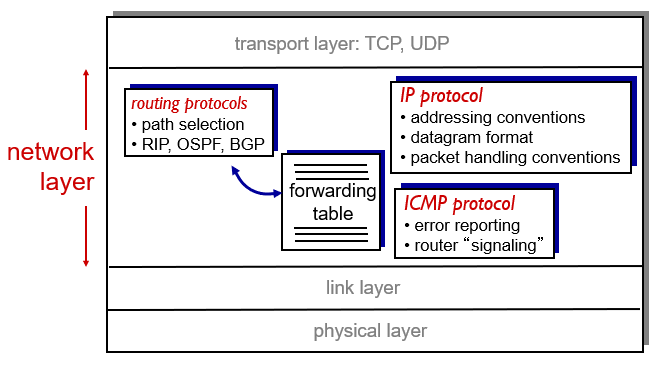
# Network Layer – Part 2

## Network Layer: Goals

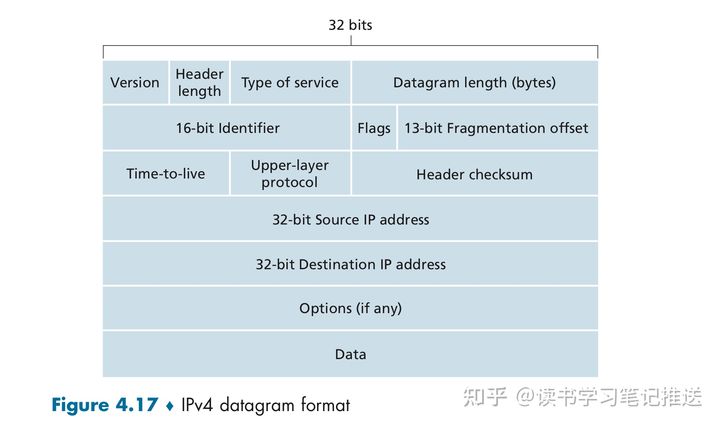
* Understand principles behind network layer services
  + Forwarding vs routing
  + Addressing
  + How a router works
  + Routing (path selection)
  + Helpful resources
    - <https://zhuanlan.zhihu.com/p/368111305>

## The Internet network layer

* network layer functions of hosts and routers…



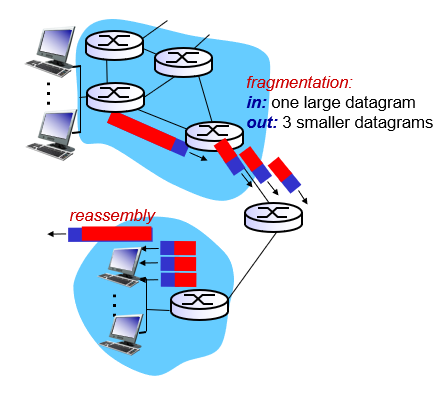
## IPv4 Datagram Format



* **[Version number]**: these 4 bits specify the IP protocol version of the datagram.
* **[Header Length]**: these 4 bits determine where in the IP datagram the payload actually begins
  + e.g. the transport-layer segment being encapsulated in this datagram
* **[Type of service]**: 8 bits are included in the Ipv4 header to allow different types of IP datagrams to be distinguished from each other
  + e.g. real-time datagrams/non-real-time traffic
* **[Datagram length]:** This is the total length of the IP datagram (header + data), measured in bytes
  + 16 bits long => maximum size of IP datagram is bytes
  + even though datagrams are rarely larger than 1500 bytes
* **[16-bit Identifier, Flags, 13-bit Fragmentation offset]:** A large IP datagram is broken into several smaller IP datagrams which are then forwarded independently to the destination, where they are reassembled before their payload data is passed up to transport layer at destination host
  + Ipv6 does not allow for fragmentation (but this is being resolved).
* **[Time-to-live(TTL)]**: This field ensures that datagrams do NOT circulate forever
  + this field is decremented by 1 each time the datagram is processed by a router
  + if the TTL field reaches 0, a router must drop that datagram
* **[Upper-Layer Protocol]**: The value of this field indicates the specific transport-layer protocol to which the data portion of this IP datagram should be passed to,
  + TCP: 6
  + UDP: 17
  + the protocol number in the IP datagram is the glue that binds the network and transport layers together
  + Recall: the port number in the transport-layer segment is the glue that binds the transport and app layers together
* **[Header checksum]**: to aid a router in detecting bit errors in a received IP datagram
* **[Source and destination IP addresses]**: when a source creates a datagram,
  + it inserts its IP address into the *source IP address* field, and
  + inserts the address of the final destination into the *destination IP address* field
  + often the source host determines the destination address by a DNS lookup
* **[Options]**: the options fields allow an IP header to be extended.
  + timestamps, record route taken, specify list of routers to visit
  + since datagram headers can be of variable length, one cannot determine from before where the data field will start
  + IP options were NOT included in the Ipv6 header since considerations for IP processing time can vary greatly in high-performance routers and hosts
* **[Data(payload)]**: the fundamental reason for the datagram in the first place!
  + the Data field of the IP datagram contains the transport-layer segment (TCP or UDP) to be delivered to the destination
  + or other types of data, e.g. ICMP messages
* **[Overhead]**:
  + an IP Datagram has 20 bytes of header (assuming NO Options)
  + 20 bytes of TCP header if the datagram carries a TCP segment
  + = 40 bytes of header along with the application-layer message overhead

## IP fragmentation, reassembly

* Network links have max. transfer unit (MTU) – size of the largest PDU that can be communicated in a single network layer transaction
  + relates to but is not identical to maximum frame size that can be transported on the data link layer; e.g. Ethernet frame
  + different link types, different MTUs
* Large IP datagram divided into smaller datagrams (“fragmented”) within network
  + one datagram becomes several smaller datagrams
  + “reassembled” only at final destination
  + IP header bits used to identify and order related fragments

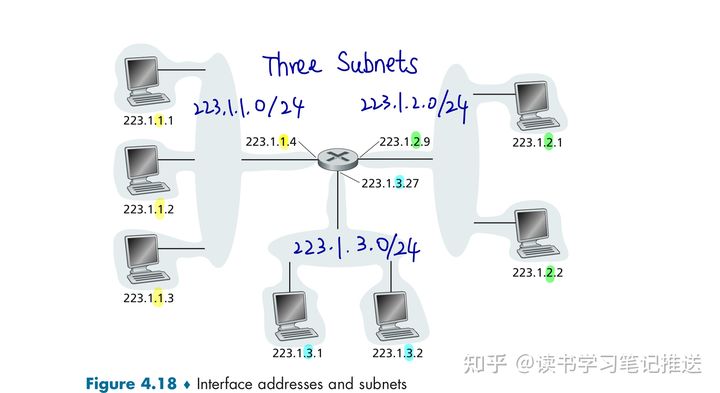


## IP fragmentation, reassembly: example

* Example:
  + Original datagram:
    - length = 4000 bytes (includes **20**/40 bytes overhead = IP + TCP head)
    - actual data = 3960 bytes (or **3980** bytes according to GeeksforGeeks)
    - DF frag flag = 0 (off; can be fragmented)
    - offset = 0 (no previous fragments)
    - frag flag = 0 + offset = 0 means original, unfragmented datagram
  + MTU (could be an Ethernet frame)
    - length = 1500 bytes (includes **20**/40 bytes overhead = IP + TCP head)
* After breaking the datagram down into the different fragments
  + Fragment 1
    - length = 1500 bytes (includes 20 bytes overhead)
    - MF frag flag = 1 (on; more fragmented datagrams)
    - offset = 0 (no previous fragments)
    - frag flag = 1 + offset = 0 means first fragment in datagram
  + Fragment 2
    - length = 1500 bytes (includes 20 bytes overhead)
    - MF frag flag = 1 (on; more fragmented datagrams)
    - offset = 185 (1480 bytes of data in previous fragment)
  + Fragment 3
    - length = 1040 bytes (includes 20 bytes overhead)
    - MF frag flag = 0 (off; no more fragments left)
    - offset = 370 (2960 bytes of data in previous 2 fragments)
* Notes:
  + The max capacity of IPv4 datagrams is 65535 bytes (including header), but there are many different transmissions links in the network and the MTU corresponding to each link is different.
  + To adapt to different links, IPv4 is designed to allow each network node to perform data fragmentation on data fragments.
  + Whether or not to do fragmentation on a datagram is determined by a special byte—the 3-bit frag flag in the IPv4 header.
    - Bit 0 is stands for Reserved (R) = usually unset
    - Bit 1 stands for Don’t Fragment (DF) = usually unset
      * DF bit = 0 means fragmentation can be done
      * DF bit = 1 means fragmentation cannot be done
    - Bit 2 stands for More Fragments = last fragment set to 0; others to 1
      * MF bit = 0 means that the fragmented datagram is the last one.
      * MF bit = 1 means there are more fragmented datagrams.
  + All the fragments belonging to the same datagram will have the same ID number, so it’s possible to know which fragment belongs to which datagram
  + Offset tells how much data is in the previous fragment; expressed in multiples of eight here.
  + The last offset and last data size are used to calculate the total data size.
  + <https://www.geeksforgeeks.org/ipv4-datagram-fragmentation-and-delays/>

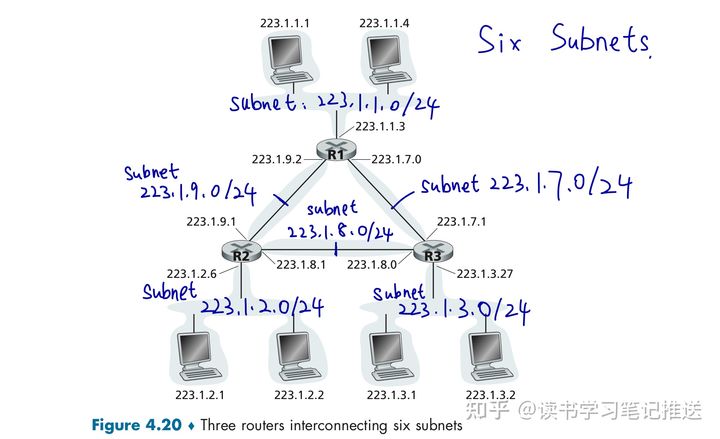
## IPv4 Addressing

* **[Interface]**: the boundary/connection between the host or router and the physical link
  + Host typically has one or two interfaces (e.g. wired Ethernet, wireless 802.11)
  + A router has multiple interfaces, one for each of its links to receive a datagram on one link and forward the datagram on some other link
  + IP requires each host and router interface to have its own IP Address
* An IP Address is technically associated with an interface, rather than with the Host or Router containing that Interface
* Each **[IP Address]** is 32 bits (= 4 bytes) long, and there are a total of approximately **4 billion** possible IP addresses.
  + these addresses are typically written in **dotted-decimal notation**
  + e.g. the address 193.32.216.9 in binary notation is…   
    11000001 00100000 11011000 00001001
* A portion of an Interface's IP address will be determined by the **[subnet]** to which it is connected



## Subnet

* **[Subnet]** is the network interconnecting multiple host interfaces and ONE router interface without passing through an intervening router
* IP addressing assigns an address to this subnet
  + e.g. 223.1.1.0/24, where /24 is a **[subnet mask]** that indicates that the leftmost 24 bits of the 32-bit quantity define the subnet address.
  + Subnet Part: Devices in same subnet have common high order bits
  + Host Part: remaining low order bits
* Recipe to define the subnets in the system:
  + Detach each interface from its host or router,
  + creating "islands" of isolated networks
  + with interfaces terminating the end points of the isolated networks
  + Each isolated networks is called a *subnet*
* In principle, the different subnets could have quite different subnet addresses
* In practice, their subnet addresses often have much in common



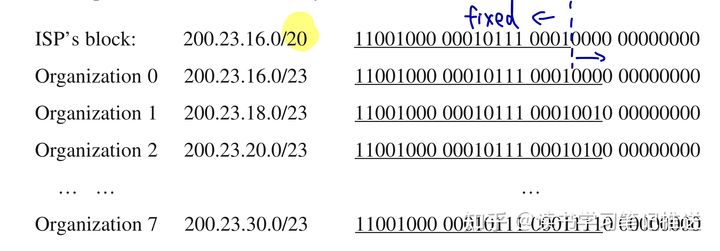
## CIDR: Classless InterDomain Routing

* CIDR generalizes the notion of subnet addressing
  + subnet portion of address of arbitrary length
  + address format: a.b.c.d/x, where x is the number of bits in the subnet portion of the address

## How to Get Subnet Part of IP Address

Q1: How does a network get IP address for itself (i.e. the subnet part of the address)?

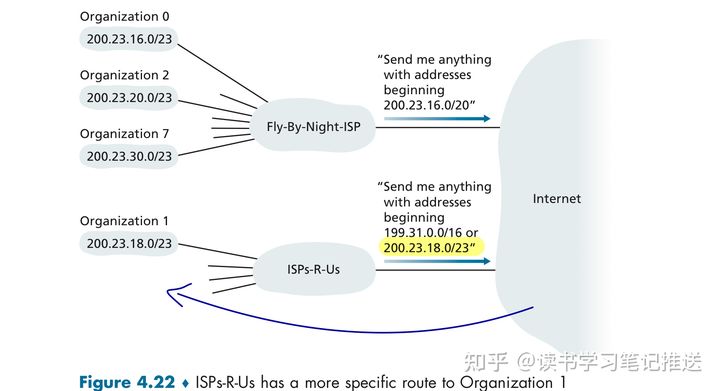
* in order to obtain a block of IP addresses for devices within in org’s subnet, a network admin might first contact an Internet Service Provider (ISP),
  + which would provide addresses from a larger block of addresses that had already been allocated to the ISP; e.g. 200.23.16.0/20
  + the ISP divides its address block^ into 8 equal-sized contiguous address blocks and assigns to eight orgs supported by this ISP



* Or by ICANN (Internet Corp for Assigned Names and Numbers:
  + allocates IP addresses through 5 Regional Registries (RIRs), which then allocate to local registries, which distribute to ISPs and so on
  + manages DNS root zone, including top-level domain (TLD) management
  + assigns domain names (.com, .edu, etc.), resolves disputes

## Hierarchical IP Addressing

* The ISP (Fly-By-Night-ISP) advertises to the outside world that it should be sent any datagrams whose first 20 address bits match 200.23.16.0/20
  + the rest of the world does NOT need to know there are 8 organizations within the address block 200.23.16.0/20
* **[Route Aggregation]**: Use a single prefix to advertise multiple networks
  + works well when addresses are allocated in a hierarchical manner



## How to Get Host Part of IP Address

Q2: How does a host get IP address within its subnetwork (i.e. host part of address)?

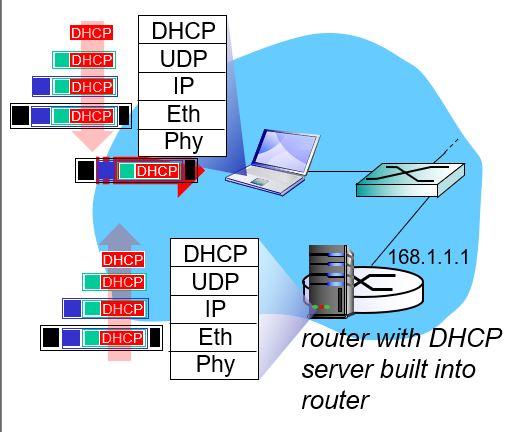
* *Hard-coded* by system admin in a file
  + Windows: control-panel → network → configuration → tcp/ip → properties
  + Unix: /etc/rc.config
* DHCP: Dynamic Host Configuration Protocol
  + dynamically get address from DHCP server
  + “plug and play” operation

## DHCP: Dynamic Host Configuration Protocol

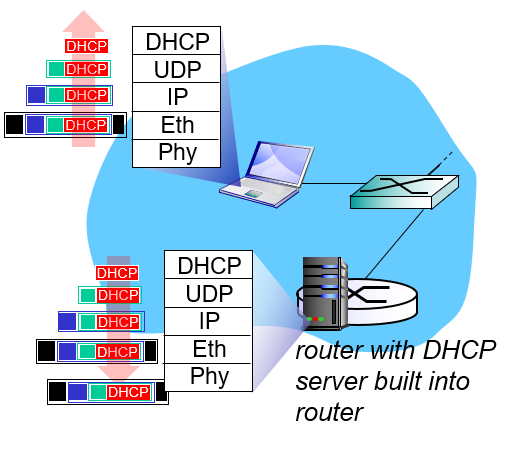
* Goal: allows a host to obtain (be allocated) an IP address automatically when it connects to the network
  + gives more info than just IP address
    - subnet mask
    - the address of its first-hop router (default gateway)
    - the address of its local DNS server
  + can renew its lease on address in use
  + allows reuse of addresses (only hold address while connected/on)
  + support for mobile users who want to join/leave network
  + “plug-and-play” or “zero-conf” protocol
* DHCP is a client-server protocol!
  + Arriving DHCP client needs IP address and other network config ingo
  + Each subnet will have a DHCP server
    - if NO server is present on subnet, a DHCP *relay agent* (i.e. router) that knows the address of a DHCP server is needed

## DHCP Process

* DHCP is a four-step process for a *newly arriving client*.
  + Step 1 and 2 are optional if a client remembers and wishes to reuse a previously allocated network address
* Step 1: DHCP Server Discovery
  + To find a DHCP server with which to interact with…
    - client sends a *DHCP discover* msg within a UDP packet to port 67
    - DHCP in client creates an IP datagram containing its *DHCP discover* msg
      * along with the *broadcast* destination IP address of 255.255.255.255
      * and a “this host” source IP address of 0.0.0.0
    - DHCP in client passes the IP datagram to the link layer, which then broadcasts this frame to ALL nodes attached to the subnet
* Step 2: DHCP Server Offer(s)
  + A DHCP server (receiving a *DHCP discover* msg) responds to the client with a *DCHP offer* msg…
    - that is broadcast to ALL nodes on the subnet
    - again using the IP broadcast address 255.255.255.255
  + Each server offer msg contains:
    - transaction ID of the received discover message
    - the proposed IP address for the client
    - the network mask and an IP address lease time (the amount of time to be valid)

  
*Steps 1 and 2*

* Step 3: DHCP Request
  + The newly arriving client will choose from among one or more *server offers*
    - and respond to its selected offer with a *DHCP request* msg
    - echoing back the config parameters
  + *DHCP request* message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
  + Ethernet frame broadcast on LAN, received at *router* running DHCP Server
  + Ethernet demux'ed to IP, IP demux’ed to UDP, UDP demux'ed to DHCP
* Step 4: DCHP Acknowledgement (ACK)
  + The server responds to the *DHCP request* msg with a *DHCP ACK* msg
    - confirming the requested parameters
  + Encapsulated DCHP *server reply* forwarded to client, demuxing up to DHCP at client
  + Client now knows its IP address, name and Ip address of DNS server, IP address of its first-hop router (or gateway)

  
*Steps 3 and 4*

## Are there enough 32-bit IP addresses?

* ICANN allocated LAST chunk of IPv4 addresses to Regional Registries in 2011
* NAT helps IPv4 address space exhaustion
* IPv6 has 128-bit address space

## DHCP client-server scenario

