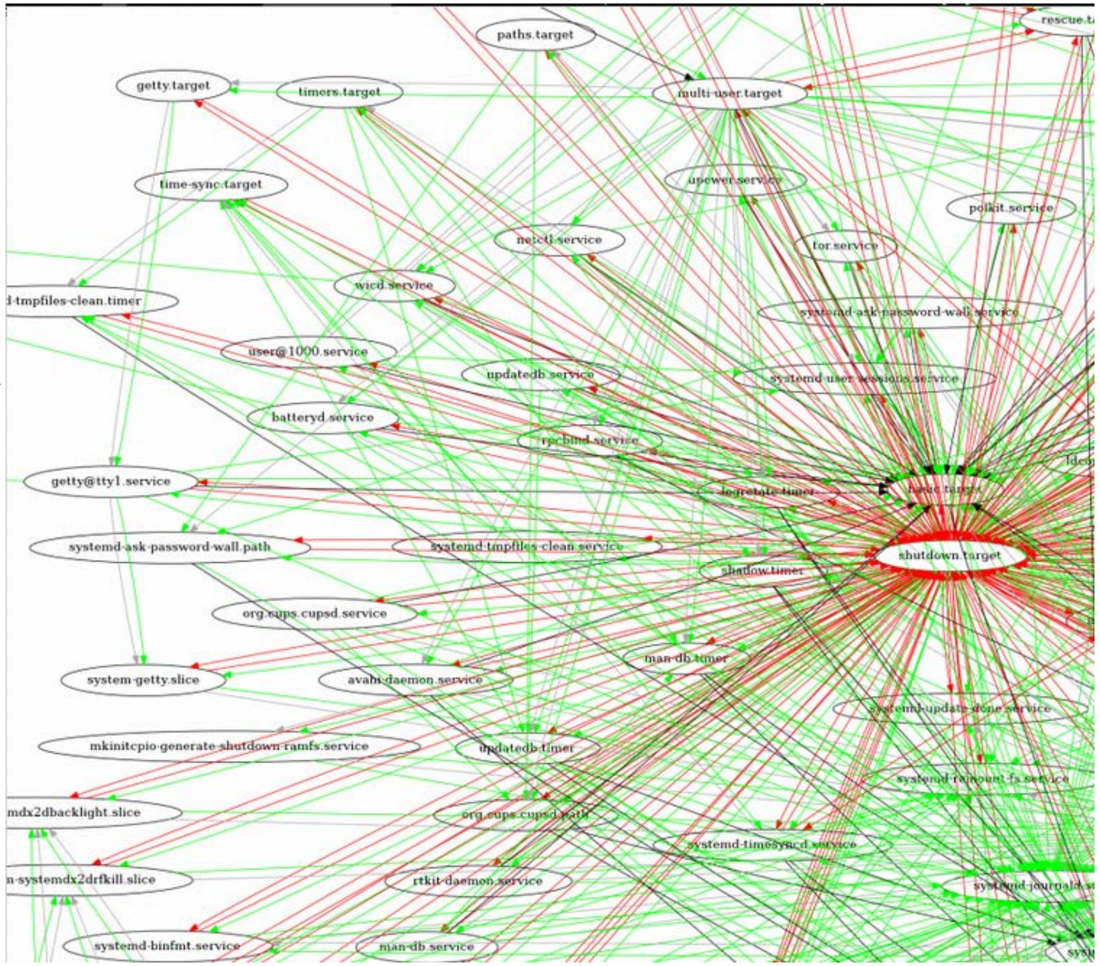
- systemd is the init system used on most modern Linux distributions
- On OS bootup, systemd performs initial service startup
  - Starts all of the service processes and puts them in the background
- After boot, systemd manages running services
  - Can monitor and restart services when they die
  - Provides tools for interacting with running or stopped services
- The base systemd object is the *unit*
- Units can be…
  - services
  - target levels
  - mountpoints
  - … and a variety of other things

systemd Unit File

```
# /usr/lib/systemd/system/ntpd.service

[Unit]
Description=Network Time Service
After=syslog.target ntpdate.service sntp.service

[Service]
Type=forking
EnvironmentFile=-/etc/sysconfig/ntpd
ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Systemd Dependencies

- Most traditional service managers start services in a prescribed sequence
- Systemd, instead, can start/stop services in parallel
    - It does this by collecting all of its *units* together and making a directed graph of dependencies
    - Any process for which the dependencies have been satisfied can start/stop
        - …no prescribed sequence required
- But, this can lead to headaches like race conditions that are hard to diagnose



systemctl – The Main systemd Interface

- Start a service
  - `> systemctl start <servicename>`
- Stop a running service
  - `> systemctl stop <servicename>`
- Restart a running service
  - `> systemctl restart <servicename>`
- Check up on a service (running or not)
  - `> systemctl status <servicename>`
- List all units that systemd knows about
  - `> systemctl list-units`

Questions?