Network Programming

Course Goal:

- Understand the basic principles of computer networks
  - Network basic
  - Basic design principles in network protocols
  - Internet protocols

- Study the programming aspects of computer networks
  - Socket programming
  - Inter-process communication

What is Network Programming?

- ***Network Programming*** *involves writing programs that communicate with other programs across a computer network.*

I. Basic Network Concepts

I.1 Networks

- Network
- Node
- Address
- Packet
- Protocol

Network

- A network is a collection of computers and other devices that can send data to and receive data from each other.

- A network is often connected by wires.

- However, wireless networks transmit data through infrared light and microwaves.

Node

- Each machine on a network is called a node.

- Most nodes are computers, but printers, routers, bridges, gateways, dumb terminals.

Address

- Every network node has an address, a series of bytes that uniquely identify it.

- The more bytes there are in each address, the more addresses there are available and the more devices that can be connected to the network simultaneously.

Packet

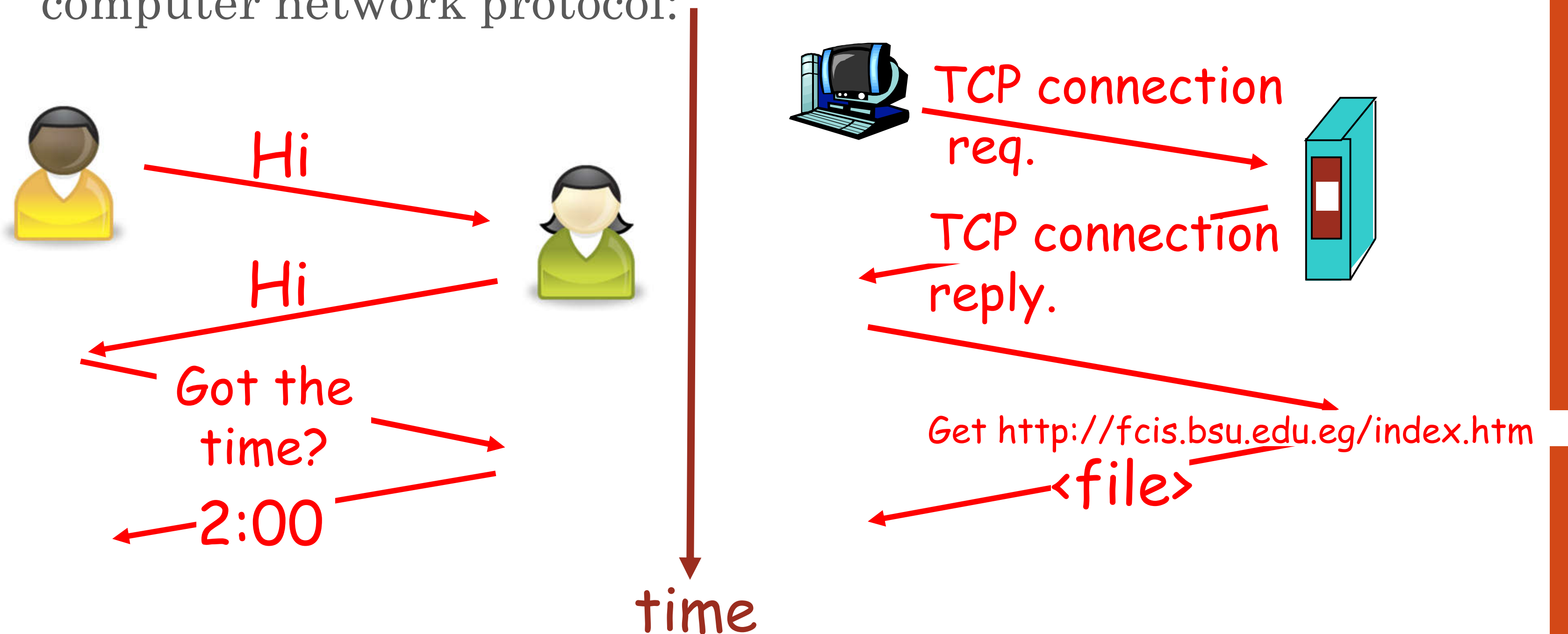
- All modern computer networks are packet-switched networks: data traveling on the network is broken into chunks called packets and each packet is handled separately.

- Each packet contains information about who sent it and where it's going.

Network Protocols

What's a protocol?

- a human protocol and a computer network protocol:



protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt

Organization of air travel

ticket (purchase)	ticket (complain)
baggage (check)	baggage (claim)
gates (load)	gates (unload)
runway takeoff	runway landing
airplane routing	airplane routing

airplane routing

Although this course is about network programming (and not about networking in general), an understanding of a complete network model is essential.

Organization of air travel: a different view

ticket (purchase)	ticket (complain)
baggage (check)	baggage (claim)
gates (load)	gates (unload)
runway takeoff	runway landing
airplane routing	airplane routing
airplane routing	

Layers: each layer implements a service

- via its own internal-layer actions
- relying on services provided by layer below

Distributed implementation of layer functionality

Departing airport

ticket (purchase)
baggage (check)
gates (load)
runway takeoff
airplane routing

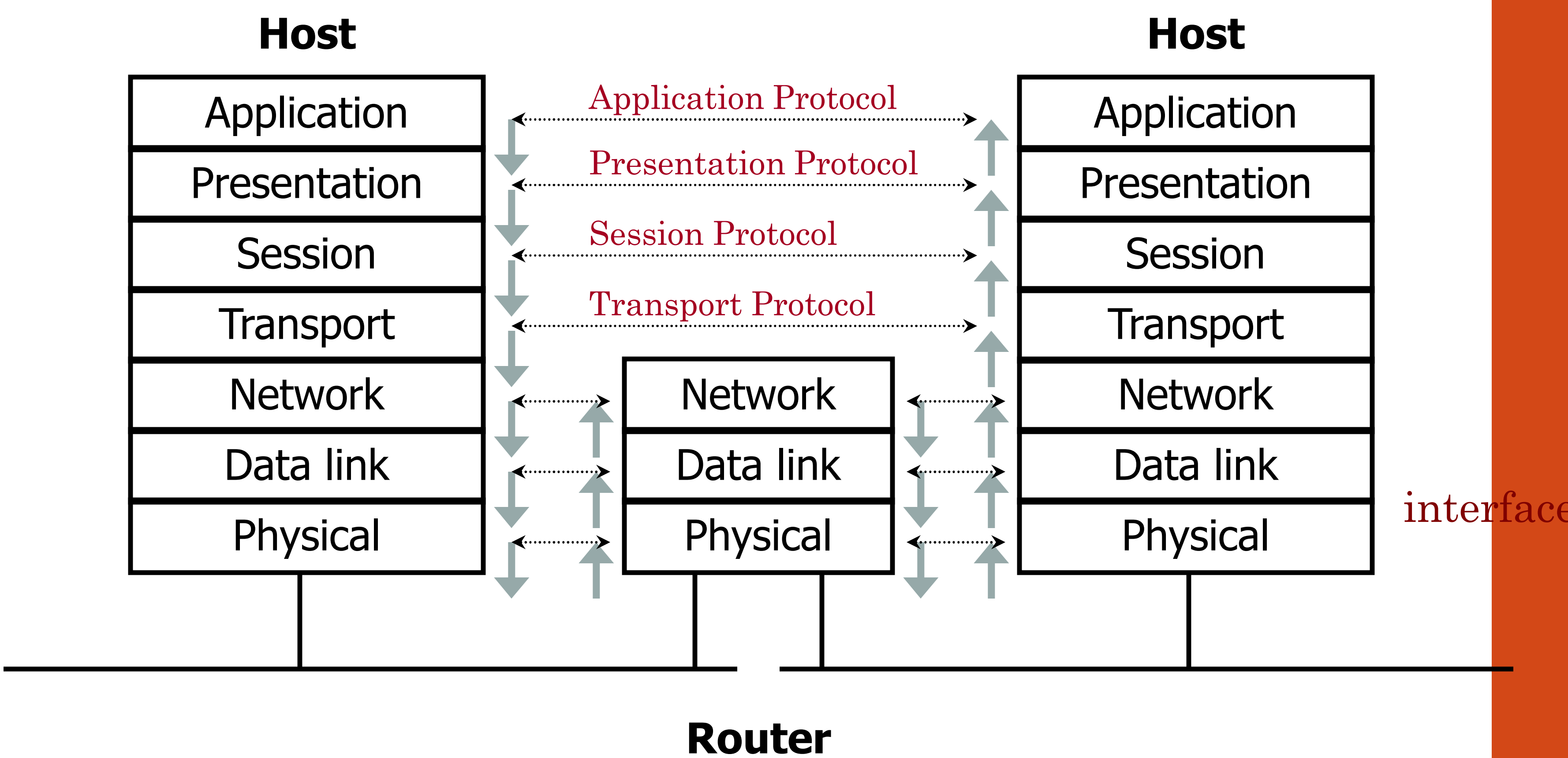
arriving airport

ticket (complain)
baggage (claim)
gates (unload)
runway landing
airplane routing

intermediate air traffic sites

airplane routing
airplane routing
airplane routing

Communicating between End Hosts



Why layering?

- Divide a task into pieces and then solve each piece independently (or nearly so).

- Establishing a well defined interface between layers makes porting easier.

- Functions of each layer are independent of functions of other layers
  - Thus each layer is like a module and can be developed independently

- Each layer builds on services provided by lower layers
  - Thus no need to worry about details of lower layers -- *transparent* to this layer

- Major Advantages:
  - Code Reuse
  - Eases maintenance, updating of system

Programs & Processes

- A *program* is an executable file.
- A *process* or *task* is an instance of a program that is being executed.
- A single program can generate multiple processes.

Client - Server

- A *server* is a process - not a machine !
- A server waits for a request from a client.
- A client is a process that sends a request to an existing server and (usually) waits for a reply.

Client - Server Examples

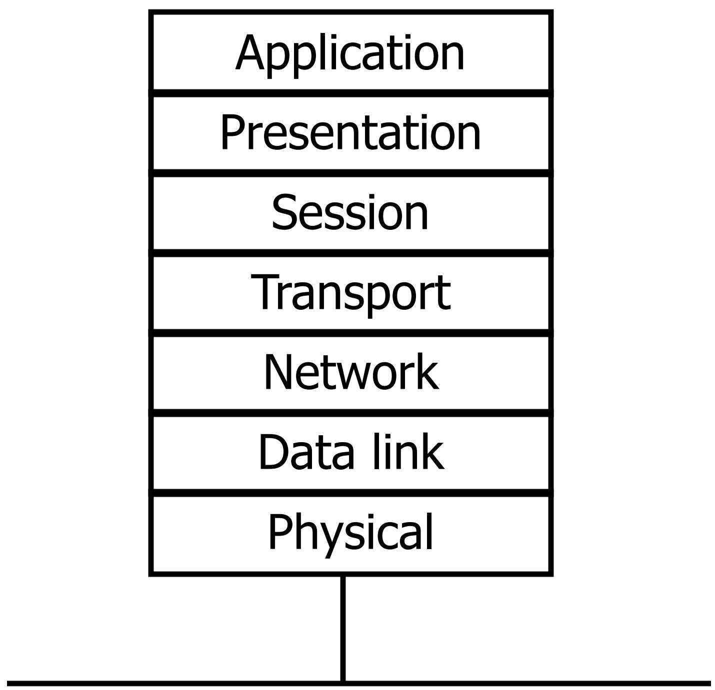
- Server returns the time-of-day.
- Server returns a document.
- Server prints a file for client.
- Server does a disk read or write.
- Server records a transaction.

Sample questions:

- Difference?
  - Subnet, a network, WAN
  - Protocols vs. interface

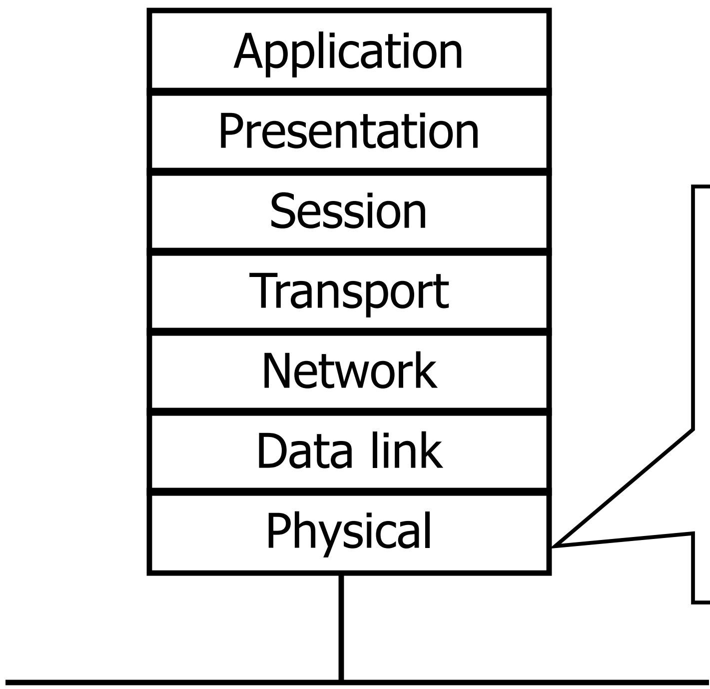
Computer Network Programming
OSI Models & Data link layer

Protocol Stack: ISO OSI Model

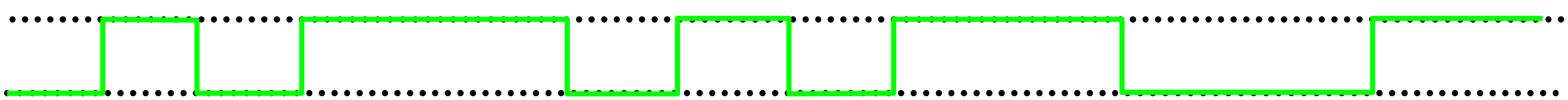


ISO: the International Standards Organization
OSI: Open Systems Interconnection Reference Model (1984)

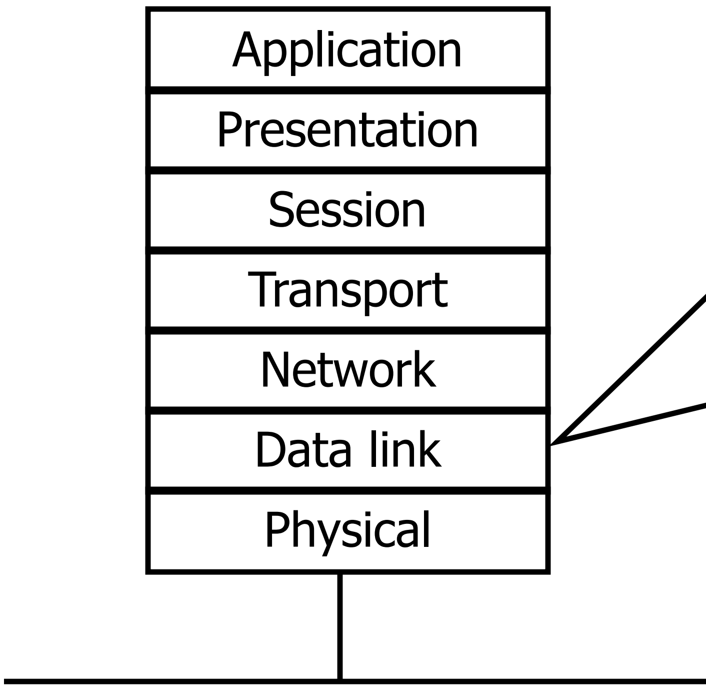
Layer 1: Physical Layer



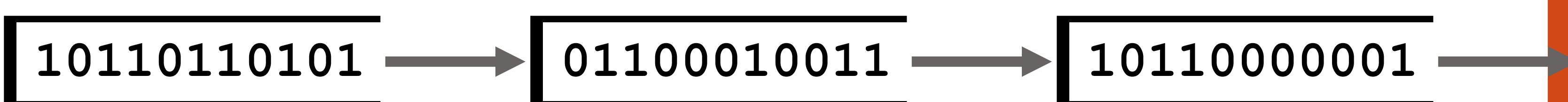
- Responsibilities:
  - Transmission of a raw bit stream
  - Forms the physical interface between devices
- Issues:
  - mechanical and electrical interfaces
  - time per bit
  - distances



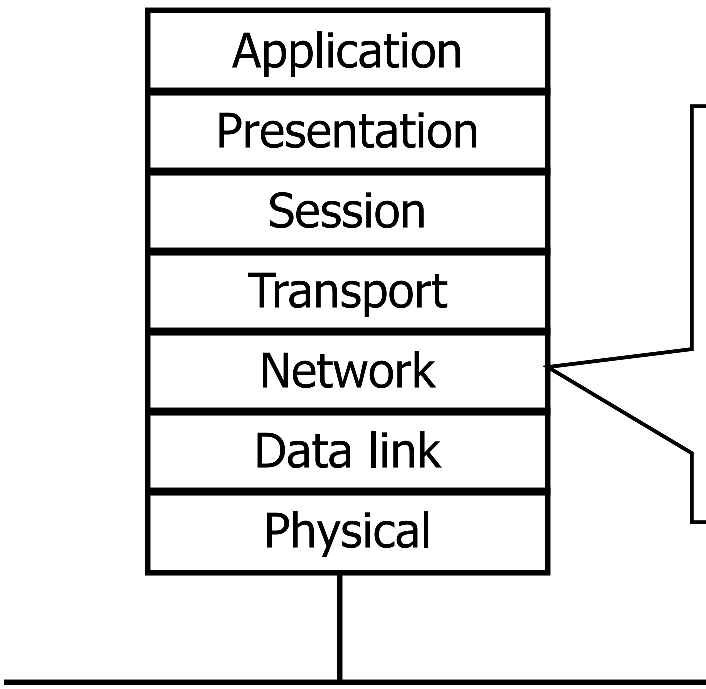
Layer 2: Data Link Layer



- Responsibilities:
  - Provides reliable transfer of information between two <u>adjacent</u> nodes
  - Provides frame-level error control
  - Provides flow control
- Issues:
  - *framing* (dividing data into chunks)
    - header & trailer bits
  - addressing

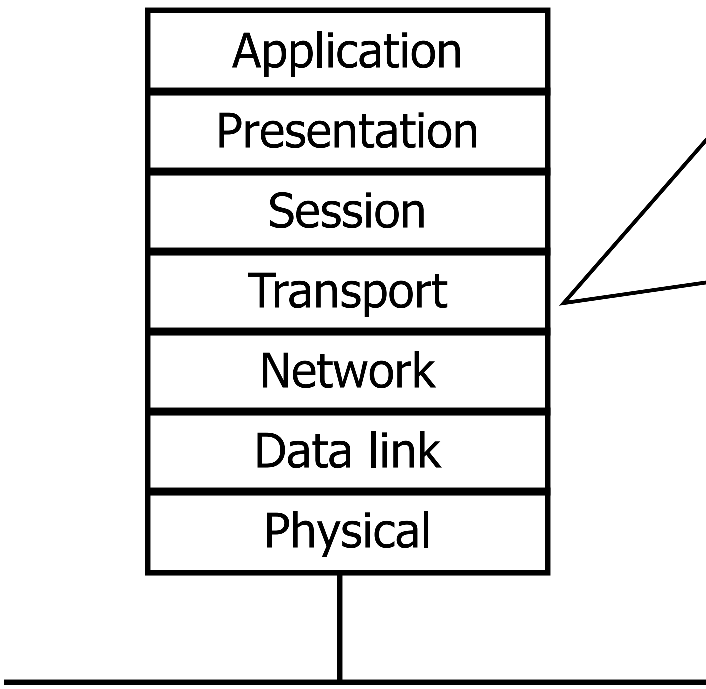


Layer 3: Network Layer



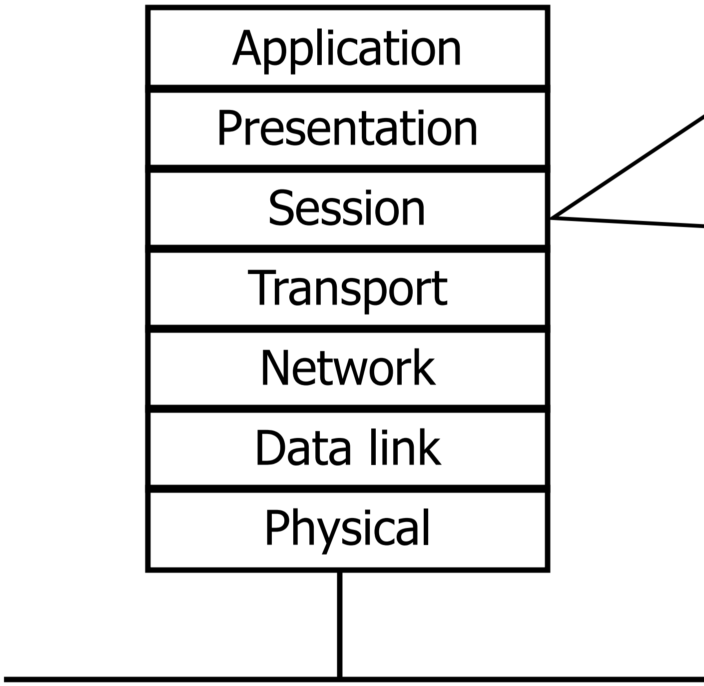
- Responsibilities:
  - path selection between end-systems (routing).
    - Dynamic routing
    - Fixed routing
  - fragmentation & reassembly
  - translation between different network types

Layer 4: Transport Layer



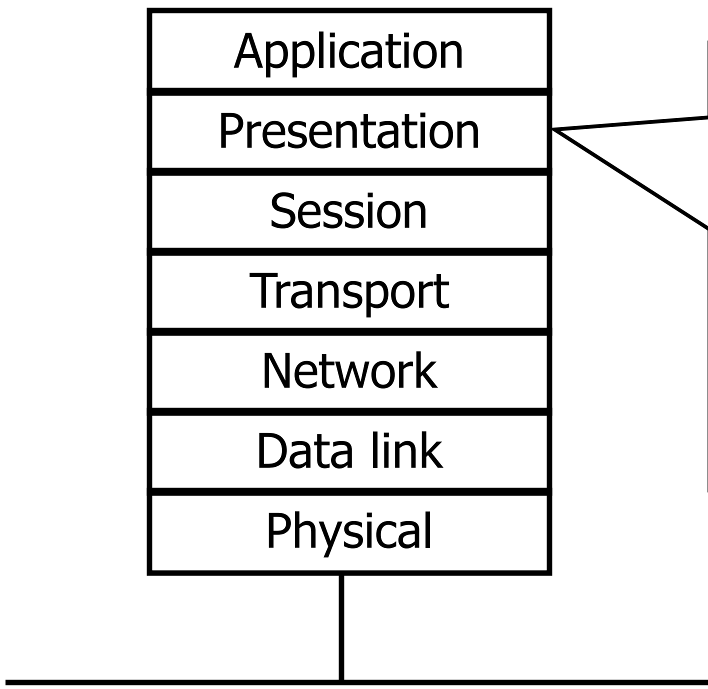
- Responsibilities:
  - provides virtual end-to-end links between peer processes.
  - end-to-end flow control
- Issues:
  - headers
  - error detection
  - reliable communication

Layer 5: Session Layer



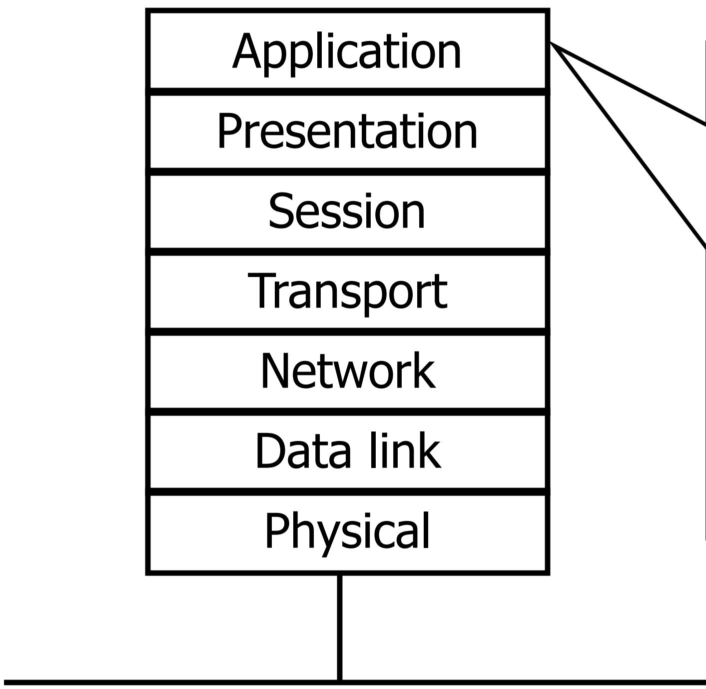
- Responsibilities:
  - Establishes, manages, and terminates a communication session with remote systems
  - Groups several user-level connections into a single “session”
- Many protocol suites do not include a Session Layer.

Layer 6: Presentation Layer



- Responsibilities:
  - Represents data properly
  - Data encryption
  - Data compression
  - Data conversion
- Many protocol suites do not include a Presentation Layer.

Layer 7: Application Layer

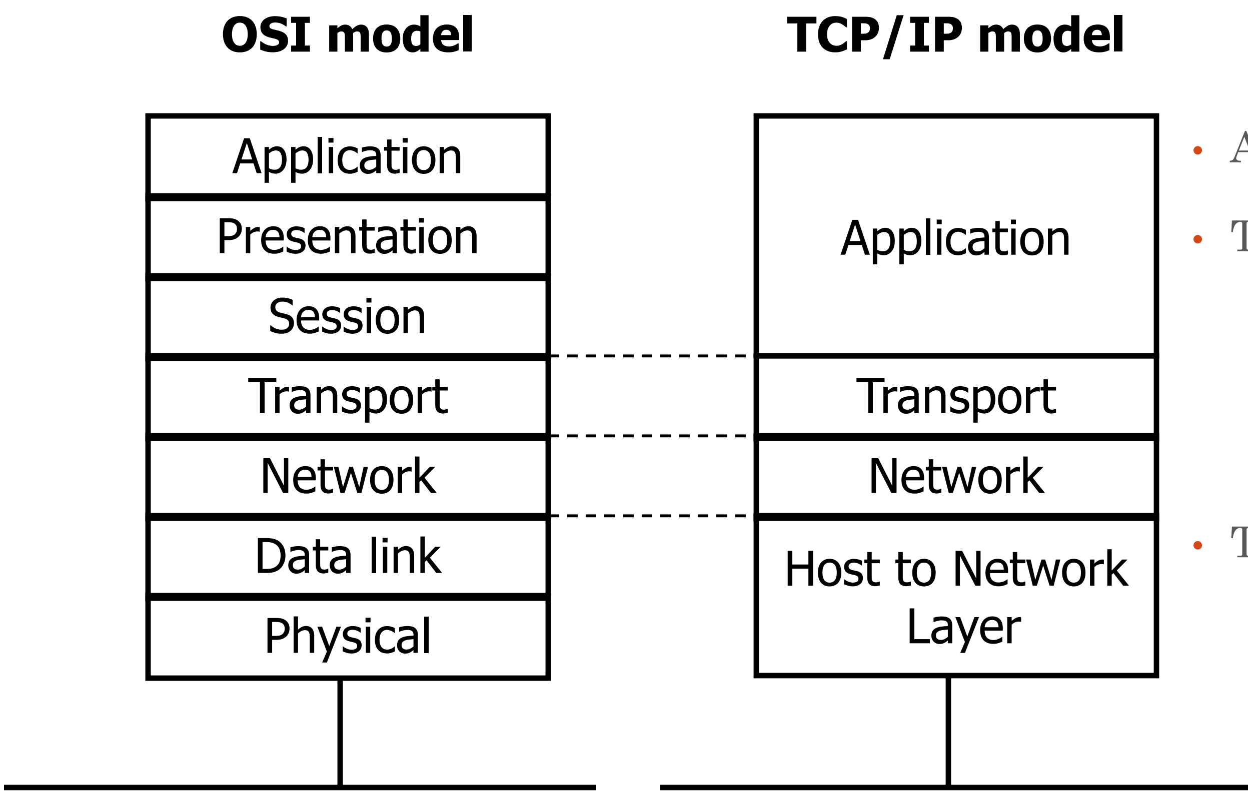


- Responsibilities:
  - Anything not provided by any of the other layers
  - Implements communication between two applications of the same type
    - Examples:
      - FTP
      - HTTP
      - SMTP/POP3/IMAP (email)

Problems

- Seven layers not widely accepted
- Standardized before implemented
- Top three layers fuzzy
- Internet or TCP/IP layering widespread

TCP/IP Layering Architecture



- A simplified model
- The network layer
  - Hosts drop packets into this layer, layer routes towards destination- only promise- try my best
- The transport layer
  - Reliable/unreliable byte oriented stream

Layering & Headers

- Each layer needs to add some control information to the data in order to do it's job.
- This information is typically pre-appended to the data before being given to the lower layer.
- Once the lower layers deliver the data and control information - the peer layer uses the control information.

What are the headers?

Physical:

no header - just a bunch of bits.

Data Link:

- address of the receiving endpoints
- address of the sending endpoint
- length of the data
- checksum.

Network layer header - examples

- protocol suite version
- type of service
- length of the data
- packet identifier
- fragment number
- time to live
- protocol
- header checksum
- source network address
- destination network address

Important Summary

- Data-Link:
  - communication between machines on the same network.
- Network:
  - communication between machines on possibly different networks.
- Transport:
  - communication between processes (running on machines on possibly different networks).

Addresses

- Each communication endpoint must have an address.
- Consider 2 processes communicating over an internet:
  - the network must be specified
  - the host (end-system) must be specified
  - the process must be specified.

Addresses at Layers

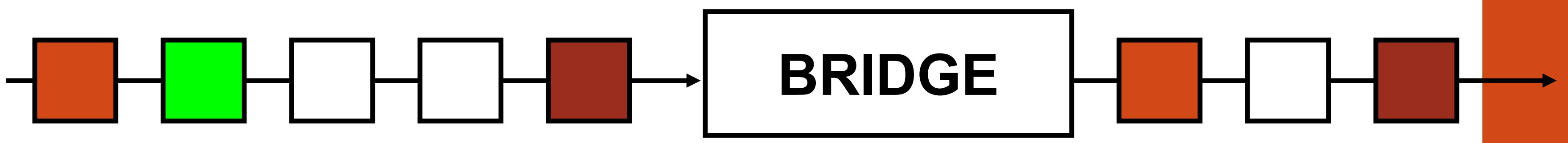
- Physical Layer
  - no address necessary
- Data Link Layer
  - address must be able to select any host on the network.
- Network Layer
  - address must be able to provide information to enable routing.
- Transport Layer
  - address must identify the destination process.

Repeater

- Copies bits from one network to another
- Does not look at any bits
- Allows the extension of a network beyond physical length limitations

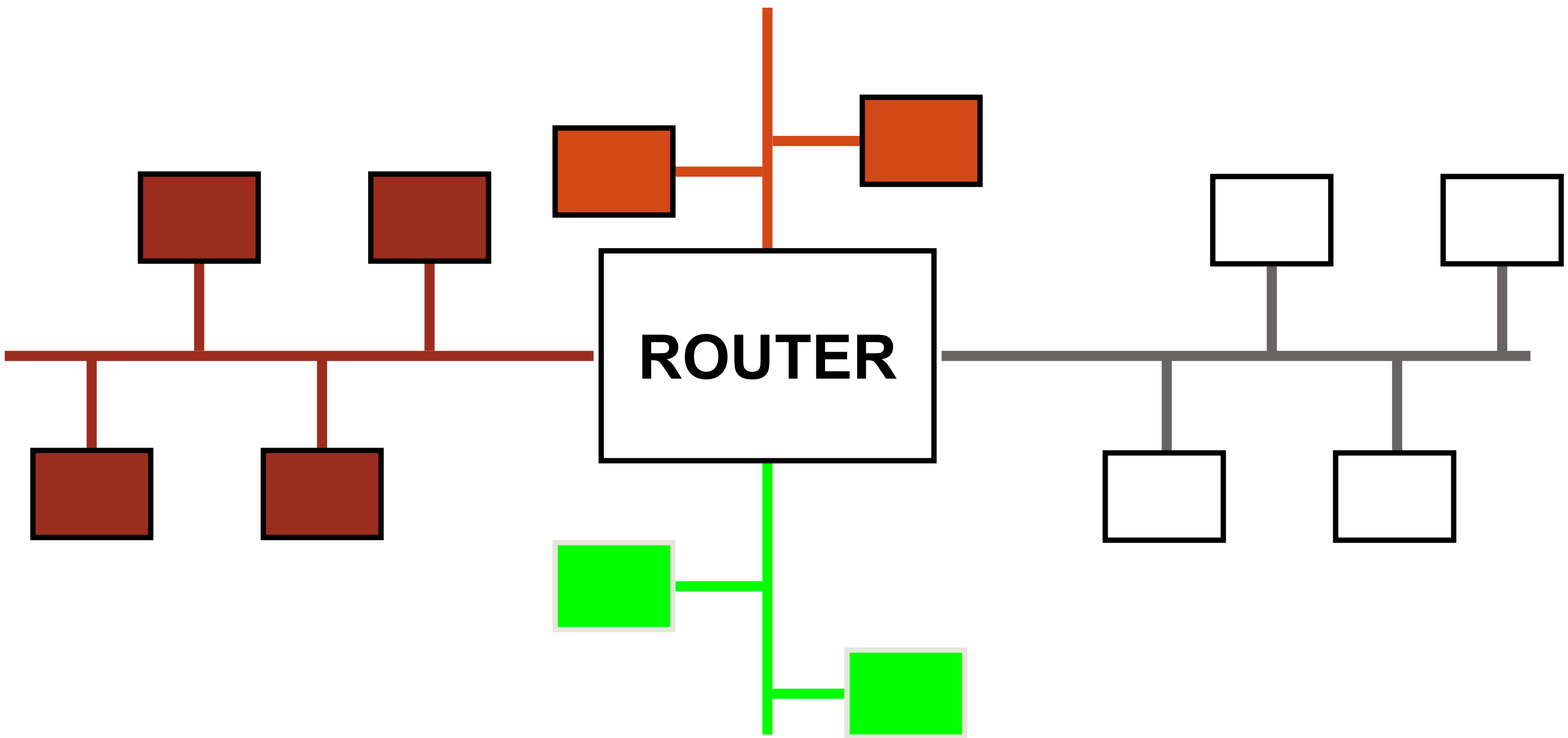
Bridge

- Copies frames from one network to another
- Can operate selectively - does not copy all frames (must look at data-link headers).
- Extends the network beyond physical length limitations.



Router

- Copies packets from one network to another.
- Makes decisions about what *route* a packet should take (looks at network headers).



Gateway

- Operates as a router
- Data conversions above the network layer.
- Conversions:

  encapsulation - use an intermediate network

  translation - connect different application protocols

  encryption - could be done by a gateway

Which layer?

- Repeater & Hub
  - physical layer
- Bridge & Switch
  - data link layer
- Router
  - network layer
- Gateway
  - network layer and above.

Hardware vs. Software

- Repeaters are typically hardware devices.
- Bridges can be implemented in hardware or software.
- Routers & Gateways are typically implemented in software so that they can be extended to handle new protocols.
- Many workstations can operate as routers or gateways.

Data Link Layer Protocol

Date Link Layer Functionality

- Convert bits to signals and recover bits from received signals
  - Encoding
- Decide on a minimum unit for sending bits
  - Frame creation
- Error detection and /or correction of frames
  - Parity, CRC
- Flow control
  - ARQ, Sliding WINDOW

Encoding

- Signals propagate over a physical medium
  - Modulate electromagnetic waves
  - e.g. vary voltage
- Encode binary date onto signals
  - e.g. 0 as low signal and 1 as high signal
  - Known as non-return to zero (NRZ)
  - Non-return to zero inverted (NRZI)
    - Make a transition from current signal to encode a 1; stay at current signal to encode a 0
  - Manchester
    - Transmit xor of the NRZ encoded data and the clock
    - Only 50% efficient

Framing

- The date unit at the date link layer is called a "frame"
- A frame is a group of bits, typically in sequence
- Issues:
  - Frame creation
  - Frame delineation
- Use starting and ending characters (tags) to mark boundaries of frame
  - Problem: what if tag characters occur in the date or control portions of the frame
    - Insert extra escape character when a tag appears in date field

Error Control

- No physical link is perfect
- Bits will be corrupted
- We can either:
  - Detect errors and request retransmission
  - Or correct errors without retransmission

- Error Detection
  - Parity bits
  - Polynomial codes or checksums

Parity bits

- Append a single parity bit to a sequence of bits
- If using 'odd' parity, the parity bit is chosen to make the total number of 1's in the bit sequence odd
- If 'even' parity, the parity bit makes the total number of 1's in the bit sequence even
  - Q: for even parity, what's the parity bit for 00010101?
- Problem: Only detects when there are an odd number of bit errors

Polynomial codes

- Can detect errors on large chunks of data
- Has low overhead
- More robust than parity bit
- Requires the use of a "code polynomial"
  - Example x^2+1
  - Message 1011 -> $1 * x^3 + 0 * x^2 + 1 * x + 1$

    $= x^3 + x + 1$

Cyclic redundancy check

- CRC: Example of a polynomial code
- Procedure:
  - 1. Let *r* be the degree of the code polynomial. Append *r* zero bits to the end of the transmitted bit string. Call the entire bit string *S(x)*
  - 2. Divide *S(x)* by the code polynomial using modulo 2 division.
  - 3. Subtract the remainder from *S(x)* using modulo 2 subtraction.
- The result is the checksummed message

Decoding a CRC

- Procedure
  - 1. Let n be the length of the checksummed message in bits
  - 2. Divide the checksummed message by the code polynomial using modulo 2 division. If the remainder is zero, there is no error detected.

Choosing a CRC polynomial

- The longer the polynomial, the smaller the probability of undetected error

- Common standard polynomials:
  - (1) CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
  - (2) CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - (3) CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$

IP – Network Layer

IP Datagrams

- IP is the network layer
  - packet delivery service (host-to-host).
  - translation between different data-link protocols.

IP Addresses

- IP addresses are not the same as the underlying data-link (MAC) addresses. WHY?
- IP is a network layer - it must be capable of providing communication between hosts on different kinds of networks (different data-link implementations).
- The address must include information about what *network* the receiving host is on. This is what makes routing feasible.

IP Addresses

- IP addresses are *logical* addresses (not physical)
- 32 bits.
  - IP Addresses are usually shown in *dotted decimal* notation: 1.2.3.4
- Includes a network ID and a host ID.
- Every host must have a unique IP address.
- IP addresses are assigned by a central authority (*American Registry for Internet Numbers* for North America).

Network and Host IDs

- A Network ID is assigned to an organization by a global authority.

- Host IDs are assigned locally by a system administrator.

- Both the Network ID and the Host ID are used for routing.

IP Addresses

- IP Addresses are usually shown in *dotted decimal* notation:

1.2.3.4 **00000001 00000010 00000011 00000100**

- cse.sc.edu is 129.252.138.8

  10000001 11111100 10001010 00001000

CSE has a class B network

Host and Network Addresses

- A single network interface is assigned a single IP address called the *host* address.

- A host may have multiple interfaces, and therefore multiple *host* addresses.

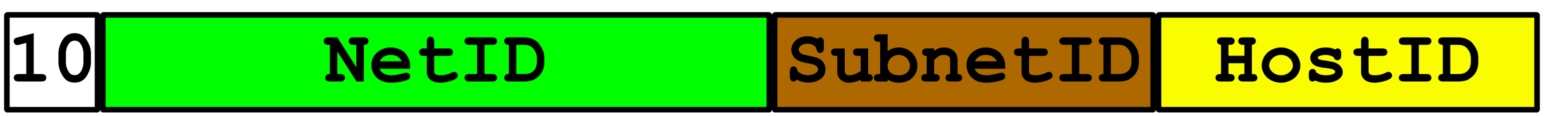
- Hosts that share a network all have the same IP *network* address (the network ID).

Special IP addresses

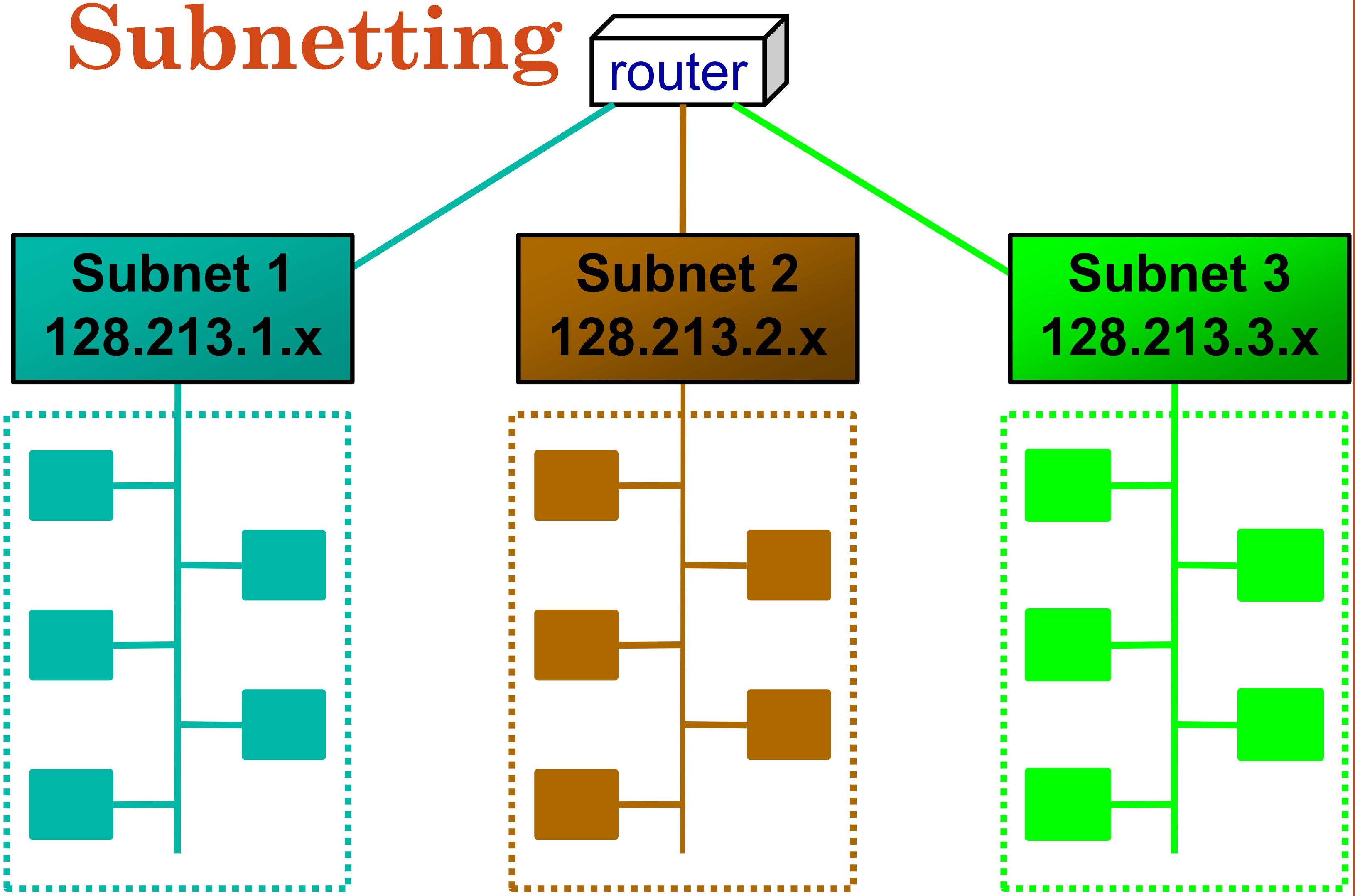
- An IP broadcast addresses has a host ID of all 1s.
- An IP address that has a host ID of all 0s is called a *network address* and refers to an entire network.
- localhost: 127.0.0.1

Subnet Addresses

- An organization can subdivide it's host address space into groups called subnets.
- The subnet ID is generally used to group hosts based on the physical network topology.

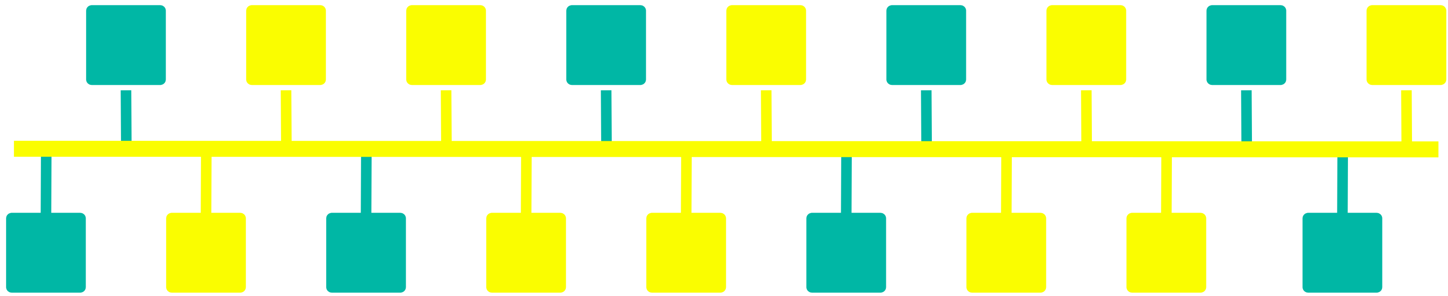


Subnetting


router
Subnet 1
128.213.1.x
Subnet 2
128.213.2.x
Subnet 3
128.213.3.x

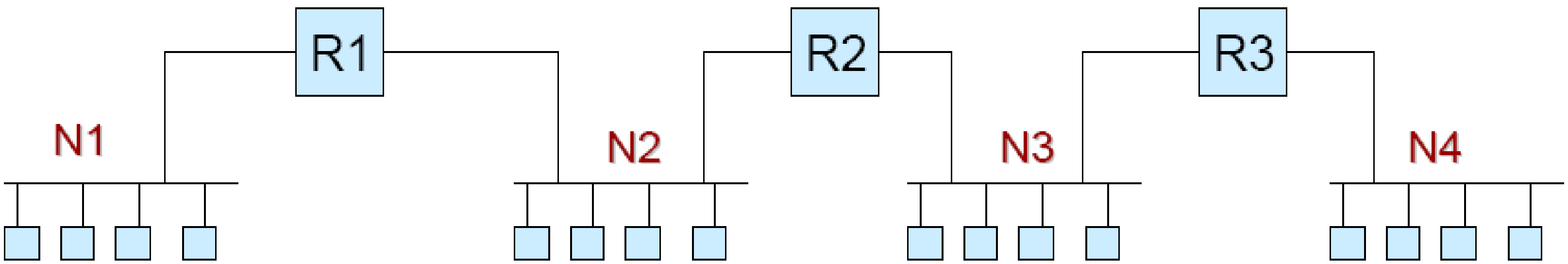
Subnetting

- Subnets can simplify routing.
- IP subnet broadcasts have a hostID of all 1s.



IP Routing

- Q: How do you get a packet from one network to another?



Routing table @ R2

Dest	Next hop
N1	R1
N2	Deliver directly
N3	Deliver directly
N4	R3/default

Routing table

- Destination IP address. Either host address or a network address
- IP address of the next hop Router
- Flags
- Network interface

- `/sbin/route`

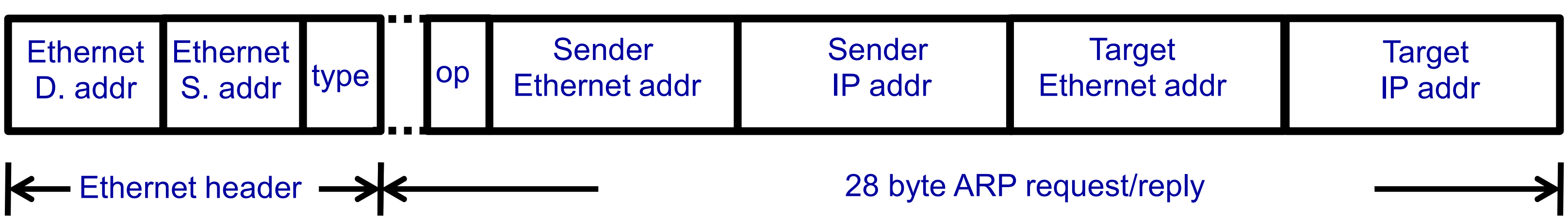
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
129.252.130.0	*	255.255.255.0	U	0	0	0	eth1
loopback	*	255.0.0.0	U	0	0	0	lo
default	SWG130.cse.sc.edu	0.0.0.0	UG	1	0	0	eth1

ARP

- The *Address Resolution Protocol* is used by a sending host when it knows the IP address of the destination but needs the Ethernet (or whatever) address.

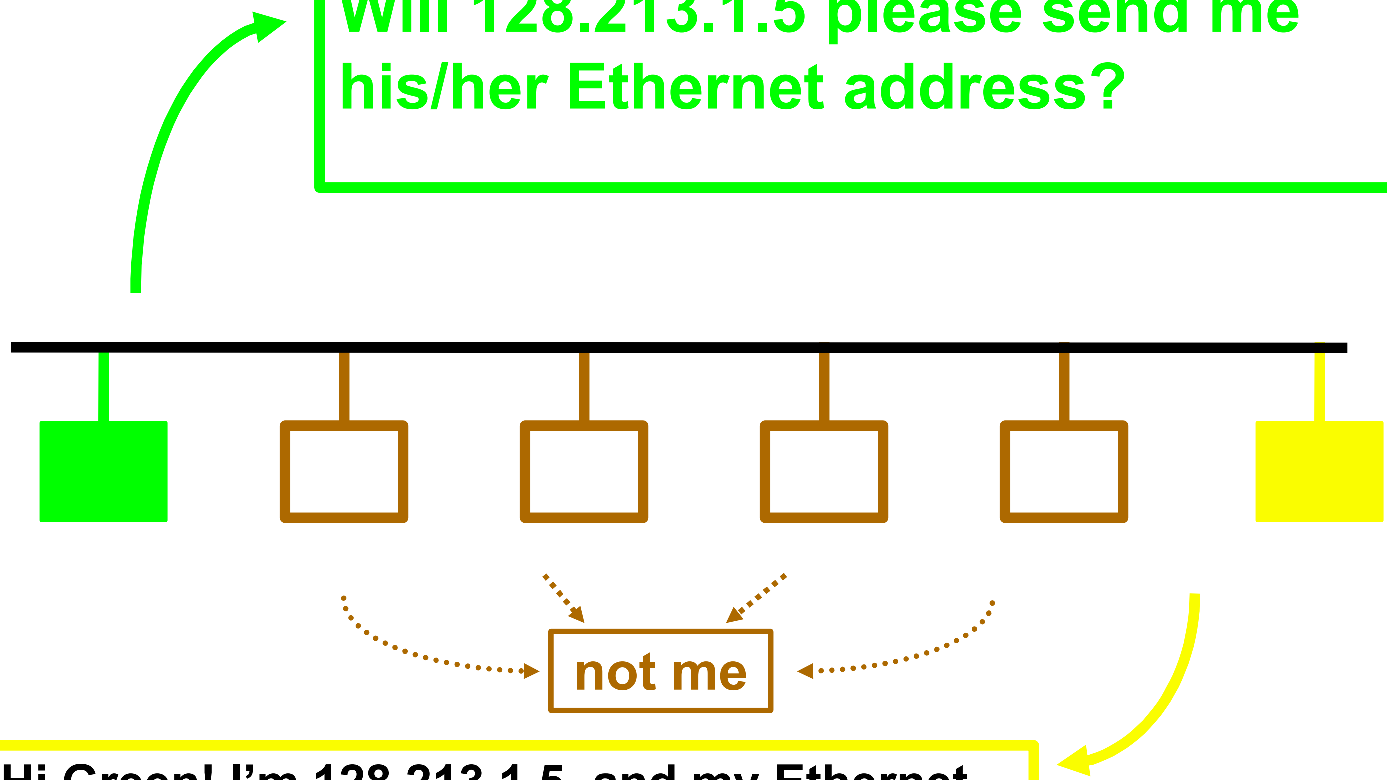
- ARP is a broadcast protocol - every host on the network receives the request.

- Each host checks the request against it's IP address - the right one responds.



ARP conversation

HEY - Everyone please listen!
Will 128.213.1.5 please send me his/her Ethernet address?



Hi Green! I'm 128.213.1.5, and my Ethernet address is 87:A2:15:35:02:C3

Reverse Address Resolution

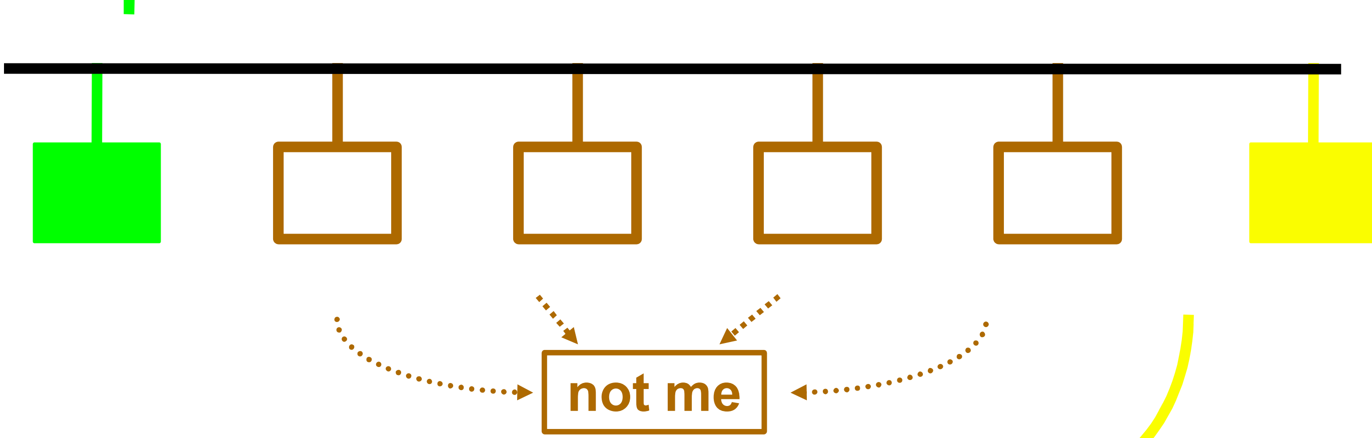
- The process of finding out the IP address of a host given a hardware address is called

Reverse Address Resolution

- Reverse address resolution is needed by diskless workstations when booting (which used to be quite common).

RARP conversation

HEY - Everyone please listen!
My Ethernet address is
`22:BC:66:17:01:75.`
Does anyone know my IP address ?



Hi Green! Your IP address is
`128.213.1.17.`

Services provided by IP

- Connectionless Delivery (each datagram is treated individually).
- Fragmentation / Reassembly (based on hardware).
- Routing.
- Error detection.

IP Datagram Fragmentation

- Each fragment (packet) has the same structure as the IP datagram.
- IP specifies that datagram reassembly is done only at the destination (not on a hop-by-hop basis).
- If any of the fragments are lost - the entire datagram is discarded (and an ICMP message is sent to the sender).

IP Flow Control & Error Detection

- If packets arrive too fast - the receiver discards excessive packets and sends an ICMP message to the sender (SOURCE QUENCH).
- If an error is found (header checksum problem) the packet is discarded and an ICMP message is sent to the sender.

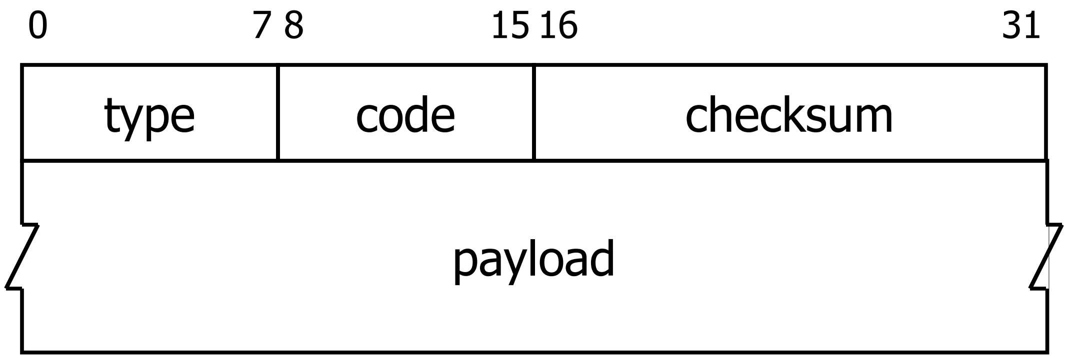
ICMP *Internet Control Message Protocol*

- ICMP is a protocol used for exchanging control messages.
- Two main categories
  - *Query message*
  - *Error message*
- Usage of an ICMP message is determined by *type* and *code* fields
- ICMP uses IP to deliver messages.
- ICMP messages are usually generated and processed by the IP software, not the user process.

IP header	ICMP Message

20 bytes

ICMP Message Format



ICMP Message Types

- Echo Request
- Echo Response
- Destination Unreachable
- Redirect
- Time Exceeded
- there are more …

ICMP Address Mask Request and Reply

- intended for a diskless system to obtain its subnet mask.

- Id and seq can be any values, and these values are returned in the reply.
  - Match replies with request

0 7	8 15	16 31	
type(17 or 18)	code(0)	checksum	
identifier		sequence number	
subnet mask			

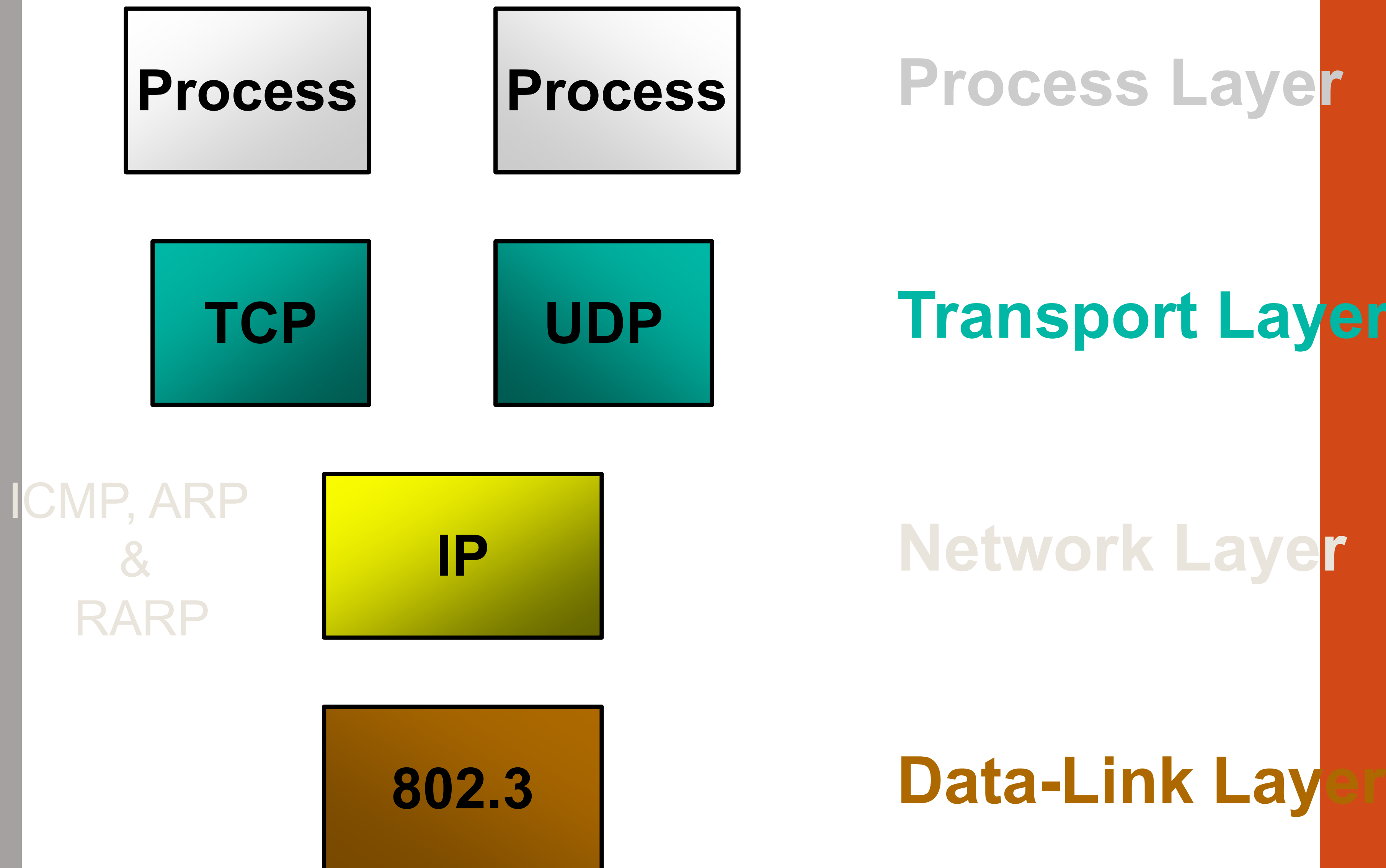
Transport Layer

Transport Layer & TCP/IP

Q: We know that IP is the network layer - so TCP must be the transport layer, right ?

A: No… well, almost.

TCP is only part of the TCP/IP transport layer - the other part is UDP (User Datagram Protocol).


Process
Process
Process Layer
TCP
UDP
Transport Layer
ICMP, ARP
&
RARP
IP
Network Layer
802.3
Data-Link Layer

UDP User Datagram Protocol

- UDP is a transport protocol
  - communication between processes

- UDP uses IP to deliver datagrams to the right host.
- UDP uses *ports* to provide communication services to individual processes.

Ports

- TCP/IP uses an abstract destination point called a protocol port.
- Ports are identified by a positive integer.
- Operating systems provide some mechanism that processes use to specify a port.

UDP

- Datagram Delivery
- Connectionless
- Unreliable
- Minimal

The term *datagram* is also used to describe the unit of transfer of UDP!

UDP Datagram Format

Source Port	Destination Port
Length	Checksum
Data	

TCP
Transmission Control Protocol

- TCP is an alternative transport layer protocol supported by TCP/IP.
- TCP provides:
  - Connection-oriented
  - Reliable
  - Full-duplex
  - Byte-Stream

Wow!

Connection-Oriented

- *Connection oriented* means that a virtual connection is established before any user data is transferred.
- If the connection cannot be established - the user program is notified (finds out).
- If the connection is ever interrupted - the user program(s) is finds out there is a problem.

Reliable

- *Reliable* means that every transmission of data is acknowledged by the receiver.

- If the sender does not receive acknowledgement within a specified amount of time, the sender retransmits the data.

Byte Stream

- *Stream* means that the connection is treated as a stream of bytes.

- The user application does not need to package data in individual datagrams (as with UDP).

Buffering

- TCP is responsible for buffering data and determining when it is time to send a datagram.

- It is possible for an application to tell TCP to send the data it has buffered without waiting for a buffer to fill up.

Full Duplex

- TCP provides transfer in both directions (over a single virtual connection).

- To the application program these appear as 2 unrelated data streams, although TCP can control and data communication by providing control information (such as an ACK) along with user data.

TCP Ports

- Interprocess communication via TCP is achieved with the use of ports (just like UDP).
- UDP ports have no relation to TCP ports (different name spaces).

TCP Segments

- The chunk of data that TCP asks IP to deliver is called a *TCP segment*.

- Each segment contains:
  - data bytes from the byte stream
  - control information that identifies the data bytes

Addressing in TCP/IP

- Each TCP/IP address includes:
  - Internet Address
  - Protocol (UDP or TCP)
  - Port Number

NOTE: TCP/IP is a *protocol suite* that includes IP, TCP and UDP.

TCP vs. UDP

Q: Which protocol is better ?

A: It depends on the application.

TCP provides a connection-oriented, reliable, byte stream service (lots of overhead).

UDP offers minimal datagram delivery service (as little overhead as possible).

TCP/IP Summary

- IP: network layer protocol
  - unreliable datagram delivery between hosts.
- UDP: transport layer protocol
  - unreliable datagram delivery between processes.
- TCP: transport layer protocol
  - reliable, byte-stream delivery between processes.

Hmmmmm. TCP or UDP ?

- Electronic commerce?
- Video server?
- File transfer?
- Email ?
- Chat groups?
- Robotic surgery controlled remotely over a network?