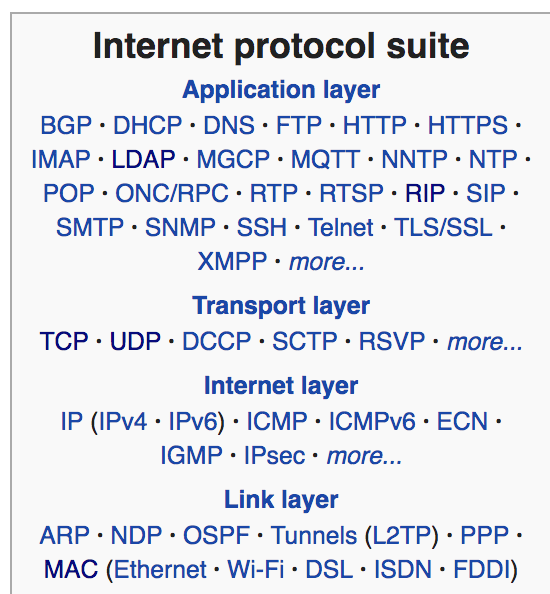
Network Notes

# How does the net work?

* Applications can run over TCP/UDP by sending application data through TCP (stream) / UDP (datagram) sockets
* Applications **can** run directly over IP by sending application data through raw IP sockets
* Client starts communication by requesting connection to server
* Server accepts or declines connection establishment from a client
* After connection has been established, no difference between client and server
* Both client and server can terminate an established connection
* DHCP = Dynamic Host Configuration Protocol, assign dynamic IP addresses to devices
  + Dynamic addressing: device can have different IP address every time it connects to network
* Request and obtain an IP address, from the available pool of IP addresses at the DHCP server
  + requesting device get valid connection to the Internet
* Obtain the address of valid DNS server(s)
  + requesting device can contact these servers to translate names into IP addresses
* Obtain a default gateway
  + Requesting device knows where to send packets when no specific routes match destination of packets
* Cannot request next-hop towards some destination d, so that device can forward a packet to d
* Cannot configure a switch inside a network to forward traffic over all available ports
* Cannot obtain IP address of a given domain name
* Cannot request an Internet route to a destination
* IP addresses = point of attachment in the Internet
  + Logical “position” of a networking interface in the network
  + GPS for internet
* NOT a pair of devices that communicate through same network link
* NOT a physical ID of a piece of hardware connected to a network, like serial number
* NOT a request/connection in Internet like ticket id in serving queue
* Change Wi-Fi you’re connected to -> different IP, different prefix
* Names: easier-to-remember ways to reach networking interfaces
* DNS store mapping of names to addresses, distributed database that can be contacted through DNS servers
  + NOT stored locally by every device connected
* URLs contain **addresses**, not names
* Names and addresses cannot be used indistinctly
* Host with single network interface, IP address 129.104.89.175, prefix 24
* The address 129.104.89.225 is local to this interface, belongs to same network link
* The address 173.104.89.174 is not local, packets addressed to it need at least a router to be delivered, need to be sent to default gateway
* A router that receives an IP packet can drop it or forward it
* Cannot modify source/destination IP addresses in the packet, re-send packet over interface which it was received, reply to the packet
* Socket = pair (IP address, port)
* Endpoint in a network communication
* NOT a network communication or a process running in a device connected to a network
* TCP (Transmission Control Protocol): reliable transport of packets, provides retransmissions if packets are lost, ensures data arrives to destination in same order sent
  + Cost: establish connection, acknowledge receive, retransmissions, high latency
  + Transport layer responsible for maintaining end-to-end communication between two devices
  + Possible to send data packets through TCP/IP network without any transport protocol
* UDP (User Datagram Protocol): doesn’t provide above, packets may be lost or in wrong order
  + Reordering or loss not critical or can be handled by application layer
  + Less cost
* Other transport protocols: SCTP, DCCP…

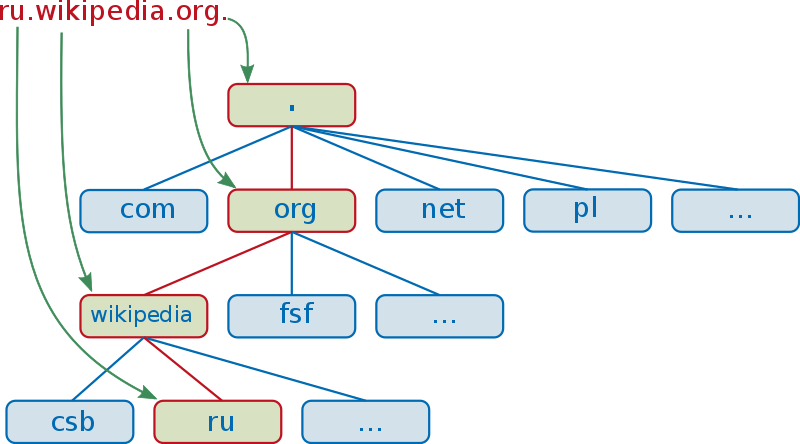
# Components of a computer network

* A layer n (n≥2) = **software**
* **Send** packet: each layer adds corresponding header, send to lower layer
* **Receive** packet: each layer **processes and removes** the corresponding header, sends it to the **upper** layer
* Header of layer n’ > n = **payload**
  + Payload **not** processed
* Asynchronous Transfer Mode (ATM): **datalink** protocol, originally designed to support voice+data traffic
  + Not an obsolete network protocol
  + Not a transport protocol
* Carries frames of **53** bytes (5 header, 48 payload)
  + Doesn’t have a MTU of 1500 bytes
* In Ethernet network (cable), collision on link when 2 stations transmit a packet at the same time
* Listen while transmit: if transmission other than own received, collision
* 2 stations sometimes know, sometimes don’t whether collision occurred
* Minimal duration of transmission < 2\*maximum propagation time: **some** collisions may be unnoticed
* Collision detected by transmitting stations: involved stations retransmit after a **random** time delay (not deterministic)
  + New collision: new transmission scheduled, time delay determined randomly over longer interval
  + Interval for retransmission doubles each time collision detected (reduce proba of new collisions) = *exponential backoff*
  + Third station doesn’t retransmit overheard transmission
* Neither of two stations will detect collision if transmissions too small or stations “too far” (propagation delay too large)
* IPv4: **Address Resolution Protocol** (ARP) finds L2 address of a network interface with given IP address, and vice versa (reverse ARP)
  + Not with Network Address Translation (NAT): assign a public IP address to private IP addresses in local network
* IPv6: exchange of **Neighbor Discovery** messages determines L2 addresses, resolve IPv6 addresses of neighbors in a link
  + Determine relationships between neighboring nodes
  + Replace ARP, ICMP
  + Determine whether L2 address of a neighbor changed, still reachable?
* L2 addresses cannot be used as IPv4/IPv6 or vice versa
* Switches don’t store on switching tables mapping between IP and L2 addresses
  + Given L2 address determine which port to send it
* Routers divide network links (broadcast domains)
* Switches divide network segments (collision domains)
* Same collision domain: cables, repeaters, hubs, bridges
* Collision domain = network segments
* Broadcast domains = minimum number of links (could be more depending on prefix configuration)



* Packet structures (sequence of headers/payload): each layer (in descending order) adds header, (in ascending order) removes header
* [Application (L7) | Transport (L4) | Network (L3) | Data link (L2) | Payload]
  + [IP (L3) | Ethernet (L2) | Payload]
  + [TCP (L4) | Ethernet (L2) | Payload]
  + [Ethernet (L2) | UDP (L4) | Payload] (?)
  + [HTTP (L7) | TCP (L4) | IP (L3) | Ethernet (L2) | Payload]
* Cannot exist:
  + [Ethernet | IP | TCP | HTTP | Payload]
  + [IP | Payload]
* Repeaters amplify signal, regenerates signal, retransmits over one port a signal it receives on the other
  + Overcome physical attenuation
  + Allow network segments to grow longer
* Do not decode and forward signal, inspect datalink header of received packet, do intelligent routing
* Same collision domain
* Hubs: multiport repeater, signal from one port goes out all other ports
* Physical layer (L1)
* Bridge: join two network segments
* Forward traffic between segments
  + If src, dst on same link: don’t forward
* Switch: like bridge with multiple ports, can perform error checking before forwarding
* Switch receive frame for destination not in switching table: forward over all ports **except** the port where frame was received
* Filter, forward packets between links
* Disjoint collision domains
* Link layer (L2)
* Routers: forward data packets along networks
* Usually use IP address
* Network layer (L3)
* TCP/IP **doesn’t** have Presentation layer (6), Session layer (5) unlike ISO/OSI model
* Application layer (7):
* Ensure communication with another application program on network is possible
* Ensure receiving device is identified, can be reached, ready to accept data
* Enables authentication between devices
* Make sure communication interfaces exist (Ethernet/wifi interface in sender’s computer?)
* Present data on receiving end to user application
* Protocol specific to type of application (file transfer, email…)
* Transport layer (4):
* End-to-end semantics (configuration, negotiation between source and destination of transmission)
* Error detection (via checksums), retransmissions in case of packet loss
* Can handle multiple data streams simultaneously
* NO physical transmission over a channel (cable, air etc)
* NO intermediate forwarding towards destination
* Network layer (3):
* Provide addressing and routing, ensure messages delivered to their destination
* Encapsulate messages from higher layers into *datagrams*
* IP: attempts to deliver datagrams (IP packets) on best-effort basis
* Routers forward incoming datagrams according to destination IP address specified in packet
* L3 addresses: logical, virtual addresses
* Ex: IP address
* Belong to single protocol
* A computer can have any number of L3 addresses
* Can be found by DNS
* Datalink (layer 2) technologies/protocols: Avian carrier, Ethernet, Token Ring, ATM, Wifi
* Roles of datalink layer:
  + Communication between 2 interfaces on same physical link
  + Error detection (eg via checksums), correction
  + Doesn’t manage end-to-end semantics (retransmissions, send/receive acknowledgements by destination)
  + Doesn’t do “best-effort” forwarding of packet from one link to another
* L2 addresses: also called hardware, link-layer (LL) addresses
  + NOT MAC address
* Pertains to the actual hardware interface in computer
* Need to be unique on a link (devices on same link = same L2 address)
  + Need not be unique on internet
* Hardcoded (not assigned dynamically) into network interface
  + Identify a networking device with 48 bits (6bytes) and vendor
  + Can be changed by user

# DNS

* Name-to-address translation in ARPANET:
* File */etc/hosts* (like Yellow Pages) manually maintained at Stanford Research institute by researcher
* File contain tuples (name, host) for all hosts connected in ARPANET
* Manual management: didn’t scale when number of hosts on internet increased
* DNS not in use yet
* DNS:
* Application-layer protocol
* Can run over TCP or UDP
  + Use TCP for zone transfers, when packet size too big
* Mostly run over UDP
  + Much faster
  + DNS requests generally very small, fit in UDP segments
  + Add reliability on application layer
* Fully Qualified Domain Names: end with ‘.’, specify exact location in tree of DNS
* [www.polytechnique.edu](http://www.polytechnique.edu).
* mail.polytechnique.edu.
* [www.lix.polytechnique.edu](http://www.lix.polytechnique.edu).
* NOT polytechnique, edu, polytechnique.edu, <http://polytechnique.edu>, <http://www.polytechnique.edu:80>
* DNS can be used for:
* Name to address translation, address to name translation
* Load-balancing
  + DNS distribute list of IP addresses in different order when respond to new client (round-robin)
  + Clients direct request to different servers: reduce traffic, distribute requests to domain across servers
* Alias record: redirection action, multiple names can point to same address
* Resource-sharing: multiple records for same domain
* Anything involving matching names and records
* **Resource record types:**
* **A**: Address record, map domain name to IPv4 address of computer hosting domain
* **AAAA:** like A, for IPv6 address
* **ALIAS:** can’t use CNAME record but can use ALIAS:

# L3 Forwarding, NAT, NAPT, Internet connection sharing

* NAT/NAPT: map one public IP address to port numbers and private IP addresses
* NAT/NAPT divide Internet users into those with “real” Internet connectivity and those restricted to use application that only use **outbound** (from user network to Internet) or **inbound** (from Internet to user network) network connections
* Enable Internet connectivity while hiding internal topology of user’s network
* Don’t enhance security or provide firewall functionalities
* Don’t preserve **end-to-end** principle
  + End-to-end principle: communications protocol operations defined to occur at end-points of communications system, or as close as possible to resource being controlled
  + Net neutrality: ISP refrain from discriminating data on network
* NAT ≠ NAPT
* NAT: **1-to-1** mapping between private and public IP addresses, port kept intact
* NAPT: **1-to-many** mapping, multiple private IP address mapped to single (outside) public IP address by using different *port numbers*
  + Reduce number of IP addresses publicly reachable from Internet
* Packet received at networking layer for given destination:
* Sent through route in routing table with **longest matching prefix** to packet destination
* Sent through **host** route in routing table for which destination = exact packet destination if exists
* NOT sent through route with lowest cost or default route in routing table
* NOT dropped if none of networking interfaces match packet destination
  + Send through default route
* Private IP addresses:
* Not forwarded by routers
* Need to be unique in network (link)
  + Cannot be used by several interfaces in same link
  + Need NOT be unique in internet
* Interface with private address CAN communicate with public IP address by way of NAT or NAPT
  + Cannot be initiated by the public address (private addr cannot be routed unambiguously on Internet)
* Private addressing: deploy local networks and local connectivity, without being assigned any public IP addressing space

Destination 125.105.26.4

prefix cost interface

(1) 125.104.26.0/16   1 eth0  
(2) 125.104.26.0/24  10 eth1  
(3) 0.0.0.0/0   1 eth0

* Forward through eth0, because it’s the interface in default entry (3) providing minimum cost (no prefix match)
* 125.104.26.0/16: 125.104.0.0 - 125.104.255.255
* 125.104.26.0/24: 125.104.26.0 - 125.104.26.255
* Between dots: 8 bits, after slash: prefix length in bits

Destination: 10.1.5.65

Network Interface Next Hop  
default e0 10.1.1.2  
10.1.1.0 /24 e0 DC  
10.1.2.0 /24 e1 DC  
10.1.3.0 /25 s0 DC  
10.1.4.0 /24 s1 DC  
10.1.5.0 /24 e0 10.1.1.2  
10.1.5.64 /28 e1 10.1.2.2  
10.1.5.64 /29 s0 10.1.3.3  
10.1.5.64 /27 s1 10.1.4.4

* Send to s0 (longest prefix): 01000000 vs 01000001
* Laptop connect to IP subnet with prefix 129.104.125.128/25
* Laptop can use: 129.104.125.231, 129.104.125.129, 129.104.125.177
* Cannot use: 129.104.125.255, 129.104.125.34, 129.104.124.129, 192.168.0.1



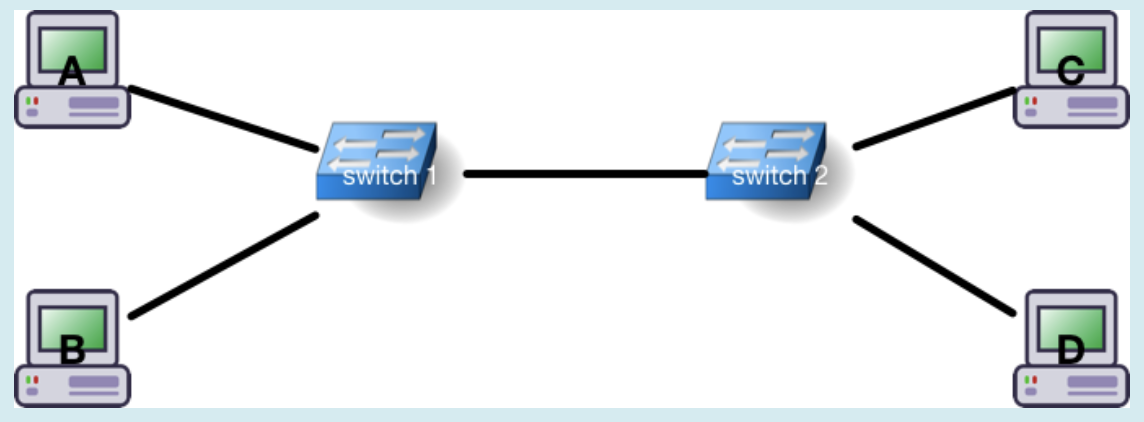
Host A: 129.104.33.25/24

Host B: 129.104.33.27/16

* A and B can communicate to each other: same network
* Doesn’t need routing

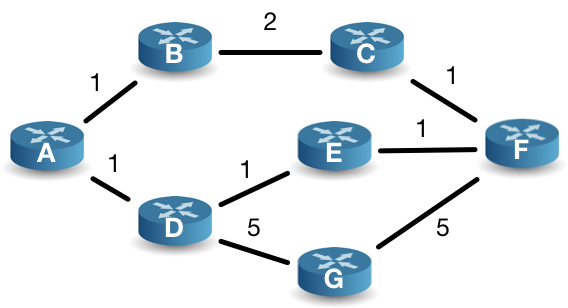
# Spanning tree protocol

* Bridge Protocol Data Unit (BPDU):
* Root ID | Cost | Switch ID | Port ID
* BPDU advertise better path if:
  + Smaller Root ID
  + Tie: smaller cost
  + Tie: smaller switch ID
  + Tie: smaller port ID
* Switch ID: uniquely identify switch on network
* On-link prefix: prefix assigned to link
* Interfaces connected to a link have an on-link prefix
* Hosts use on-link prefix to determine if packet destination on link, or need to be routed beyond corresponding router (gateway)
* Several **on-link** prefixes can be merged in **aggregation/summary** prefix of **shorter** length: efficiently announce prefixes in rest of internet
* Spanning tree protocol: L2 protocol
* Prevent loops on network, find out how switches are organized
* Initialization: each switch assumes themselves is root, creates own BPDU and forward through all ports
  + Switch ID = lowest port ID
  + Root Switch = switch with lowest ID
* Receive better packet: Change root ID to switch ID of better packet, cost +1
* Send new BPDU over port P iff new BPDU better than any received BPDU over P
* **Root port**: port via switch reaches root in spanning tree (non-root switch)
* **Designated port**: Port over which switch sent its own BPDU in last SPT round
  + 1 designated port per segment
  + last port to get root?
* **Blocked port**: doesn’t participate in frame forwarding, discards frames received in corresponding network segment
* Switch receives over port 20 (10,1,50,50), port 21 (30,0,30,32), 22 (30,1,40,41)
* Send over port 21 (10, 1, 20, 21), port 22 (10, 2, 20, 22)

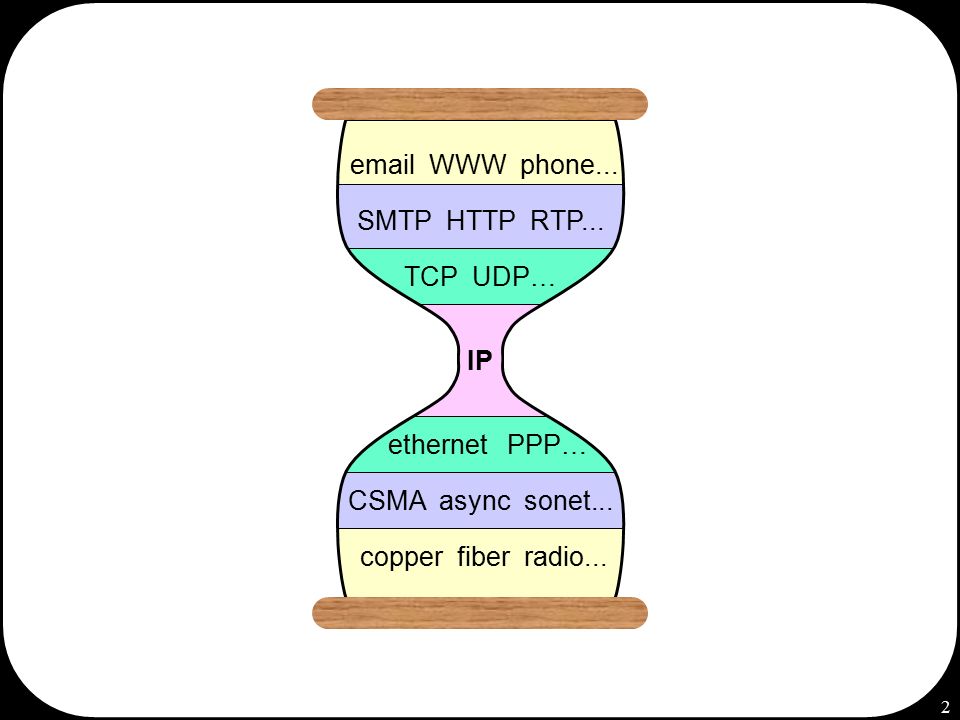


* Switch 1 receives frame from A with destination B
* If switching table of switch 1 has entry with host B, only host B receives packet
* If switching table of switch 1 doesn’t have entry with B, B C D receive packet
* Don’t care about switch 2 table (B not connected to it)
* Destination F: doesn’t exist in network, no entry in switch 1 table, B C D receive packet (**don’t** drop)
* Traffic received at switch at port P is never forwarded if P is **blocked port**
  + P is root port, switch is root, P not designated port, P connected to link whose other endpoints is blocked: irrelevant

# Routing and Routing protocols

* Routers report their list of neighbors to their neighbors (1-hop) via HELLO
* Routers report symmetric links between them and their neighbors to every other router in network through LSA
  + Don’t send their full view of network topology (with neighbors nor other routers)
  + Don’t advertise their costs to reach every destination
* Routers compute shortest paths to every possible destination (Dijkstra) based on topology info received through LSA
* Routing protocol used by **routers**, allow devices running the routing protocol to build routing tables
  + Doesn’t forward packets
* Routing: build paths between sources and destinations
* Forwarding: operation, networking device determines packet received in interface has to be sent through another interface, on which one
* Only routers (not hosts) involved in routing and forwarding
* Not every router has full view of network topology during protocol
* Packets send from same source to same destination **doesn’t always** follow same path
* Hosts receive routing configuration (default route, default gateway) through DHCP or SLAAC
* Hellos: verify **bi-directional connectivity** between neighbors
  + **never** forwarded, send to each neighbor
  + Sent to 1-hop link local multicast destinations
  + include symmetric neighbors and heard neighbors
  + Sent periodically, or triggered by external events
  + Describe info about neighbors of originating router
  + A knows B is bidirectional neighbor if A receives HELLO from B, A listed as “heard” or “symmetric” neighbor of B in HELLO
* LSAs: **flooded** throughout network, information about topology, announce bidirectional links to other routers
  + Discover global topology: learn local topology of flooder
  + Retransmit exact copy of each LSA received exactly once
  + Don’t contain all network links known by originator at time it was generated
  + Describe links from originating router to its neighbors
  + “heard” neighbors not included (only symmetric)
  + Set of all LSA: describe full view of network topology as recorded by routers at given time
  + **Unique** in network, identified by ID of origin router + sequence number
  + Dijkstra run by each router over links described in **all** LSAs received
  + Links may appear in more than one LSA at a time
  + Each router receives a copy of LSA **at least once**
  + Each router retransmits LSA **exactly once** (when they first receive it)
* **Forwarding Information Base** (FIB) derived from Routing Information Base (RIB)
  + Contains the actual rules followed when packet received at incoming interface, select which exiting for packet
* **Routing protocol** allows each participating router to determine its RIB
* **On-link** prefix: interface configured to know, all destinations with this prefix on same link as I
  + Determine if given address reachable directly on same link
* **Summarization** prefix: router representing several destination prefixes through single routing table entry
  + Way for router to combine set of (longer) prefixes received in LSA from other routers, to single routing table entry
  + Reduce size of routing table
* **Aggregation** prefix: router advertise several destination prefixes through as single LSA
  + Way for router to combine set of (longer) prefixes to advertise as single entry in LSA
  + Reduce number and/or size of advertisements router generates
* ****Link-stating routing protocol converged, link (D,E) breaks:
* D and E detect link broke, no longer receive HELLO messages from each other
* D and E update HELLOs they generate (remove D,E as symmetric neighbor)
* D,E issue new LSA describing their new local topology (D: {A,G}, E: {F})
* A receive LSA: A recompute shortest path to F, install new routing table to F (via B), send to F through {A, B, C, F}

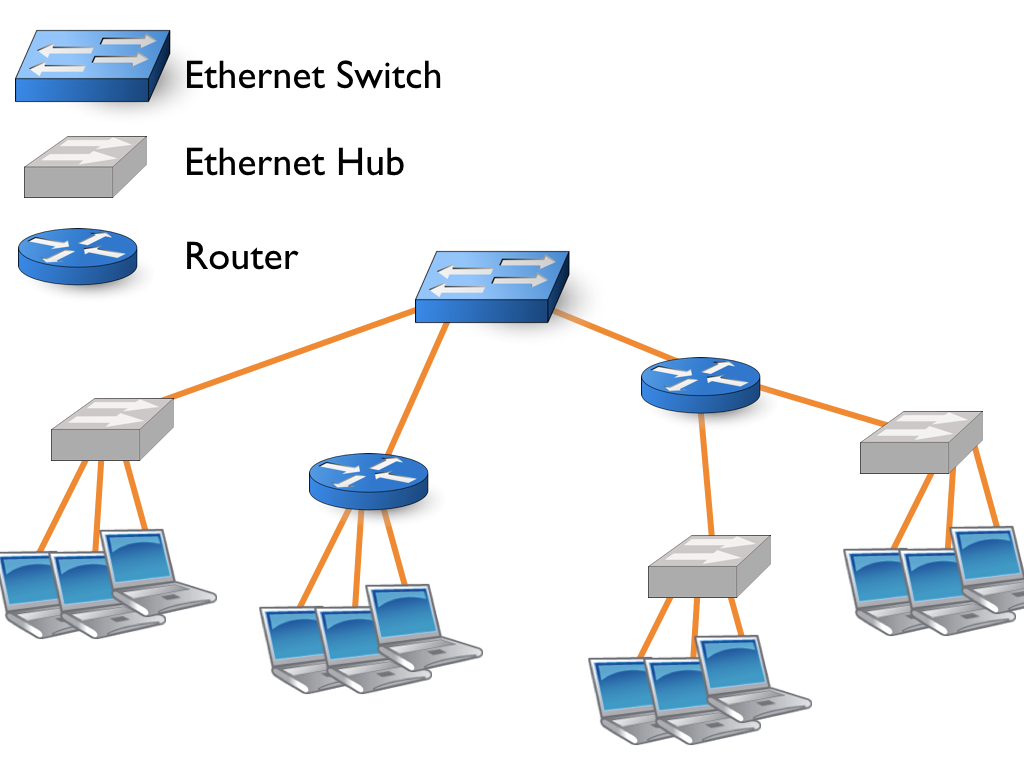
# BGP, DNS and IP

* **Unicast, anycast**: communication between exactly 1 sender, 1 recipient
* Unicast:sender, recipient uniquely identified
* Anycast: sender uniquely identified, recipient chosen from unknown set of candidates
* **Multicast**: 1 sender, set of recipients
  + Recipients are named but unknown to sender
* Multicast addresses different from unicast addresses
  + Operations by router when processing multicast packet (packet replication, transmission over multiple interface) different than unicast (identification of egress interface, forwarding)
  + Routing tables, protocols for multicast different than unicast
* BGP: routing protocol
* Exchange data about internet paths to IP prefixes between groups of interconnected networks (Autonomous System)
* Runs on border routers in each AS
* **Classful** addressing: original IP addressing scheme
  + IP divided into class A-E
  + Class A: prefix of 8 bits, 2^(32-8) addresses
  + Class B: 16 bits prefix, 2^16 different addresses
* **Classless** Interdomain Routing (CiDR): allow to use network prefixes of any size
* Use variable number for network and host portions of address
* More flexible allocation of IP addresses
* Internet Assigned Numbers Authority (IANA): main non-profit responsible for allocation of IP addresses
* Delegates for each region a Regional Internet Registry (RIR) allocation of IP address blocks
* 2011: IANA run out of IPv4 unallocated addresses
* RIR level: slightly later
* Dual-stack, transition technologies: accelerate IPv4 to IPv6 transition
* Dual-stack: allow devices to configured in both IPv4, 6
  + Happy eyeballs: speed up connections, try simultaneously IPv6 and IPv4, stay with fastest
* 6to4 and derivation 6rd: transition mechanisms
  + use IPv6 in endpoints, transfer IPv6 over IPv4 infrastructures by tunneling
  + 6rd: algorithmic method to map IPv4 to 6 addresses
* IPv6 traffic higher during weekends: better support at residential environments
* IPv4 -> IPv6:
* Addresses grow from **4 bytes** to **16 bytes**
* Basic troubleshooting tools (ping, traceroute) roughly same
* IPv6 version of DHCP: DHCPv6
* DNS basically preserved
* IPv6 Neighborhood Discovery performs tasks handled by IPv4 DHCP and ARP

## Practice Exam

* Protocol stack: abstract mechanism to represent complex communicating system
  + Allows heterogenous networks: between sender/receiver, complex multi-hop network across several different link types (WiFi, Ethernet, Optical Fiber…) may exist
  + Lower layers don’t know, don’t care about higher layers, two endpoints may define and use their own protocols without requiring an update to all intermediaries (routers)
  + Provide abstraction: higher layer needs not be aware of what lower layers are used
  + DOESN’T make networks faster because packets are smaller
  + DOESN’T mean only single transport protocol is ever going to be needed
* Protocol: horizontal interaction between 2 instances of **same** layer
* Layer: well-defined role in communication framework, specific role (task) with specific well-defined responsibilities
  + Each layer interacts with immediately adjacent layers in same stack through send()/receive() primitives
  + Payload at layer (n-1) = (header, payload) from layer n
  + DOES NOT specify finite set of protocols, protocol operations permitted
* Protocol number: allows network layer to identify to which transport layer instance a given data packet should be delivered
* Port number: allows transport layer to uniquely identify an application on computer, determine which application given packet should deliver
  + TCP: 0 reserved, can’t be used, 1 process per (IP, port)
  + UDP: source port is optional, 0 = no port
  + Bind socket (file descriptor) with a transport protocol, IP address and port number
* Application on internet is uniquely identified by *(IP address, protocol number, port number)*
* Maximum transmission unit (**MTU**): largest size of packet or frame that can be sent
  + Too large: retransmissions if packet encounters router that can’t handle its size
  + Too small: more header overhead, acknowledgements that have to be sent/handled
  + Layer 2: larger than MTU, silently discarded
  + Layer 3: larger than MTU, fragment (in IPv4)
  + Ethernet: 1518 bytes (1500-bytes payload)
* Minimum transmission unit (**mtu**):
  + Synonym with *minimum frame size*
  + Important to detect collisions: enough time for transmitter to get collision signal during its transmission, know it has to retransmit
  + Ethernet: 64 bytes (18-byte header, 46-bytes payload)
* **Transport** **layer**: ensure end-to-end properties of communication
* Uniquely ID process by **port number**
* Offers transport service used by application
* Some protocols provide reliable ordered data delivery, others not same end-to-end semantics
* TCP, UDP transport layer protocols
  + NOT SNMP, SMTP, HTTP
* Different applications on same computer may use different transport layers
* Expects that (inter)networking layer offers (nothing more/less than) end-to-end forwarding of data packets through a multi-hop network
* Demultiplexing construct used to determine to which application a packet is to be delivered, *commonly* called port number (TCP UDP is port number)
* Can handle flow control, congestion control (adapt transmission rate of sender to capacity of receiver and of intermediate routers on path)
  + Some transport protocols do proactive flow/congestion control, others don’t
* DOES NOT present application data to receiving application
* **Network layer** provide connectivity across multi-hop network
* Identify unique process by **IP addresses + protocol number**
* Aka convergence layer
* Network layer protocols distinguished between each other (on same machine) by way of Ethertype
  + Ethertype: field in Ethernet frame (layer 2), indicate which protocol to use in layer 3
  + Almost like port number for layer 2
* Receive unicast L3 datagram: network layer inspects destination address
  + Address identify *itself*: hand datagram off to appropriate transport layer (determined by protocol number)
  + Other than itself: forward datagram, look up in the FIB interface over which next hop (along shortest path to destination) is reachable, transmit over it
* **Forwarding Information** **Base** contain pairs *(destination, interface)*: “to get packet closer to *destination*, send it over this *interface*”
  + destination itself or next hop towards destination
  + **At most** a single entry per destination in FIB
  + Match destination address from incoming diagram and FIB: use *longest prefix matching*
  + Contain at least destination, cost, next hop
  + May also contain quality of service, interface (i.e eth0)
* Unicast routing table of a router contain:
  + DOESN’T contain path packet will take between router and ultimate destination
  + DOESN’T contain tuple (destination, next-hop, distance, next-hop-queue-length)
  + DOESN’T contain copy of all received control packets (all BGP paths received, or LSAs for OSPF…)
* **Routing Information Base** (RIB) contain *(destination, next-hop, distance)*: “to get packet closer to *destination*, if you send it to this *next-hop* neighbor, then cost of whole path to destination is *distance*”
  + Zero, one or more entries in RIB
* **Best Effort**: for incoming diagram, decision can be immediately made to
  + Deliver packet to local process (via transport layer)
  + Forward packet over well-defined and identified interface
  + Silently drop packet
  + DON’T buffer packet or start discovery process to determine path towards destination
  + Makes internet more scalable
  + DOESN’T provide quality-of-service guarantees or reliability
* IPv6 example of network layer protocol
* NOT responsible for providing exclusive access to a communications channel (network access control)
* NOT responsible for providing determined end-to-end semantics to applications
* Routing Protocols (like RIP, OSPF) are NOT network layer protocols
  + RIP runs over UDP, application layer
  + OSPF encapsulated in IP, datalink layer
* ICMP NOT network layer protocol (transport layer)
* In the internet, a router does not:
* Increment TTL field of an IP packet
  + De-incremented by router
* Run protocol such as Spanning Tree Protocol to setup IP routing tables
  + Ran by switches
* Encapsulate incoming IP packet with another
  + IP in IP: add header, done in transport layer
* Fragment IPv6 packet before forwarding it
  + Never in IPv6: packets exceeding MTU are dropped
  + IPv4: Router can break packets into smaller pieces so that fragments can pass through link with smaller MTU (or drop and send ICMP)
* **What does a router do?**
* **Data-link layer**: provide transmission of data across a single link
  + NOT against a physical link
* Responsible for Medium Access Control: responsible for moving packets from a Network Interface Card to another across a shared channel
  + Also (framing, error correction ?)
* Identify unique process by **HW Address + Ethertype**
* DOESN’T need to provide guarantee that data packet handed to Network Layer will be successfully/correctly delivered to intended destination



* 5 collision domains
* 5 broadcast domains
* Minimum number of IP links such that all interfaces are part of a link = 5
* Need minimum of 5 on-link prefixes
* Minimum number of on-link prefixes required = 4