**PART 2: Network Layer**

**Subnet: 223.1.1.0/24**

**IP Addressing**

**IP Address** is a 32-bit identifier for a host and router interface

**Interface** = connection between a host/router with a physical link

* Router w/ physical link: typically have multiple interfaces
* Host w/ physical link: typically have one or two interfaces

IP addresses associated with each interface

**Wired Ethernet interfaces connected by Ethernet switches**

* 223.1.1.1 | 223.1.1.2 | 223.1.1.3 etc.

**Subnets**

**Subnet part** = high order bits

**Host part** = low order bits

**Wireless WiFi interfaces connected by WiFi base station**

A **Subnetting** is dividing a major network into multiple smaller networks

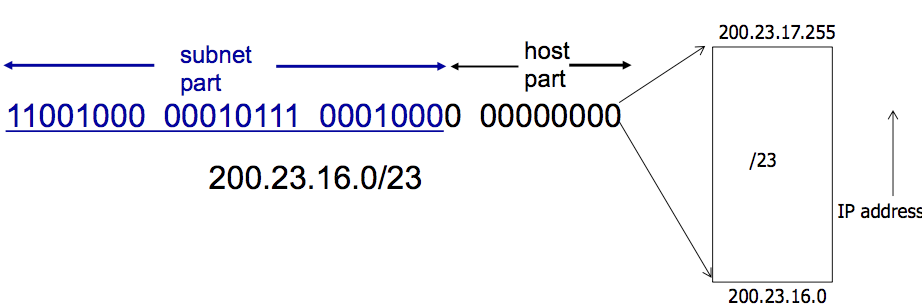
* They can physically reach each other without an intervening router
* To determine the subnets, detach each interface from its host or router,  
  creating islands of isolated networks.
* Each isolated network = **Subnet**

*Having an organisation divide its network into subnets allows it to be connected to the internet with a single shared network address. Without subnets, an organisation could get multiple connections to the internet, which would require an unnecessary use of limited network numbers that the internet has to assign. It also means that Internet routing tables outside of the organisation would have to know about / manage routing that should be handled within an organisation.*

**Today’s Internet Addressing**

**Classless InterDomain Routing (CIDR)**

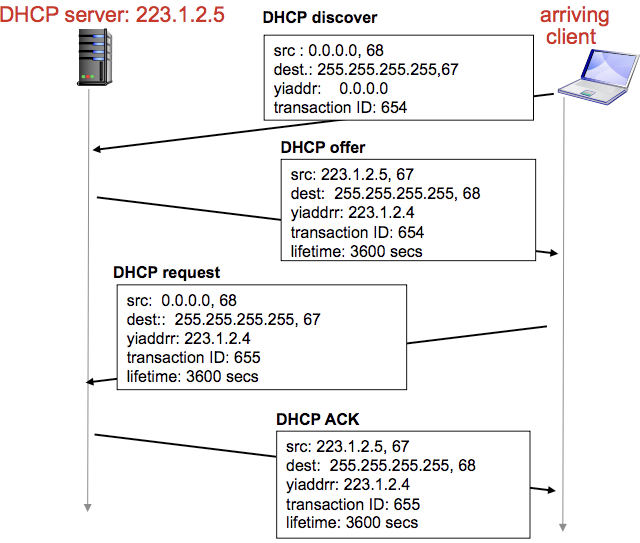
* The subnet portion of an address is arbitrary length
* Address format: [ a . b . c . d / x ], where X is #bits in the subnet portion of the address



A **Subnet Mask** separates the IP address into the NETWORK PART and HOST PART of the address.

Example:

|  |  |  |
| --- | --- | --- |
|  | **Decimal Address** | **Binary** |
| **IP Address** | 223.1.1.2 | 11111101 00000001 00000001 00000010 |
| **Subnet Mask** | 255.255.255.0 | 11111111 11111111 11111111 00000000 |
|  | | **IP & Subnet Mask** |
| **Network part** | 223.1.1.0 | 11111101 00000001 00000001 00000000 |
|  | | **Remainder Bits** |
| **Host part** | 0.0.0.2 | 00000000 00000000 00000000 00000010 |

How does a HOST get an IP address?

1. Hard-coded by a system admin in a file
2. **DHCP**: **Dynamic Host Configuration Protocol**

**DHCP** allows a host to dynamically obtain its IP address  
from a network server when it joins the network

* Host can renew its lease on address in use
* Allows reuse of the addresses  
  (only holds address while “connected” / on)
* Support for mobile users who want to join the network

DHCP Overview

* Host broadcasts “**DHCP discover**” message
* DHCP server responds with “**DHCP offer**” message
* Host requests IP address: “**DHCP request**” message
* DHCP server sends address: “**DHCP ack**” message

DHCP can return more than just allocated IP address on subnet:

* Address of first-hop router for client
* Name and IP of DNS server
* Network mask (indicating network vs host portion of address)

**DHCP Process**

1. **DHCP Request**
   1. Connecting device (client) needs**: [ IP address ] [ address of first-hop router ] [ address of DNS server ]**
   2. DHCP request is encapsulated in UDP 🡪 then encapsulated in IP 🡪 then encapsulated in 802.1 Ethernet  
      (Ethernet controls how data is transmitted over a Local Area Network –A network that covers a small area)
   3. Ethernet frame is broadcasted on LAN
   4. Frame is received at the router running the DHCP server.
   5. Ethernet demuxed to IP 🡪 IP demuxed to UDP 🡪 UDP demuxed to DHCP request  
      (Demux = Demultiplex i.e. converting data streams back into the original, separate channels)
2. **DHCP Ack / Reply**
   1. DHCP server formulates ACK: **[ client IP ] [ first-hop router IP for client ] [ DNS server name & IP ]**
   2. Encapsulation of DHCP server and the frame is forwarded to the client, then the client demuxes back to DHCP.
   3. Client will now know its IP address, name and IP address of DNS server, IP address of its first-hop router.

Further DHCP details

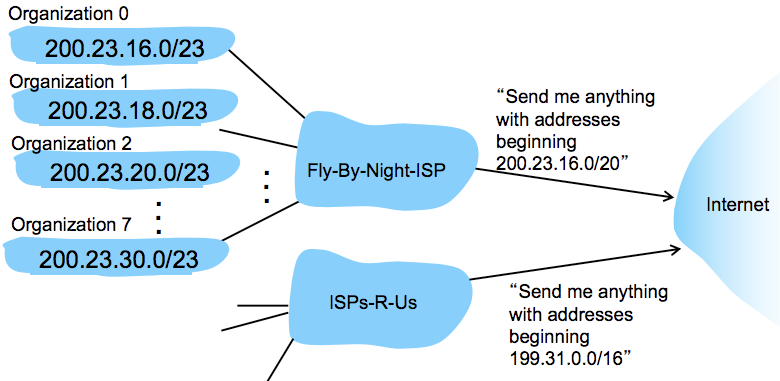
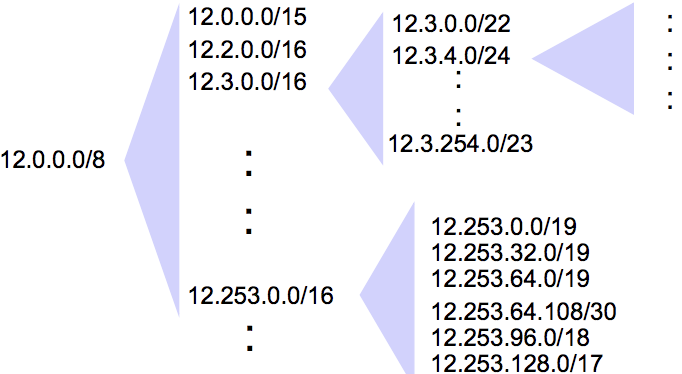
* DHCP uses UDP and port numbers **67 (server side port)** and **68 (client side port)**
* Usually the **MAC address is used to identify clients**
  + DHCP server can be configured with a “registered list” of acceptable MAC addresses
* DHCP message includes IP address, length of lease, subnet mask, DNS servers, default gateway
* DHCP security holes:
  + DoS attack by exhausting pool of IP addresses
  + Masquerading as a DHCP server
  + Authentication for DHCP

How does a Network get the subnet part of its IP address? It gets allocated a portion of its provider ISP’s address space. Example:

* ISP: 11001000 00010111 0001**0000** 00000000 🡪 200.23.16.0/20  
  ORG #0: 11001000 00010111 0001**0010** 00000000 🡪 200.23.16.0/23  
  ORG #1: 11001000 00010111 0001**0100** 00000000 🡪 200.23.18.0/23  
  …. … … … …  
  ORG #7: 11001000 00010111 0001**1110** 00000000 🡪 200.23.30.0/23

**CIDR** uses hierarchical address allocation: addresses are allocated in continuous chunks (prefixes)

* Hierarchical addressing allows efficient “advertisement” of routing information



What happens when organisation #1 wants to switch from Fly-By-Night to another ISP-R-Us?

* Org #1 keeps its IP addresses in 200.23.18.0/23
* Fly-By-Night continues to advertise address block 200.23.16.0/20 and ISP-R-Us continues with 199.31.0.0/16
* However, **ISP-R-Us now will also advertise the more specific block of addresses of Organisation #1 200.23.18.0/23**
* When other routers in the larger internet see address blocks 200.23.16.0/20 (from Fly-By-Night) and 200.23.18.0/23 (from ISP-R-Us) and want to route to an address in Organisation #1 block (200.23.18.0/23), they will use a **longest prefix matching rule** **and route towards ISP-R-Us, as it advertises the** **longest (more specific) address prefix that matches the destination address**.
  + Routers in the internet will have two entries in their routing tables and will match the destination address (Organisation #1’s address) with the entry that has the longer / more specific prefix  
    (entry belonging to ISP-R-Us)

How does an ISP get a block of addresses?

* **ICANN** **(Internet Corporation for Assigned Names and Numbers)**: An organisation that allocates addresses, manages DNS, assigns domain names, resolves disputes.
* **Regional Internet Registries (RIR)** act as intermediaries e.g. **APNIC (Asia Pacific Network Information Centre)**

Example:

* ICANN gives APNIC several /8s
* APNIC gives Telstra one /8 🡪 129.**0**/8 with Network Prefix: **10000001**
* Telstra gives UNSW a /16 🡪 129.**94**/16 with Network Prefix: 10000001.**01011110**
* UNSW gives CSE a /16 🡪 129.94.**242**/24 with Network Prefix: 10000001.01011110.**11110010**
* CSE gives me a specific address 🡪 129.94.242.**51** with Network Prefix: 10000001.01011110.11110010.**00110011**

**STUDY THIS EXAMPLE**

Question: Do these two subnets have overlapping IP addresses?

* Subnet 1: **10.1.2.0/23 |** Subnet 2: **10.1.3.248/30**
* STEP 1: Convert to binary (RED = Subnet part)
  + 10.1.2.0/23 🡪 00001010.00000001.00000010.00000000
  + 10.1.3.248/30 🡪 00001010.00000001.00000011.11111000
* STEP 2: Apply Subnet Mask
  + Subnet 1 Mask = 23 bits set to 1 **00001010.00000001.0000001**0.00000000 Bitwise &  
    11111111.11111111.11111110.00000000  
    --------------------------------------------------------  
    00001010.00000001.00000010.00000000 **= 10.1.2.0 start of range**
  + Subnet 2 Mask = 30 bits set to 1  
    **00001010.00000001.00000011.111110**00 Bitwise &  
    11111111.11111111.11111111.11111100  
    --------------------------------------------------------  
    00001010.00000001.00000011.11111000 = **10.1.3.248 start of range**
* STEP 3: Calculate SIZE of the address range = Complement of the Subnet Mask
  + ~ Complement of Subnet 1 Mask  
    11111111.11111111.11111110.00000000 ~  
    --------------------------------------------------------  
    00000000.00000000.00000001.11111111 = 0.0.3.255 end of range
  + ~ Complement of Subnet 2 Mask  
    11111111.11111111.11111111.11111100 ~  
    --------------------------------------------------------

00000000.00000000.00000000.00000011 = 0.0.0.3 end of range

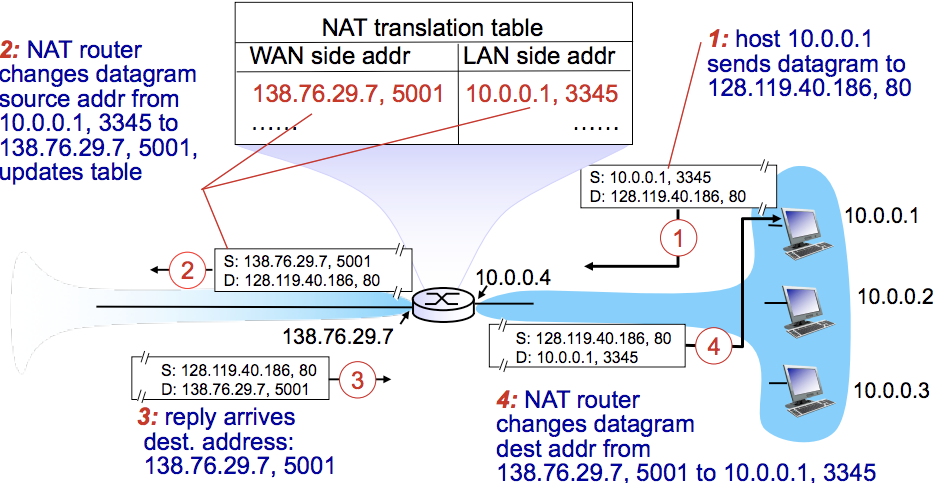
* STEP 4: Compare ranges
  + Subnet 1 Range = **10.1.2.0 🡪 10.1.3.255**
  + Subnet 2 range = **10.1.3.248 🡪 10.1.3.3**
* Subnet 2 IP address range is completely within the Subnet 1 address range = **OVERLAP EXISTS**

**Private Addresses and Network Address Translation (NAT)**

10.0.0.0/7 | 172.16.0.0/2 | 192.168.0.0/16

These addresses cannot be routed. Anyone can use them. They are typically used for **Network Address Translation (NAT)**

* Network Address Translation allows a single device, such as a Router, to act as an agent between the Internet (public network) and a local (private) network. Only a single unique IP address is required to represent an entire group of computers.



Advantages of NAT

* A range of addresses are not needed from an ISP. Just one IP address for all devices in the local / private network.
* Can change addresses of devices in local network without notifying the outside world
* Can change ISP without changing addresses of devices in the local network.
* 16-bit port-number field = 60,000 simultaneous connections with a single LAN-side address

NAT is controversial as routers should only process up to the Network Layer, thus violating the end-to-end argument

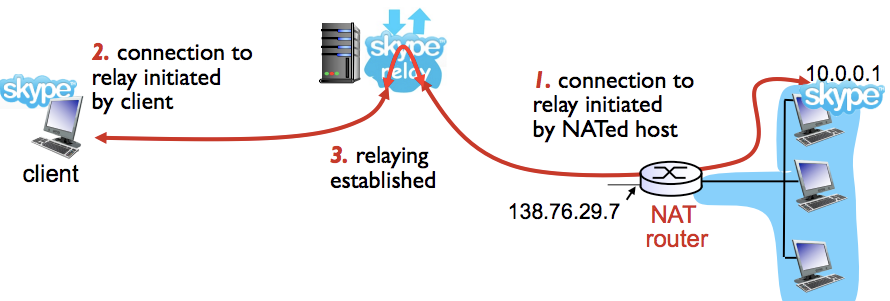
NAT issues

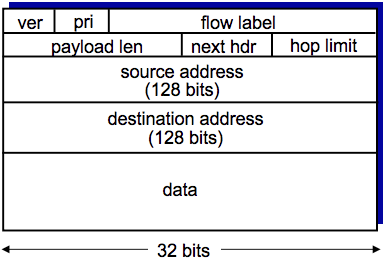
* NAT modifies port # and IP address 🡪 **requires recalculation of TCP and IP checksum**
* **Some apps embed IP addresses / port # in their message payloads**
  + For legacy protocols, NAT must look into these packets and translate the embedded IP addresses / port #
  + What if these fields are encrypted?!
* **NAT traversal problems:** A client wants to connect to a server with addr 10.0.0.1, but they can’t use that addr to connect as it is a local address.
  + SOLUTION #1: **Statically configure NAT** to forward ALL incoming connection requests at a given port to server
  + SOLUTION #2: **Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol**. Allows NAT hosts to:
    - Learn the public IP address
    - Add / remove port mapping

This solution = Automatic static NAT port map configuration (as opposed to sol 1 which is manual)

* SOLUTION #3: **Relaying (used in Skype)** where:
  + - NATed client establishes connection to relay
    - External client connects to relay
    - Relay bridges packets between the two connections

This solution = MIDDLEMAN SOLUTION





**IPv6**

**Initial Motivation:** 32-bit address space will soon be completely allocated

**Additional Motivation**: IPv6 Header format helps speed up processing / forwarding

IPv6 Datagram Format:

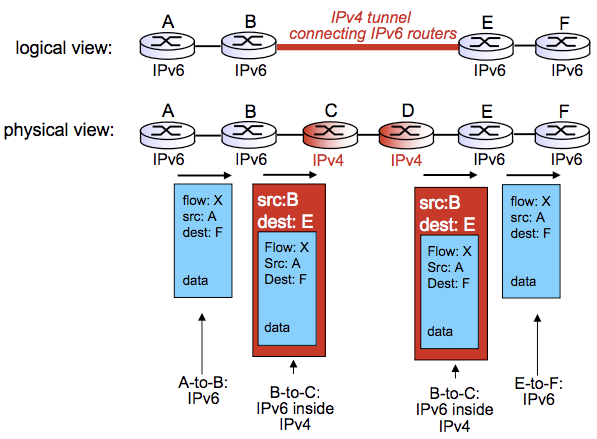
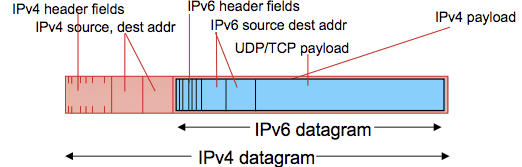
* Fixed-len 40byte header, no fragmentation allowed
* **Priority**: identify priority among datagrams in flow (traffic class)
* **Flow Label**: identify datagrams in the same flow
* **Next Header**: Identify the upper layer protocol for data

Changes from IPv4

* **Checksum** removed entirely to reduce processing time at each hop
* **Options** allowed, but outside of the header, indicated by “next header” field
* **ICMPv6**: new version of ICMP

How will networks operate with mixed IPv4 and IPv6 routers?

* **Tunneling:** IPv6 datagram is carried as a payload in an IPv4 datagram among IPv4 routers.



**Virtual Circuit Network (replaces IP Network Design)**

**Datagram Networks**

Datagram networks provides a network-layer connectionless service.

* Not setup at network layer.
* Routers have no network-level concept of a “connection” / doesn’t keep state about end-to-end connections
* Packets are forwarded using destination host address (packets btwn same source-dest pair may take diff paths)

**Virtual Circuit Networks**

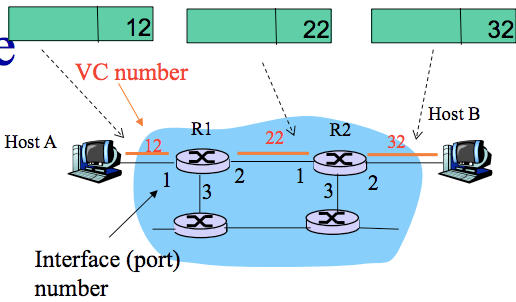
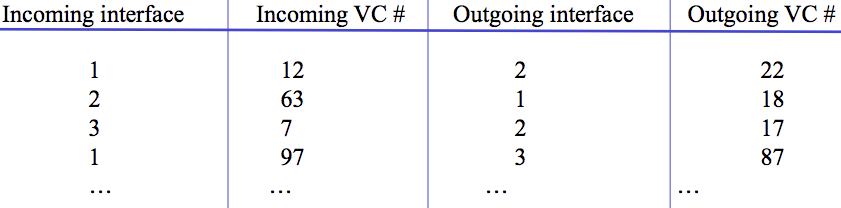
Virtual Circuit Network which provides a network-layer connection service.

* Signalling protocols used to setup and maintain teardown of Virtual Circuits
* Not used in today’s internet, but offers a great alternative for Quality of Service guarantees at a smaller scale

VC implementation: A virtual circuit consists of:

1. Path from source to destination
2. Virtual Circuit numbers, one number for each link along path
3. Entries in forwarding tables in switches along the path

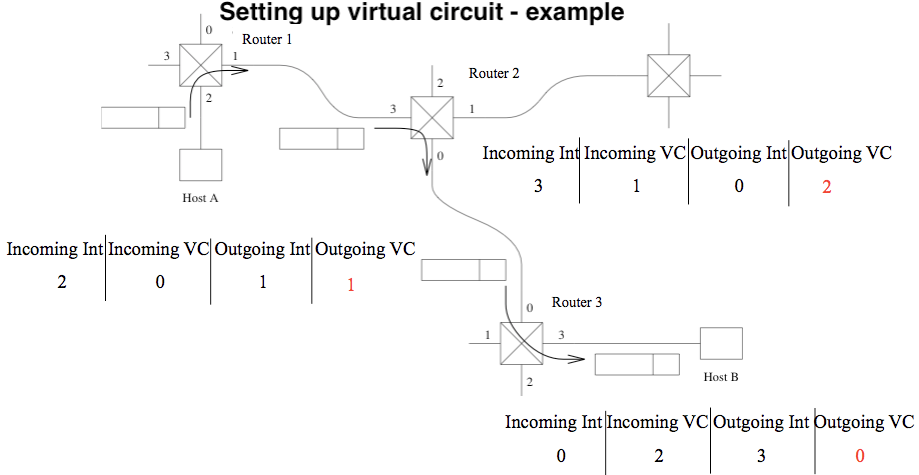
* Packet belonging to VC carries a VC number (rather than a destination host address)
* VC number may be changed on each link 🡪 New VC number comes from the forwarding table



**Forwarding Table in Router R1**(switches/routers maintain connection state info)

Setting up a Virtual Circuit

* Source host sends a setup message 🡪 setup message contains destination address
* All intermediate routers
  + Chooses an unused VC # (usually lowest available) as the incoming VC #.
  + Determine the outgoing interface (from Routing Table)
  + Create an entry in the VC table with the incoming interface, incoming VC #, outgoing interface but an empty outgoing VC #.
  + Forward the setup message to the next hop.
* When the setup reaches the destination
  + The dest chooses an available VC # as the incoming VC #
  + The chosen incoming VC # = outgoing VC # of all routers in this Virtual Circuit (except for the last router)
  + The chosen VC # is inserted in an acknowledgement of connection setup, which is sent along the reverse path.
* The intermediate routers:
  + Completes the VC table entry
  + Sends acknowledgement of connection setup upstream (reverse path back to source)
  + The acknowledgement contains the outgoing VC # that the upstream router requires

Example: Virtual Circuit Setup

**Outgoing VC should be initially EMPTY when setup message is being sent from SRC 🡪 DEST**

**Incoming VC = Outgoing VC for all**