Network Layer Protocols:
IPv4 (Internet Protocol Version 4)

Internet Protocol Version 4 (IPv4)

- Network layer protocol in TCP/IP suite
- Defined originally in RFC 791 (*what is a RFC?*)
- Connectionless (i.e., no explicit connection setup/termination phase before/after data transfer), datagram-oriented
- Message broken up into packets, packets switched between routers. Header attached to each packet (IP header).
- Main issues handled:
  - Routing
  - Fragmentation and reassembly
- Unreliable, best-effort service. Packets can be lost, duplicated, received out-of-sequence.
- Encapsulated in data link layer frame (for ex. Ethernet)

IP Address

- Required to specify source and destination of packet
- Each host connected to the Internet is identified by a unique IP address (actually, each network card on a host has an IP address)
- An IP address is a 32-bit quantity.
  - Expressed as W.X.Y.Z, where W, X, Y, Z are the four octets
  - Consists of two logical parts:
    - A prefix for network id
    - A suffix for host id within that network
- Important to remember that an IP address is actually assigned to an interface (NIC), not a machine
  - A machine may have more than one IP addresses if it has more than one NIC
    - Ex. – your laptop (the machine) can have one IP address assigned to the LAN interface and another to the wireless interface
  - Though the term host, machine, device, …used interchangeably and informally very commonly when talking about IP address assignment

Classful Addressing

- All addresses are to fall in five well-defined IP address classes

Class A	UNICAST
Class B	UNICAST
Class C	UNICAST
Class D	MULTICAST
Class E	RESERVED

- Each class has a pre-defined number of bits for the network part

Class	Address Range	High-order bits	Network bits	Host bits
A	0.0.0.0 – 127.255.255.255	0	7	**24**
B	128.0.0.0 – 191.255.255.255	10	14	**16**
C	192.0.0.0 – 223.255.255.255	110	21	**8**
D	224.0.0.0 – 239.255.255.255	1110		
E	240.0.0.0 – 255.255.255.255	1111		

CLASS A

0 || Network	Host	Host	Host

No. of networks: $2^7 - 1 = 127$

No. of hosts/network: $2^{24} - 2 = 16,777,214$

CLASS B

10 || Network	Network	Host	Host

No. of networks: $2^{14} - 1 = 16,383$

No. of hosts/network: $2^{16} - 2 = 65,534$

CLASS C

110 || Network	Network	Network	Host

No. of networks: $2^{21} - 1 = 2,097,151$

No. of hosts/network: $2^8 - 2 = 254$

Network Specification

- Specified as W.X.Y.Z/p where
  - W.X.Y.Z has the same form as an IP address
  - p is an integer saying first p bits of the address identify the network, the remaining (32 – p) bits can be used for assigning host addresses in that network
- For class A, B, C, p is 8, 16, 24 respectively
  - 64.0.0.0/8 is a Class A network
  - 129.100.0.0/16 is a Class B network
  - 223.200.100.0/24 is a Class C network

Some Special IP Addresses

- Loopback address
  - Loopback address - allows applications on same host to communicate using TCP/IP
  - Anything starting with 127., usually 127.0.0.1
- Net-directed Broadcast (to *netid*)
  - NETID = *netid*, HostID = all 1's
  - Not assigned to any host
- Network address
  - NETID = *netid*, HostID = all 0's
  - Not assigned to any host
- Private IP addresses
  - 10.0.0.0/8, 172.16.0.0-172.31.255.255, 192.168.0.0/16
  - Should be used only internally in an organization, should never go on the internet
  - No central allocation, anyone can use it

Problems with Classful Addressing

- IP block allocation was done by class
  - An organization gets only multiples of Class A, B, and C
  - Wastage of IP addresses
    - Organizations may not need a full class
      - Especially for Class A and B
    - Big problem, as IP address space is bounded
      - May run out of IP addresses for new machines eventually

Classless Addressing

- No well-defined class
  - No fixed size prefix for network part
  - Choose the prefix size as per need
- Allocate only what is needed
  - Value of p (number of network bits) can be arbitrary
  - Ex: An organization needs only 56 IP addresses
    - Classful addressing option: Allocate a whole Class C, say, 220.100.200.0/24 (24 network bits, 8 host bits), wastes around 200 addresses
    - With classless addressing, you can allocate 220.100.200.0/26 (256 network bits, 6 host bits)
    - Allows the allocation of 220.100.200.64/26, 220.100.200.128/26, and 220.100.200.192/28 to 3 other organizations with similar needs (needs to connect < 64 machines)

IP Address Allocation

- Done centrally by IANA (www.iana.org)
- IANA assigns unallocated IP blocks to RIRS (Regional Internet Registries)
  - AFRINIC, APNIC, LACNIC, ARIN, RIPE NCC
- RIRs allocates to ISPs and other users in their region
- Defined policies exist for allocation
- Region/ISP based allocations keep IP address blocks in a region mostly contiguous
  - Important for keeping routing tables small (we will see)

IP Subnetting

- Subnet (or subnetwork)
  - A subdivision of a network
  - Breaks host part further into subnet part and host part
- Allows better network administration and management
- Reduces route table size in non-local routers (will see later how)

- Uses network masks (subnet mask)
  - In binary, the mask is a series of contiguous 1's followed by a series of contiguous 0's.
  - The 1's portion identifies the network portion of the address (see example)
  - The 0's portion identifies the host portion of the address in the original
  - To check if two IP addresses belong to the same subnet or not, check if the bit-wise AND of the two addresses with the netmask is the same or not.

An Example

- Network mask 255.255.255.240 is applied to a class C network 195.16.100.0
- Mask = 11111111 11111111 11111111 11110000
- Address of 1st host on this subnet = 195.16.100.1 (0 = 0000 is special)
- Address of last host on this network = 195.16.100.14 (15 = 1111 is special)
- Next subnet will start from 195.16.100.16
- Addresses 195.16.100.3 and 195.16.100.12 are in the same subnet by the previous rule
- Addresses 195.16.100.3 and 195.16.100.19 are in different subnets by the previous rule

Note: original subnet specs (RFC 950) do not allow subnets with all 0's and all 1's (so 195.16.100.0 and 195.16.100.240 subnets cannot be used). Current systems allow with suitable configuration.

Subnetting Example

- We want to break 195.6.100.0/24 into 2 subnets with 64 addresses each, 4 subnets with 32 addresses each
- We first get 4 no. 64 address subnets
  - 64 addresses needs 6-bit host address
  - So network part is 32 – 6 = 26
    - Top 2-bits of the last 8 bits (the host part in the original network) are to be included in network part of the subnets
  - Get the 4 subnets by varying the 7th and 8th bit (from right) from 00 to 11
    - 195.6.100 part of the address remains the same
  - The four subnets are
    - 195.6.100.0/26 (Subnet 1, bit value = 00)
      - Addresses 195.6.100.0 to 195.6.100.63
    - 195.6.100.64/26 (Subnet 2. bit value = 01)
      - Addresses 195.6.100.64 to 195.6.100.127

        - 195.6.100.128/26 (Subnet 3, bit value = 10)
            - Addresses 195.6.100.128 to 195.6.100.191
        - 195.6.100.192/26 (Subnet 4, bit value = 11)
            - Addresses 195.6.100.192 to 195.6.100.255
- Now we break Subnet 3 into 2 no. 32 address subnets
    - 32 addresses need 5-bit host part
    - So network part is 32 – 5 = 27 bits
        - Top 1-bit of the last 6 bits (the host part in Subnet 3) is to be included in network part of the new subnets
    - Get the 2 subnets by varying the 6th bit (from right) from 0 to 1
        - Remaining bits of Subnet 3 remains the same
    - The two subnets are
        - 195.6.100.128.0/27 (Subnet 3(a), 6th bit value = 0)
            - Addresses 195.6.100.128 to 195.6.100.159
        - 195.6.100.160/27 (Subnet 3(b), 6th bit value = 1)
            - Addresses 195.6.100.160 to 195.6.100.191
- Break Subnet 4 similarly (work it out writing out the bit patterns)

Natural Masks

- Class A, B and C addresses each have natural masks, which gets defined from the definition of the classes themselves.
  - Class A :: natural mask is 255.0.0.0
  - Class B :: natural mask is 255.255.0.0
  - Class C :: natural mask is 255.255.255.0
- Network address can be specified by either specifying the netmask explicitly, or by the / notation

Packet Forwarding

- Each machine has a routing table
  - We have already seen typical fields in a routing table entry
  - Routing table entries populated statically or by a routing protocol running in the background
- Any IP packet received (from the Ethernet layer) is checked against the routing table
- Forwarded to higher layer (TCP/UDP) if this machine is the destination
- Forwarded to next hop using the interface specified in the entry
- But first lets see what is in the IP header

IP Header

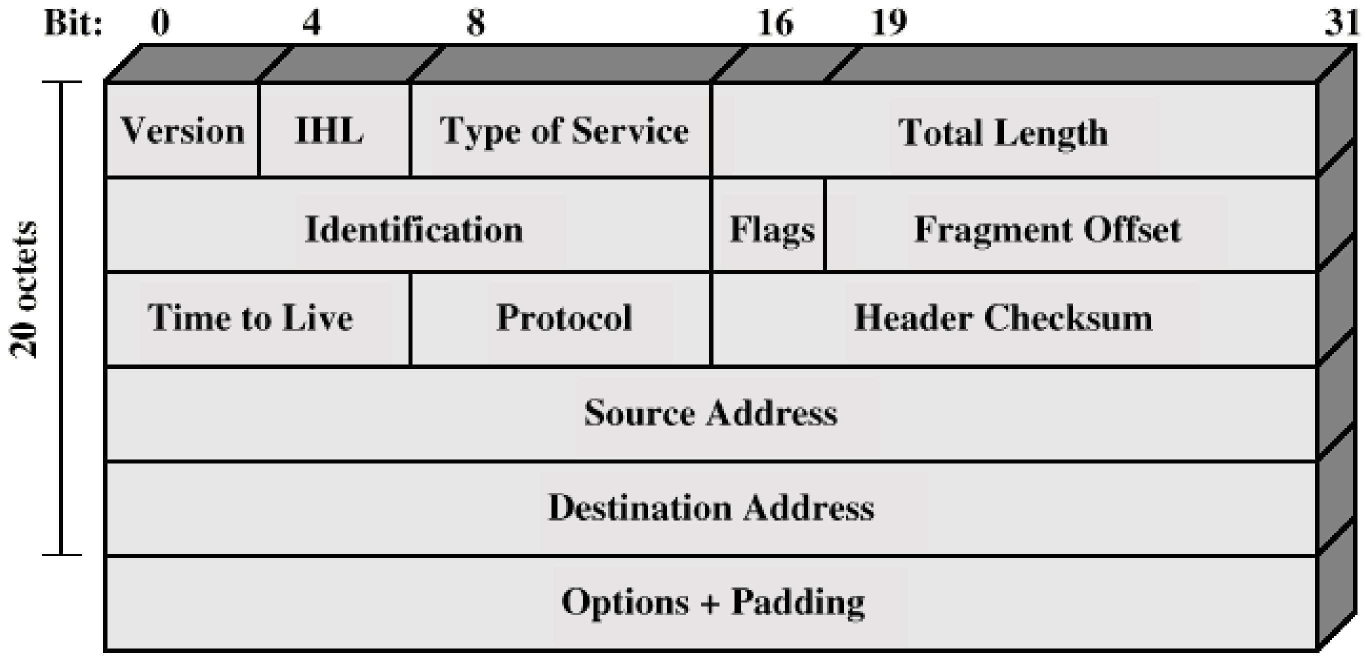


Figure 15.6 IPv4 Header

What’s Stored in an IPv4 Header?

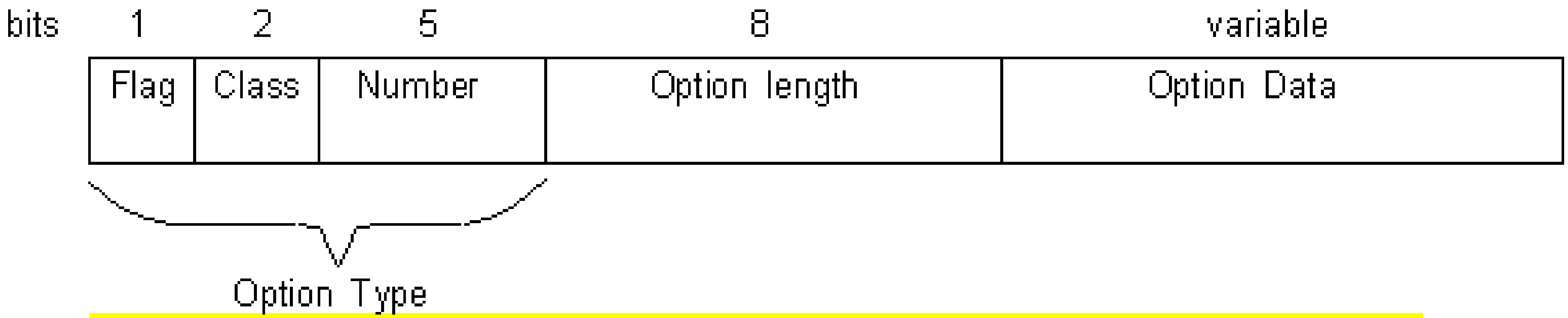
- Version: 4 bit field specifying the IP version (4)
- Header length: specified in 32 bit words. Range is 5..15 words, or 20..60 bytes
- Type of Service (8 bits)
  - 3 bit precedence field, one “must be zero” field
  - 4-bit field specifying desired service qualities.
    - Minimize Delay
    - Maximum Throughput
    - Maximize Reliability
    - Minimize Monetary Cost
  - Only one bit can be set. None set is “normal service”
  - Largely ignored by routers & IP implementations

IPv4 Header (contd.)

- Datagram length: header + data, in bytes
- Identification
  - Unique value, used with flags & fragment offset if a message must be fragmented
- Flags, Fragment offset – discussed later
- Time to live field - upper limit on the number of "hops" a message can go before being dropped
- Protocol: identifies higher layer protocol like TCP, UDP *etc*.
- Header checksum: checksum of just the header
- Source address
- Destination address
- Options

Options Field

- Can specify a variable number of options, each of the form



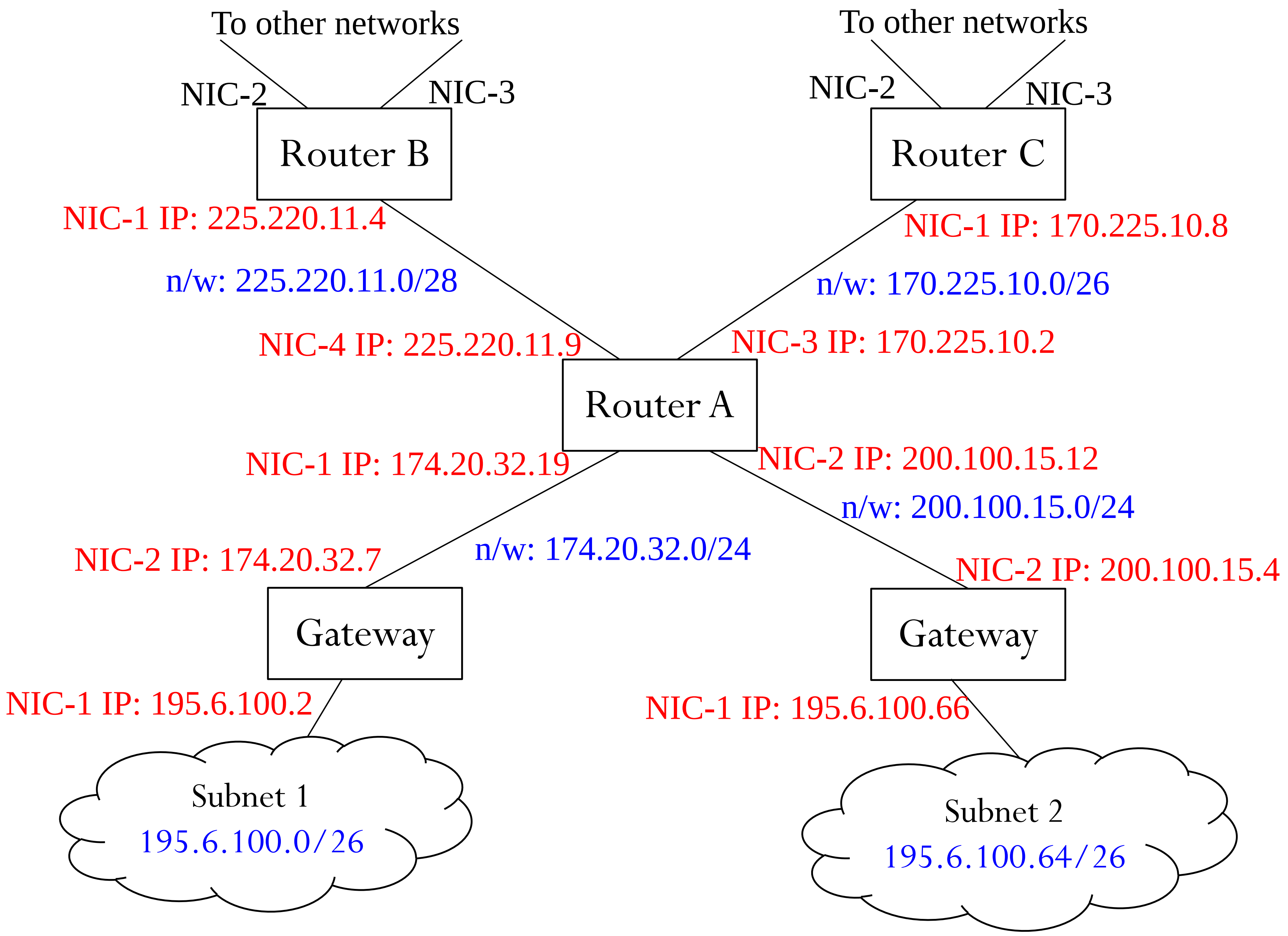
- Flag – 0 means the option is NOT to be copied to each fragment if the datagram is fragmented, 1 means to be copied
- Class – 0 (Normal), 2 (debugging)

- . Option Number
    - 0 - the end of the option list,
    - 1 - No Operation
    - 2 - Security
    - 3 - Loose Source Routing
    - 4 - Internet Timestamp
    - 7 - Record Route
    - 8 - Stream ID
    - 9 - Strict Source Routing
- Option-Length - variable and not present for the NOP and the end of Option List
- Option-Data - variable and not present for the NOP and the end of Option List. See RFC 791 for the detail on the data content for each of the options if you want
- Overall, options are not much used for internet communication

Back to Packet forwarding

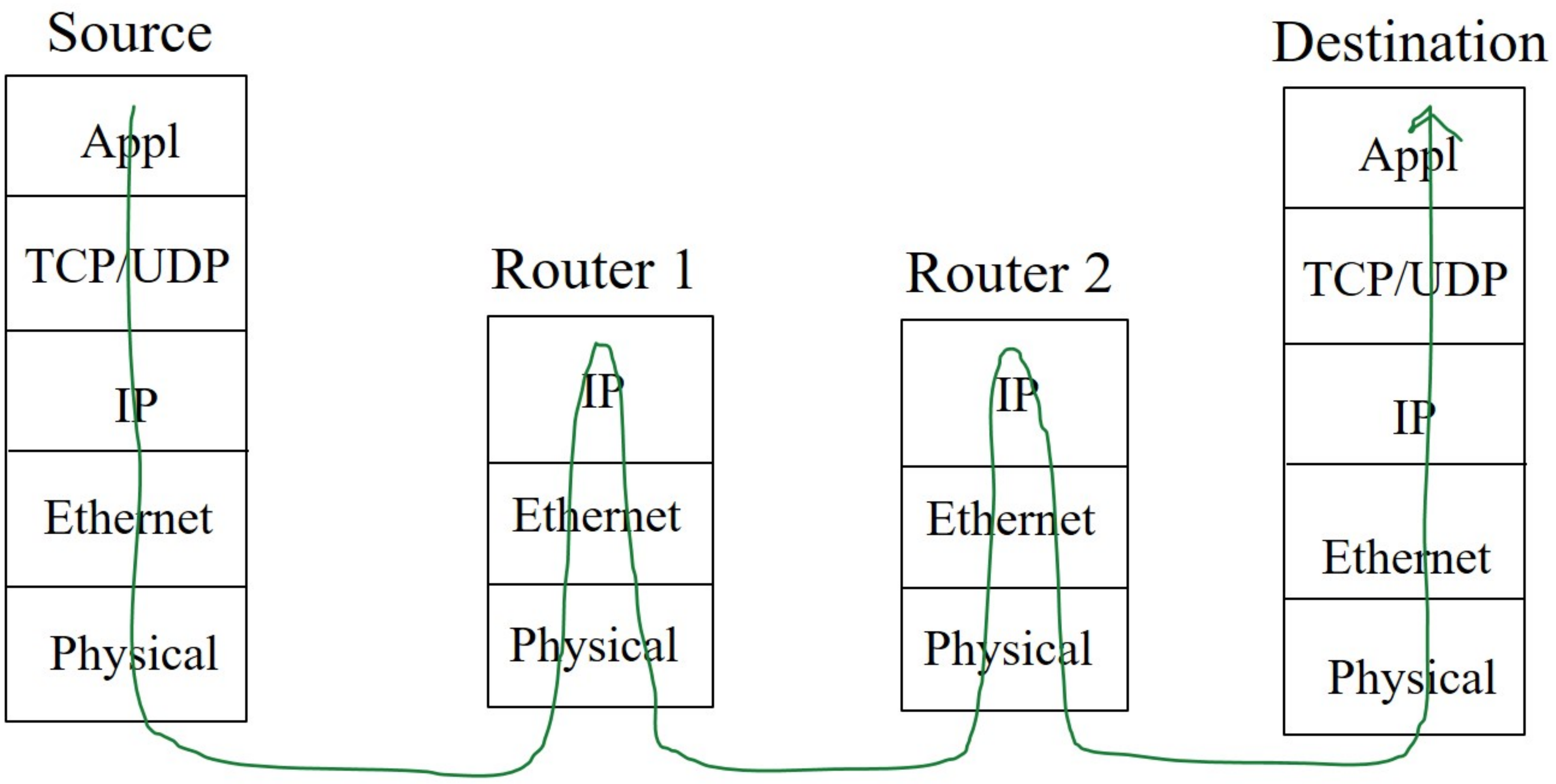
- As mentioned, forwarding will check destination IP (from header) against routing table entries
- Two types of machines
  - End devices that are sources and destinations of data (ex. Your PCs/Laptops etc.)
    - Belongs to some subnet
    - Can have only one NIC with an IP address assigned from that subnet
      - May have more, ex. a wired and a wireless interface
    - Routing tables are usually static
      - Built by system during network configuration
      - Routes can be added manually also
      - No routing protocol runs to dynamically update routes
      - Small and simple tables
    - Each subnet will have a gateway (router) that connects it to the rest of the larger network

- Routers
  - Routes packets, not source or destination
  - Connects more than one network
    - Connected to the gateway/router of each network
    - So must have more than one NIC, one for connecting to each router
  - Each NIC has an IP address assigned from the network it is connected to
  - Runs some routing protocols to build the routing table dynamically
  - Static routes can also be added in addition
  - Larger size, more complex



To other networks
NIC-2
NIC-3
Router B
To other networks
NIC-2
NIC-3
Router C
NIC-1 IP: 225.220.11.4
n/w: 225.220.11.0/28
NIC-1 IP: 170.225.10.8
n/w: 170.225.10.0/26
NIC-4 IP: 225.220.11.9
NIC-3 IP: 170.225.10.2
Router A
NIC-1 IP: 174.20.32.19
NIC-2 IP: 200.100.15.12
n/w: 200.100.15.0/24
NIC-2 IP: 174.20.32.7
n/w: 174.20.32.0/24
NIC-2 IP: 200.100.15.4
Gateway
Gateway
NIC-1 IP: 195.6.100.2
NIC-1 IP: 195.6.100.66
Subnet 1
195.6.100.0/26
Subnet 2
195.6.100.64/26

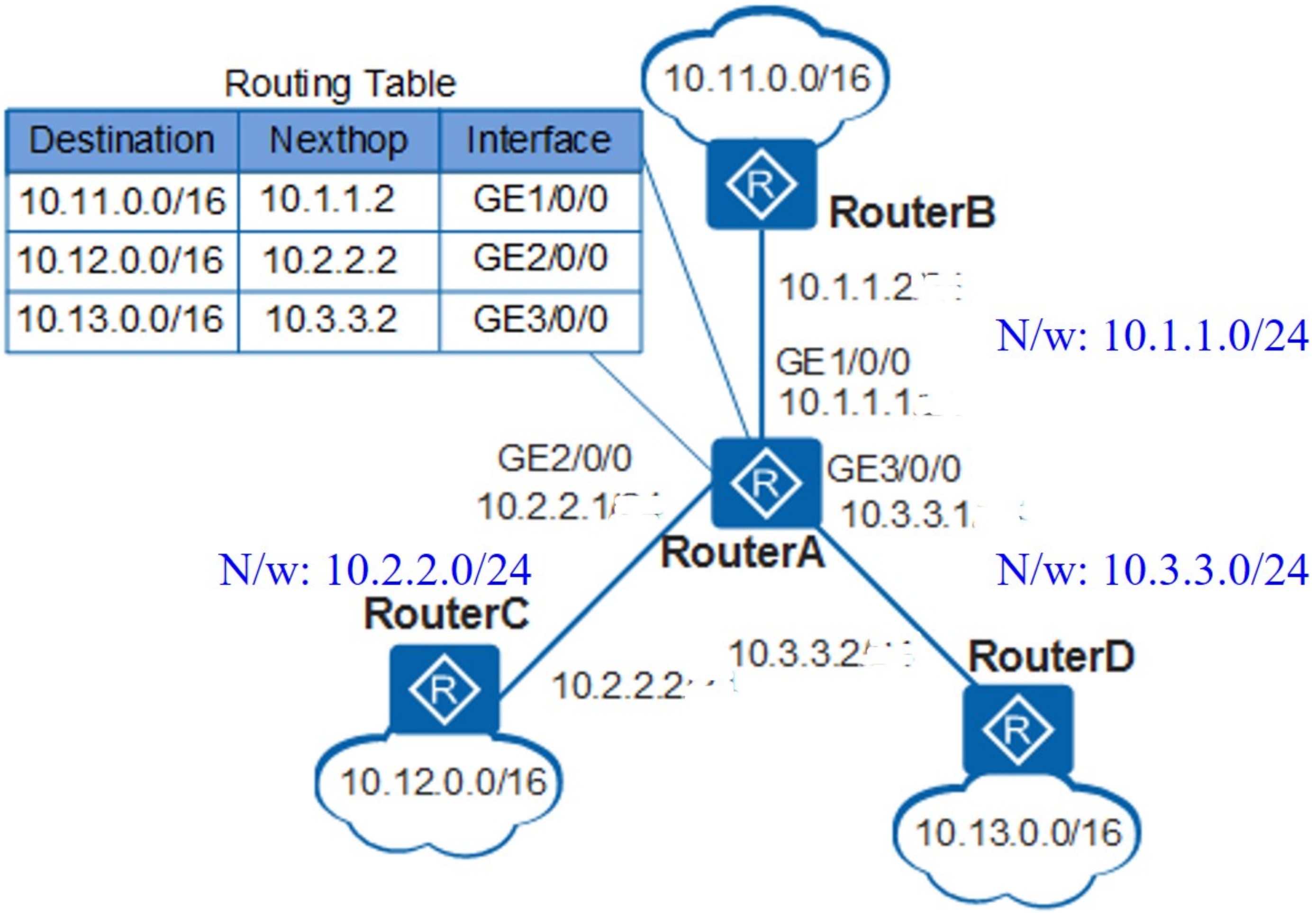
Layers that an IP packet flows through from Source to Destination



A router has only till IP layer. No need for higher layers.

Routing by the Network

- When IP packet comes, destination address checked with routing table to find next hop's IP address
- So should each router keep one entry for each host address existing in the network?
  - Will explode the routing table at higher level routers
    - Think of the no. of hosts they will have to keep track of
    - Increases both storage and search time
- Solution: Route by Network
    - Routing entries specify networks, not hosts
    - In IP routing, the "destination" field in each routing entry is a network address, not a host address
      - Look at the IP address in it in conjunction with the subnet mask or the / part


Routing Table

Destination	Nexthop	Interface
10.11.0.0/16	10.1.1.2	GE1/0/0
10.12.0.0/16	10.2.2.2	GE2/0/0
10.13.0.0/16	10.3.3.2	GE3/0/0

- Routing by network will still need one entry per subnet
  - Still very large
- CIDR (Classless Inter Domain Routing)
  - Allocate IPs in contiguous blocks using classless addressing
  - Combine contiguous networks into a larger network for the purpose
  - Saves IP space and reduces routing table size
  - Also called Supernetting or Route Aggregation

Example

- Suppose company X needs 1000 IP addresses from its ISP Y
- ISP Y allocates the network 192.60.128.0/22
  - A contiguous block of 1024 addresses
- Company X now subnets these into 8 subnets with 128 addresses each
  - 192.60.128.0/25 (Subnet 1)
  - 192.60.128.128/25 (Subnet 2)
  - 192.60.129.0/25 (Subnet 3)
  - 192.60.129.128/25 (Subnet 3)

  .

  .

  .
  - 192.60.131.128/25 (Subnet 8)

- Router of X, R_X, will have 8 entries in its routing table, one for each subnet
  - Each subnet will have its own gateway
  - A subnet's entry in R_X will have the subnet's gateway as its next hop
- Router of Y, R_Y, will have only one entry, for the "network" 192.60.128.0/22 in its routing table
  - The next hop for that entry will be R_X
- Any higher level router that R_Y connects to will also have just one entry for the 8 subnets, with R_Y as the next hop

- Possible only due to contiguous allocation, so that higher level routers can just send it to lower level routers (in this case company A's router) using one entry only for all the subnets. Lower level router will distinguish.
  - If the 8 subnets assigned were not contiguous, R_Y would need 8 entries also
- Possible because of classless addressing
  - Note that the 1024 address block is nothing but 4 contiguous Class C blocks
  - But looked at as classless, using subnet mask to distinguish

- Routing table at all higher level routers:
  - 192.60.128.0/22 - send to R_Y (next hop on way to Company X's router R_X)
- Routing table at R_X :
  - 192.60.128.0/25 – send to router of first subnet
  - 192.60.128.128/25 – send to router of second subnet
  - 192.60.129.0/25 – send to router of third subnet
  - 192.60.129.1/25 – send to router of fourth subnet
  - ….

- So routing table will contain networks (in prefix form or explicit net/mask form) and maybe some hosts (32 bit prefix), and a next hop address for each of them, plus some other information
- Routers always do longest prefix match. If two entries match, longest match is taken.
  - Example: two entries in table: one for 192.65.0.0/16 and one for 192.65.128.0/24. If address is 192.65.128.4, second entry will be used even though it matches both.
- Usually a routing cache is also there. Cache contains recent routing decisions. Destination address first looked up in cache. If not found, longest-prefix match done in routing table.

Routing Table Example (PC that is source and destination of data)

```
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      10.124.52.2   10.124.52.149     35
      10.124.52.0    255.255.255.0         On-link    10.124.52.149    291
    10.124.52.149  255.255.255.255         On-link    10.124.52.149    291
    10.124.52.255  255.255.255.255         On-link    10.124.52.149    291
        127.0.0.0        255.0.0.0         On-link        127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link        127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link        127.0.0.1    331

===========================================================================
```

Basic Packet Forwarding Rule

- Match the destination IP for network part against routing table entries
  - For each routing entry, bitwise AND the subnet mask to the destination IP, and see if the result is equal to the network part in the routing entry. If yes, match found
  - If multiple matches found, choose the longest prefix match
- Check the gateway (next hop) part of the matched entry
  - If local ("On-Link" in our earlier example, can be specified in other ways like IP of this machine etc.), the destination IP is in the same network, no need for any further routing.
    - Send to destination IP (how?)
  - If some other IP, send to it for further routing (how?)

- Sending to destination m/c in local network
  - If this m/c is the destination, send up to TCP/UDP layer
  - Otherwise,
    - Can send using broadcast MAC address in Ethernet frame
      - All nodes on network receive it, pass it up till IP layer, only destination m/c accepts (based on destination IP), others drop it
    - Can send with destination m/c MAC address in Ethernet frame if known
      - Only the destination passes it up to the IP layer, rest all drops it at Ethernet layer itself
      - MAC can be known by ARP (see later)
- Sending to next hop through interface specified
  - Same as above
  - Note that the next hop must have at least 2 network interfaces in this case, one in this network, one to some external network (its next hop towards the final destination)
  - One of them must be in the same network as this machine

ARP (Address Resolution Protocol)

- Provides IP to Hardware address mapping
- ARP packet encapsulated in Ethernet
- Type field in frame specifies it is an ARP packet
- Broadcast IP for which hardware address is needed to all nodes, destination picks it up and replies
- ARP cache maintained for faster mapping

ARP Packet Format

0	8	16	31
Hardware type = 1		ProtocolType = 0x0800	
HLen = 48	PLen = 32	Operation	
SourceHardwareAddr (bytes 0 – 3)			
SourceHardwareAddr (bytes 4 – 5)		SourceProtocolAddr (bytes 0 – 1)	
SourceProtocolAddr (bytes 2 – 3)		TargetHardwareAddr (bytes 0 – 1)	
TargetHardwareAddr (bytes 2 – 5)			
TargetProtocolAddr (bytes 0 – 3)			

RARP

- Opposite of ARP
- Gets IP address given Hardware Address
- Used during bootup by diskless workstations etc. to initialize IP
- No role in packet forwarding

Example

- IP packet sent from A=144.16.192.55 to B=192.15.32.3 through gateway C=144.16.192.2
- Since A sends to C first, should A change the destination address in the packet to 144.16.192.2 (address of C)?

NO!!!

- IP address is for end-to-end communication, C will need it to know that the packet is for B
  - Replacing the destination IP field from B to anything else loses the final destination permanently
- A will look in ARP cache to see if C's MAC address is already there
- If found,
  - Decrement TTL field in IP header by 1 if not the source
  - Recompute checksum in header
  - Send the IP packet to Ethernet along with MAC address of C
  - Ethernet layer will put its own header with destination MAC = C's MAC address and send

- If not found,
  - Create an ARP packet with target protocol address = C's IP address (144.16.192.2)
  - Send to Ethernet with broadcast MAC address
  - Ethernet puts its own header with destination MAC = all 1's and sends
  - C receives the ARP and sends an ARP response with its MAC address
  - ARP response received by Ethernet passed back to ARP processing software
  - ARP protocol adds this to the ARP cache, and also gives to the waiting packet
  - Rest of the steps is same as the earlier case (entry found in ARP cache) for sending the packet
- C will then follow the same steps to send towards B
  - Look up routing table, find next hop etc.

- Questions
  - What happens if the user gives a wrong destination IP address?
  - What happens when the TTL becomes 0?
  - What happens if the gateway/next hop is dead?
- We have so far looked at forwarding using the routing table
- Next lets see how the routing tables are built
  - Most of the concepts covered already
  - Will look at some IP network specific things briefly

Internet Routing Architecture

- Internet divided into multiple Autonomous Systems (AS)
  - Each AS is under a single administrative domain
  - Each AS has a AS number administered by IANA
- Routing broken up into two levels
  - Routing among AS's
    - Done by Exterior Gateway Protocols like Border gateway Protocol (BGP)
  - Routing within an AS
    - Done by Interior Gateway Protocols like RIP (Distance vector protocol), OSPF (Link state protocol) etc.
- For now, we will introduce only Interior Routing Protocols
  - Will do some BGP at end if time permits

IP Routing Protocols: RIP

- Routing Information Protocol
- Simple implementation of distance vector routing
- Routing updates sent to neighbors every 30 seconds or so
  - An update is a list of <destination network, metric> pairs advertised
- Metric used – hop count (no. of hops to destination)
- Does not replace equal cost routes to prevent oscillations

- Route invalidation – if the route is not advertised for some time (usually around 180 seconds)
- $\infty = 16$ to prevent long convergences for counting to infinity problem
  - Limits it to small networks where legitimate hop counts has to be maximum 15
- RIP packets are carried over UDP port 520
- Message formats etc. not important for us now

IP Routing Protocols: OSPF

- Open Shortest Path First
- An example of link state routing protocol
- Much more powerful than RIP
- Features
  - Load balancing
    - Can specify multiple routes with same cost, traffic distributed equally over all routes
  - Handle type of service
    - Multiple routes can be installed, one for each type of service
    - Forwarding uses both destination address and type of service

- Supports Authentication
- Supports hierarchical routing
  - Allows partition of network into smaller self-contained Areas
- Supports the use of different cost metrics
- Works directly on IP, so less overhead

OSPF Basics

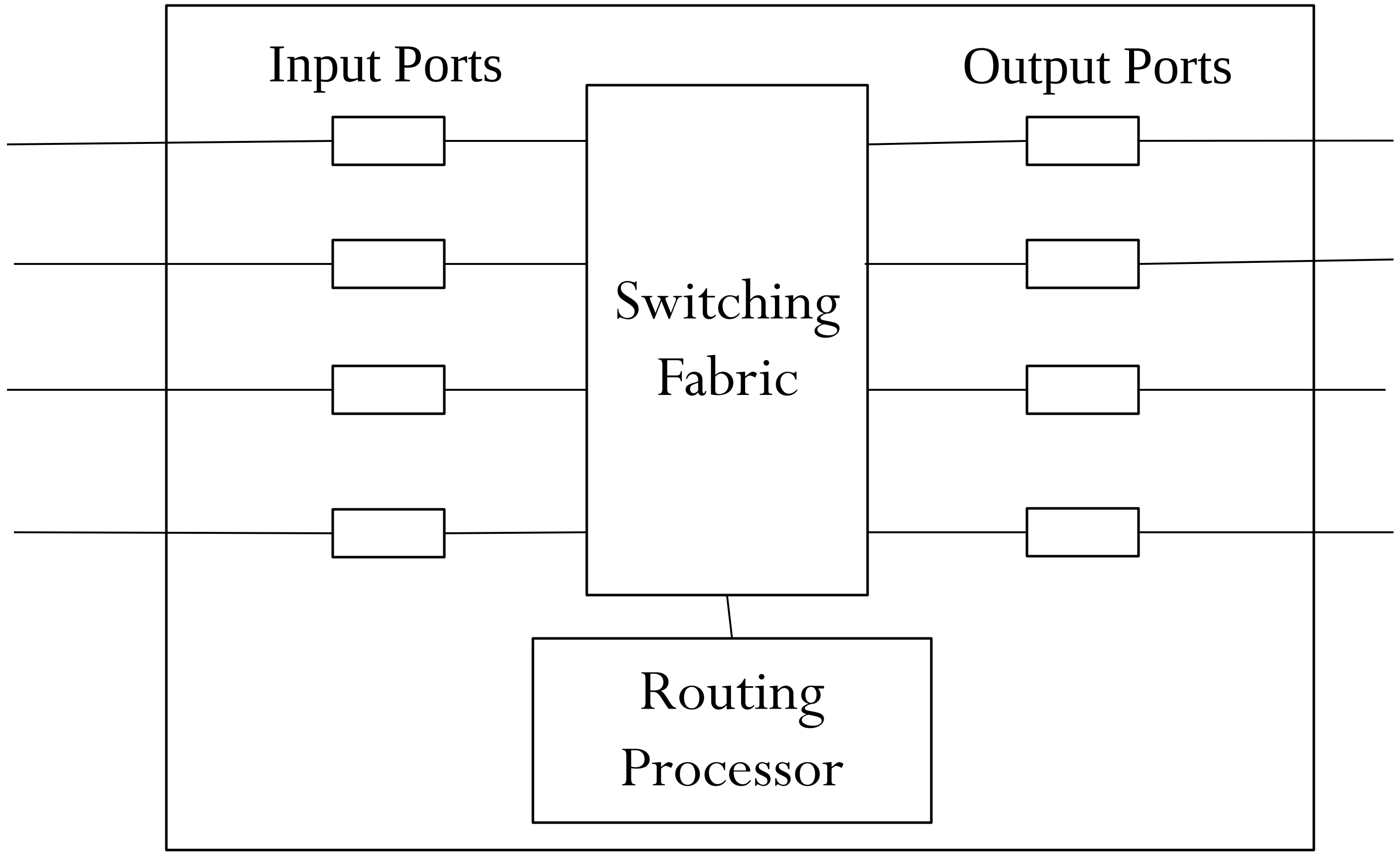
- Each node maintains a database of Link State Advertisements (LSA)
  - Each LSA contains information about a link
  - Age field in each entry to keep track of how long that LSA is not updated
    - LSA removed if not updated within some time (typically 1 hour)
  - Sequence numbers with each LSA to identify stale LSAs
- Databases synchronized when two neighboring routers first discover each other
  - All LSA's sent (in multiple messages if needed)
- LSA's flooded also periodically
  - But period is large, around 30 minutes
- LSA's are checksum'ed for error detection, acknolwedged for reliability

OSPF: Basic Messages

- Hello : sent on each link periodically to discover neighbors and test status (sent to predefined multicast address)
  - Typically every 10 second, link assumed dead if no hello received for 4 intervals (typical value)
- Link State Request (LSR): sent to a neighbor to ask for status of some specific links
- Link State Update (LSU): sent on detecting a link change and also periodically by a node
  - Each message contains a number of LSA entries
- Link State Acknowledgement (LSA): Ack for the Link Status Update message
- Database Description (DBD): used to synchronize the link state databases of adjacent routers

- Works directly on IP, with protocol value = 89
- All OSPF message sent with TTL=1 in IP header
- Powerful protocol widely used for interior routing
- More details beyond the scope of this class given the time

Basic Structure of a Router



Router and L3 Switch

- L3 Switch – another name for the device doing the basic routing and forwarding that we have seen
- The difference between a router and a good L3 switch is confusing
    - Most functionalities are the same
- However, generally, routers
    - Have more features and configuration settings
    - Can connect to a wider variety of WAN technologies
    - Have lower number of ports than L3 switches
    - Costs more
- Nevertheless, the difference is blurred nowadays

A 48-port switch

A chassis-based higher capacity switch

IP Fragmentation & Reassembly

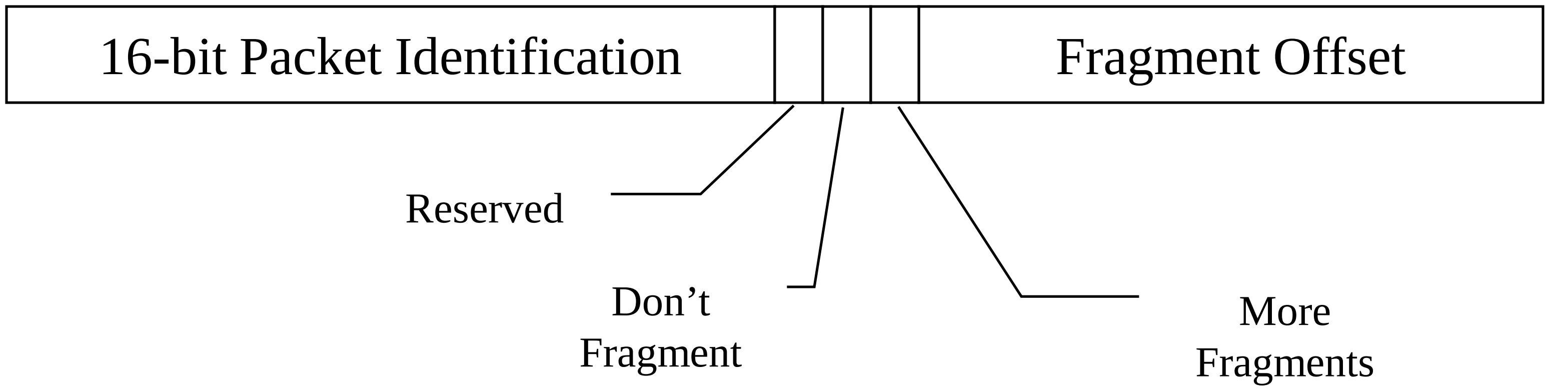
IP Fragmentation

- Maximum Transfer Unit (MTU) – maximum size of a physical frame
  - 1518 for Ethernet (including header), so 1500 bytes for the data
- When a router transmits a packet that is too large for the MTU of the outgoing link, the packet is fragmented
  - Otherwise the link layer will not be able to carry it
- Fragments may also be fragmented
- Fragmented packets are not reassembled until they reach their final destination
- Typically, if any fragment is lost, a router will discard all fragments. Routers usually only discover fragment loss if they drop the fragment themselves.
- The endpoint assumes fragments are lost after 30-60 seconds

Packets *vs.* Datagrams

- An IP datagram is the unit of end-to-end transmission at the IP layer (before fragmentation & after reassembly)
- A packet is the unit of data passed between the IP layer and the data link layer.
- A packet can be a complete IP datagram or a fragment

IP Header Fields Used for Fragmentation



Identifying and Ordering Fragments

- Fragments are identified using the datagram identification field in the header
  - All fragments of a datagram has the same identification field value
- Sequencing of fragments done using the Fragment Offset field
  - Contains offset from beginning of data carried by this fragment (in units of 8 bytes)
- Last fragment detected by “More Fragment” field
  - 1 bit of the Flags field in the header
  - Set to 1 on all but the last fragment of a datagram; set to 0 for the last fragment

Example

- IP datagram of size 2000 bytes (without header) with identification field = 12345 (say)
- MTU = 1500 bytes (without header)
  - So IP header + data in each packet must be ≤ 1500 bytes
  - Assume IP header size = 20 bytes
  - So maximum data that can be packed into one fragment = 1480
- Will break into 2 fragments of size 1480 and 520 bytes
- Put an IP header on each fragment (20 bytes header)
- Header fields
  - Fragment 1
    - Identification = 12345, Offset = 0, MF = 1, Total length = 1500
  - Fragment 2
    - Identification = 12345, Offset = 185, MF = 0, Total length = 540
  - Rest of the fields same (except checksum).
- Send the two fragments as independent IP packets

- Reassembly at receiver
  - Can happen only after all fragments are received
  - If any one fragment is lost/corrupted, the datagram cannot be reassembled
    - All fragments received of that datagram dropped after a timeout
  - How to know all fragments received?
    - Offset at each fragment says how many bytes present in other datagrams before this.
    - Gaps can be easily detected

- If a fragment is fragmented again at an intermediate router
  - Say at router A, the next link MTU is 900
  - Fragment 2 will pass fine
  - Fragment 1 will be fragmented again into 2 fragments
    - Fragment 1a and 1b with size 880 bytes (= 900 – size of IP header) and 600 (1480 – 880) bytes
    - Fragment 1a
      - Identification = 12345, Offset = 0, MF = 1, Total length = 900
    - Fragment 1b
      - Identification = 12345, Offset = 110, MF = 1, Total length = 620
  - What if the MTU was 400, so both fragments will get fragmented again?
    - Work out yourself. Think how the reassembly will happen

- What if we do not want a packet to be further fragmented in the middle by any router?
  - Set the DF (Don't Fragment) flag
  - If it cannot be sent by a router because MTU is smaller, it will be dropped
  - So why will we ever want to use it? Will see a scenario.

Internet Control Message Protocol (ICMP)

ICMP

- Required protocol with IP
- Used for reporting errors back to the source of an IP packet or for monitoring/measurement/feedback
- When a node drops an IP packet, an ICMP packet is sent back to the source
- Sent in some other cases without any error also
- Only error reporting, no error correction. Correction is left to the source node.
- RFC 792 and RFC 1122 (lists updated types)

ICMP Transmission

- ICMP packet contains ICMP header and may contain other information depending on type of message
- ICMP packet carried in data portion of an IP packet
- IP packet is routed normally back to the source
- Only difference: No ICMP packet is generated on error in IP packet carrying ICMP
- ICMP is not a transport layer protocol, it is a required support protocol at IP layer

ICMP Header Format

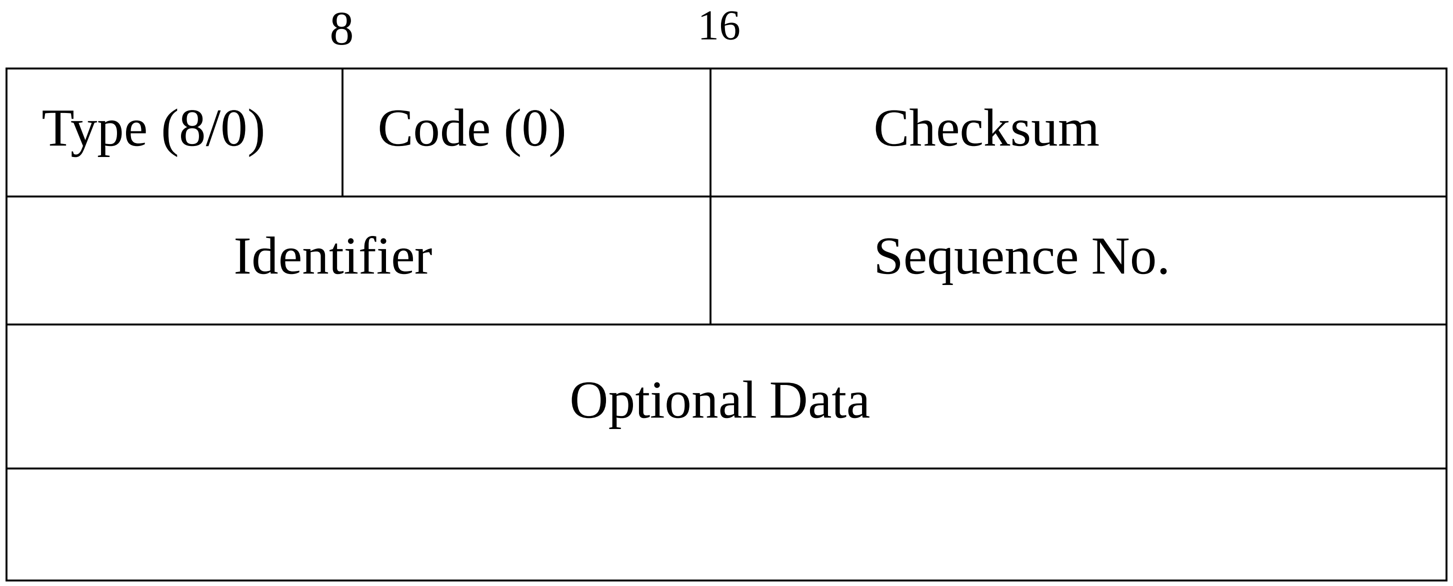
- 8-bit Type field
- 8-bit Code field
- 16-bit Checksum

ICMP Message Types

ICMP Message	**Type**
Echo reply	0
Destination unreachable	3
Source quench	4
Route redirect	5
Echo request	8
Time exceeded	11
Parameter problem	12
Timestamp request	13
Timestamp reply	14
Address mask request	17
Address mask reply	18

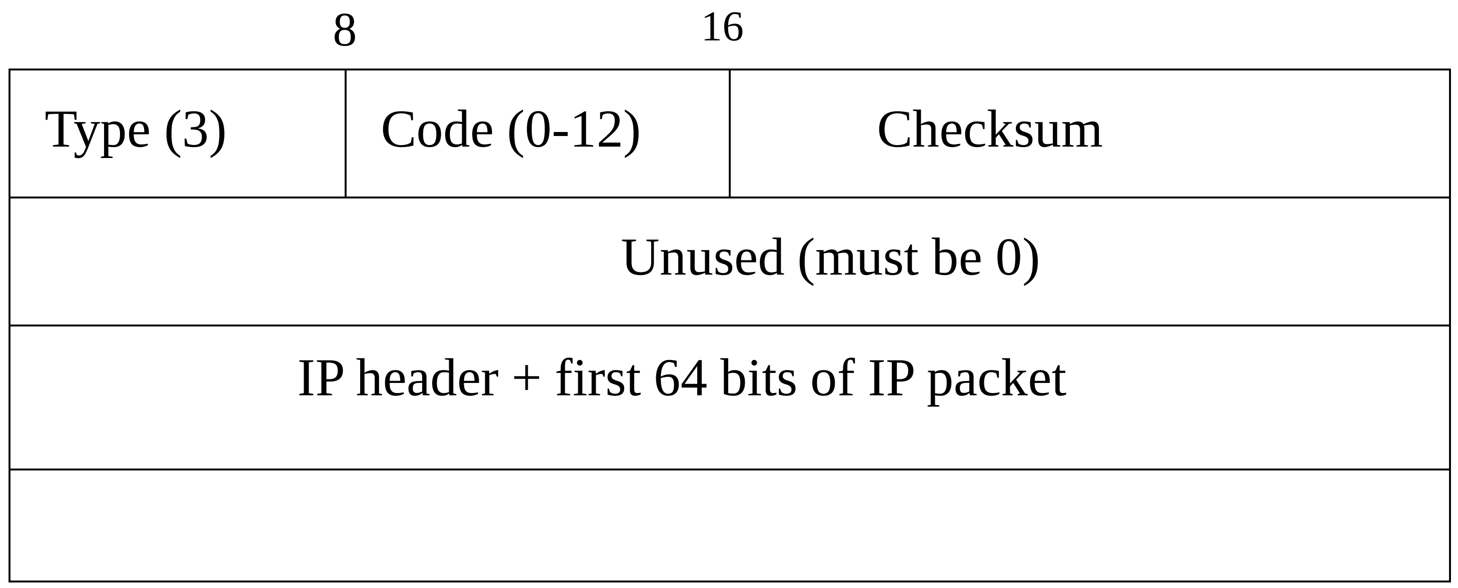
Echo Request/Reply

- To see if a destination is up and reachable
- Identifier/Sequence no. used to match request with reply
- Data may be sent in request, same data returned in reply, matched to see if destination is up
- Basis of *ping* tool



Destination Unreachable

- Sent by router when unable to forward or deliver a packet

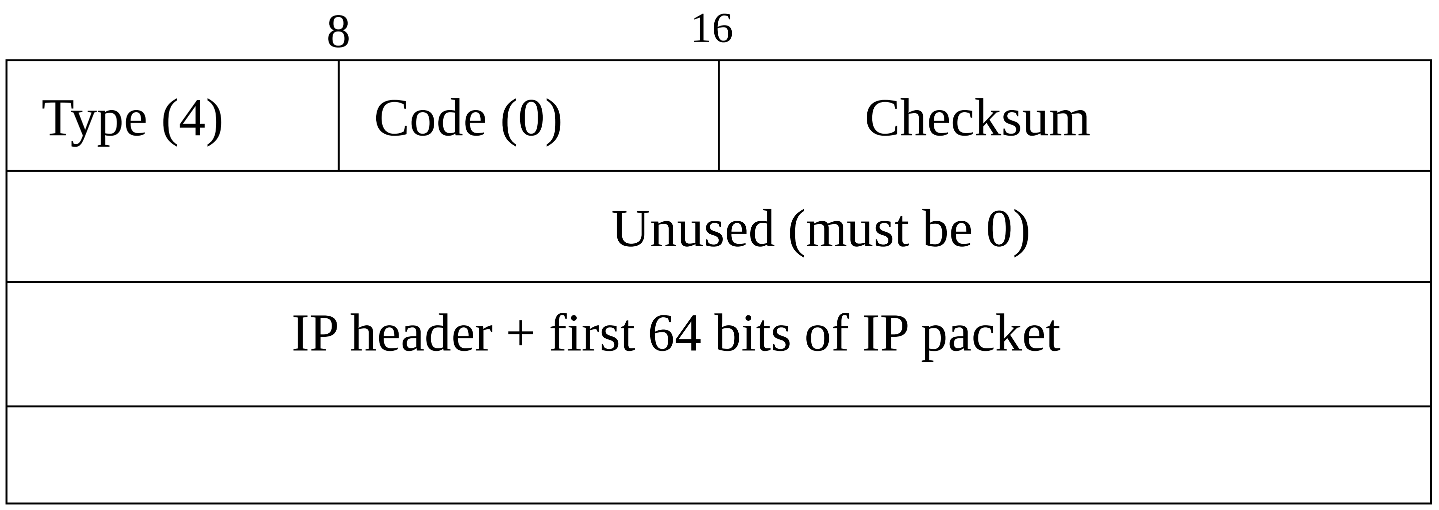


Example Code Values

Code	**Meaning**
0	network unreachable
1	host unreachable
2	protocol unreachable
3	port unreachable
4	fragmentation needed and DF set
5	source route failed
6	destination network unknown
7	destination host unknown

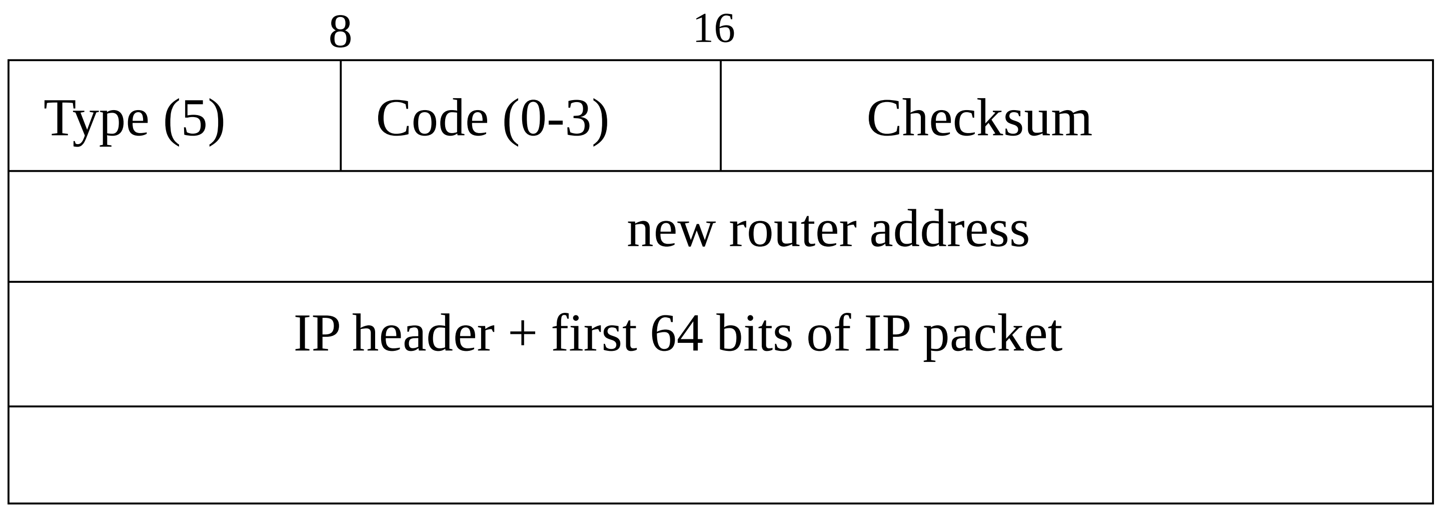
Source Quench

- Report congestion to source
- Sent when packet is dropped due to buffer overflow
- Source should reduce flow



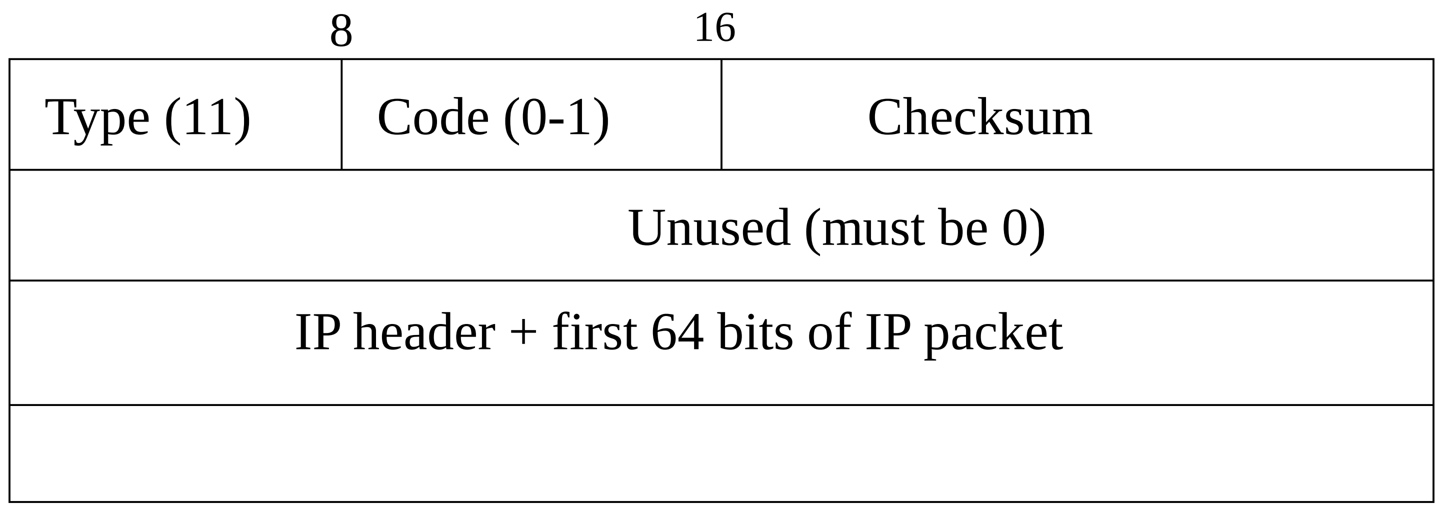
Route Redirect

- Used to let source know a better route to use for the destination address
- Packet is still forwarded



Time Exceeded for Datagram

- Used to detect circular or very long routes
- Sent when router discard packet because TTL = 0 (code 0) or fragment reassembly timer expires (code 1)
- Basis of *traceroute* tool



Parameter Problem

- Sent for any other errors that cause packet to be discarded (ex., incorrect option field)

8 16

Type (12)	Code (0-1)	Checksum
Pointer	Unused (must be 0)	
IP header + first 64 bits of IP packet		

Timestamp Request/Reply

- May be used for clock synchronization

8 16

Type (13/14)	Code (0)	Checksum
Identifier		Sequence Number
Originate Timestamp		
Receive Timestamp		
Transmit Timestamp		

Address Mask Request/Reply

- May be used to obtain subnet mask of local network at boot

	8	16
Type (17/18)	Code (0)	Checksum
Identifier		Sequence No
Address Mask		

Network Address Translation (NAT)

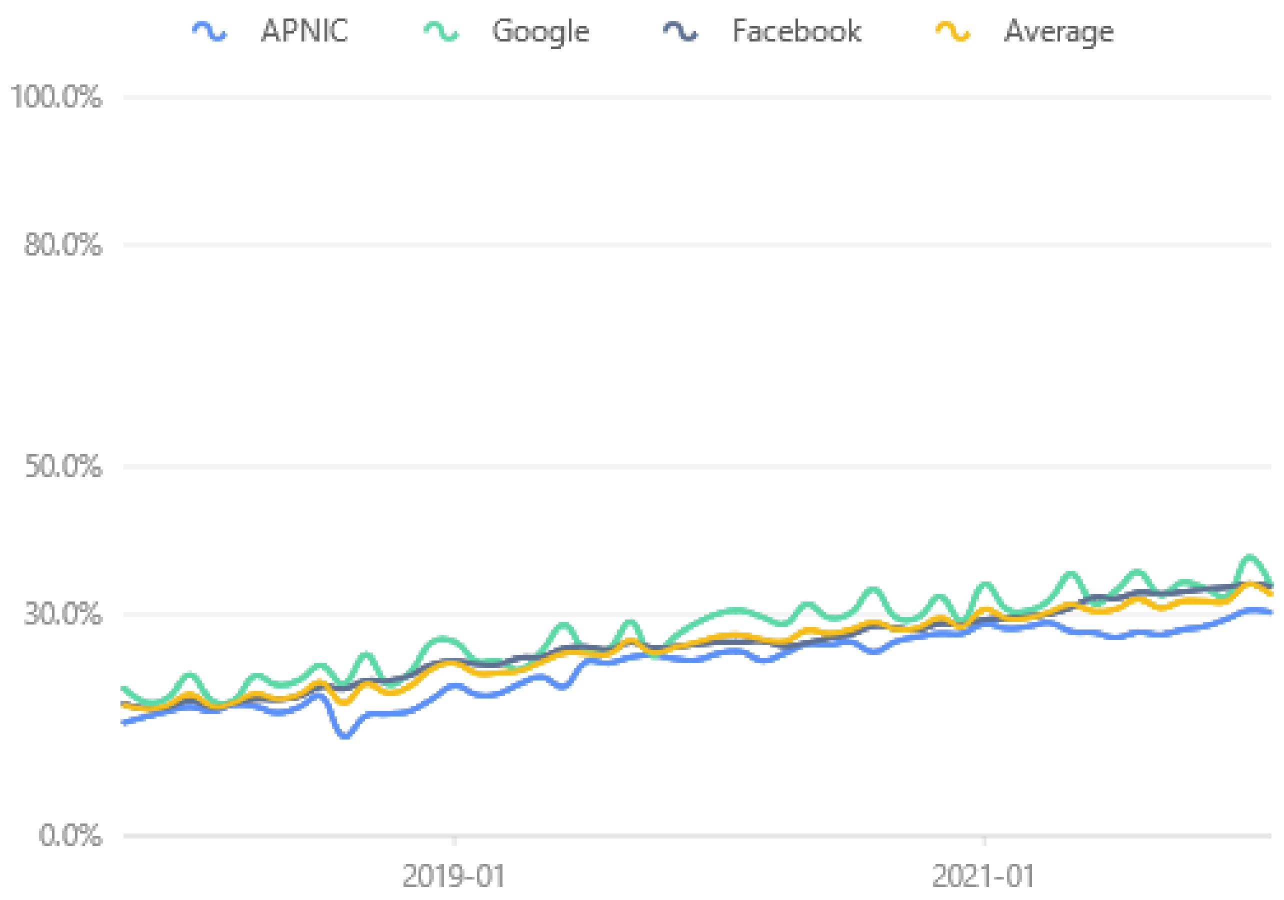
The Status of IPv4 Address

- IPv4 address space is limited
- No. of devices connected to the internet has exploded
  - Tens of billions of devices, and growing fast
- It was feared from early 2000’s that we will run out of IPv4 addresses
- And we should have, even if each device needed one IP address
- Why haven’t we?
  - Better allocation policies
  - IPv6
  - Use of private IPs

IPv6

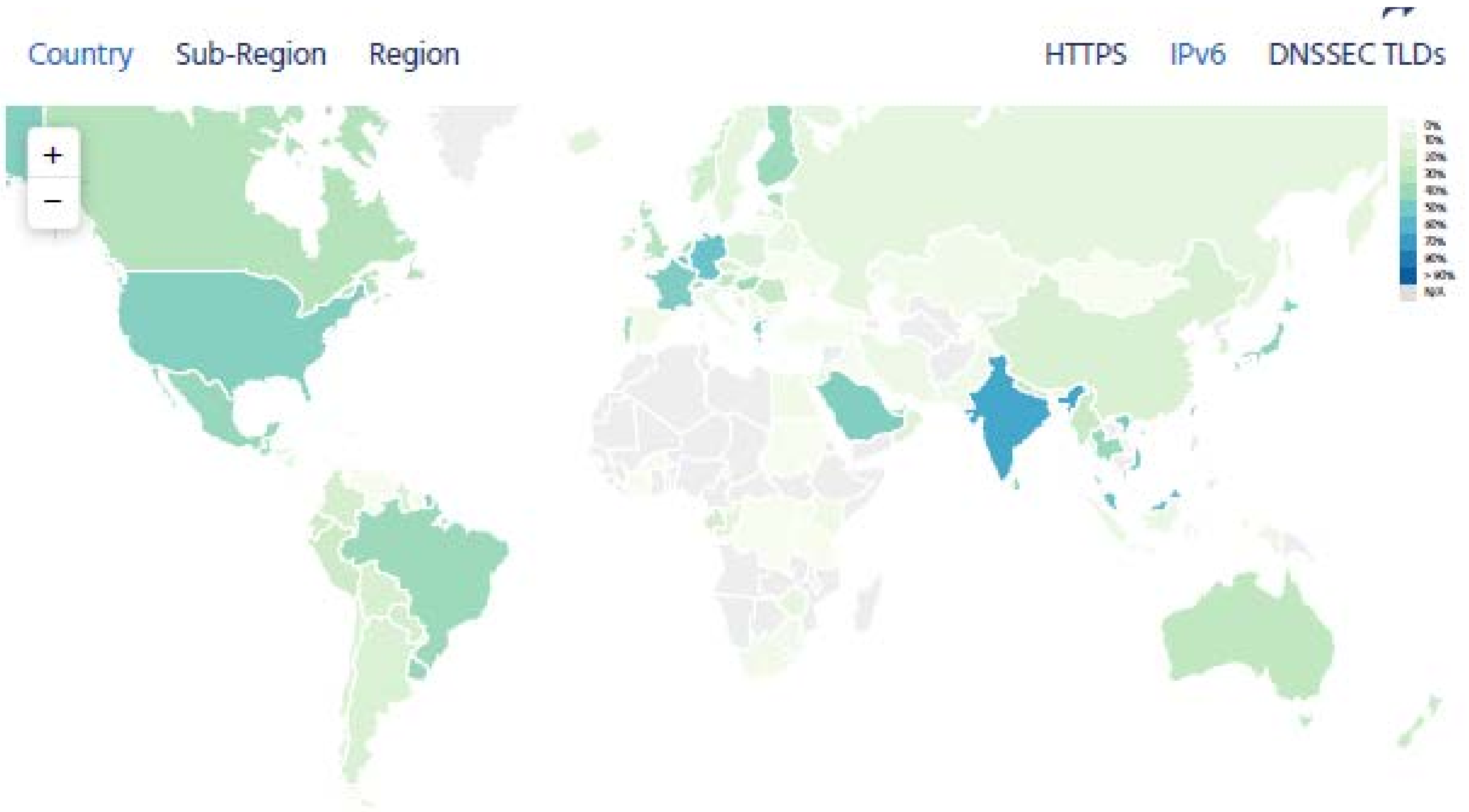
- Next generation IP protocol
- Allows for 128 bit addresses
  - Large enough for any foreseeable need
  - Other features also, but we will not cover here
- In early 2000's it was believed that IPv4 addresses will run out and the entire internet will transition to IPv6
- Hasn't happened, though there is different level of IPv6 penetration in different regions
  - Only about 34% of the top 1000 websites world-wise support IPv6 (Source: https://pulse.internetsociety.org/)

Percentage of IPv6 Traffic in Google, Facebook, APNIC



Source: https://pulse.internetsociety.org/

IPv6 Adoption by Region



Source: https://pulse.internetsociety.org/

- So IPv4 is still very much here, and will be
- But allocation policies are stricter, cannot just get a large block of IP addresses from the RIRs or from ISPs
- So how do we connect a very large number of devices with IPv4 addresses?
  - Use Private IP addresses

Using Private IP Addresses

- Think of your institute
- Internally, every machine gets a 10.*.*.* address from DHCP, which is a private IP
- The 10.*.*.* addresses never go out of IIT Kharagpur
  - No IP packet leaving or entering IIT Kharagpur has 10.*.*.* as either source address or destination address in the IP header
- The end network device (NAT-capable router for example) connecting IIT Kharagpur to the external world replaces the 10.*.*.* address with a public IP by a technique called NAT (Network Address Translation)
- Lets take an example

- Suppose your laptop in hostel has IP address 10.40.30.5.
  - Initially assume TCP connections only
- You connect to google with IP address 142.250.189.206, port 443
- Your IP packet goes with source IP = 10.40.30.5, source port = 20000 (say, assigned by system), destination IP = 142.250.189.206, destination port = 443
- The NAT-capable device intercepts the packet
- Suppose public IP used by IIT Kharagpur is 203.110.245.242

- NAT (Network Address Translation)
  - The device replaces the source IP 10.40.30.5 with 203.110.245.242 in the IP packet before sending it
  - Remembers the mapping (<10.40.30.5>, <142.250.189.206>)
  - When a packet is received with destination IP 203.110.245.242 and source IP = 142.250.189.206, the mapping is looked up, the destination IP is changed to 10.40.30.5, and routed inside the IIT network to your laptop
  - So one network address (private) is replaced by another network address (public)
  - There can be a pool of public IPs instead of one

- Problem: Allows only one internal host to make one connection to the internet at one time with a single IP
  - If 5000 people want to connect at the same time, NAT will need 5000 public IPs (anything > than the number of public IPs in the pool)
  - Getting so many public IPs is not feasible, does not scale
- Solution: Use port addresses also in NAT
  - Idea: Public IP addresses are scarce, but port addresses are plenty
    - Reuse the same public IP but with a different port
  - Also called NAPT (Network Address and Port Translation)

An Example of Using Ports in NAT

- In the earlier example, suppose another user connects from his/her laptop to the same site from 10.100.20.5 with port 30000
- Network device does the following:
  - When packet from 10.40.30.5 is received
    - Assigns a new port, say 34780
    - Creates the mapping (<10.40.30.5, 20000>, <142.250.189.206, 443>, <34780>)
    - Changes source IP to 203.110.245.242, source port to 34780 in the packet
    - Sends the packet

- When packet from 10.100.20.5 is received
  - Assigns a new port, say 34781
  - Creates the mapping (<10.100.20.5, 30000>, <142.250.189.206, 443>, <34781>)
  - Changes source IP to 203.110.245.242, source port to 34781 in the packet
  - Sends the packet

Internal IP	**Internal Port**	**External IP**	**External Port**	**NAT Port Assigned**
10.40.30.5	20000	142.250.189.206	443	34780
10.100.20.5	30000	142.250.189.206	443	34781

- When a packet is received with say source IP = 142.250.189.206, source port = 443, destination IP = 203.110.245.242, destination port = 34781
  - Look up the table see that the fields match the entry for 10.100.20.5
  - Change destination IP to 10.100.20.5
  - Change destination port to 30000
  - Send the packet to the internal host
- So the laptops do not see the public IP and the ports 34780/34781
- The external network does not see the 10.*.*.* addresses
- A single public IP can support multiple connections by changing port numbers

- We have assumed only TCP connections so far. What if there are other protocol packets?
  - One more field, protocol, added to the table
- What if two connections from internal machine uses the same client port
  - NAT port assigned will be different anyway, so packet will go with distinct source port
- How is the mapping table built?
  - Primarily by connections initiated from inside to outside
  - Other methods are there, including static mapping of some private IPs to fixed public IP
- Other uses of NAT
  - Organization has more machines than public IP, but only a small number of them access the internet at one time
  - Hiding internal addresses from the external world