Announcements/Reminders

- Homework 4 will cover the network layer.
- Final exam:
  - March 22\textsuperscript{nd}, 12-3pm.
  - Comprehensive.
  - Closed books/notes.
Project

- Deliverables:
  - Project demo.
  - Code (documented).
Last Class

- Finished transport layer (Chapter 3).
- Started network layer.
Today

- Network layer.
  - Chapter 4.
The Internet Network Layer
The Internet Network layer

Host, router network layer functions:

- **Routing protocols**
  - path selection
  - RIP, OSPF, BGP

- **IP protocol**
  - addressing conventions
  - datagram format
  - packet handling conventions

- **ICMP protocol**
  - error reporting
  - router "signaling"

Transport layer: TCP, UDP

Network layer
### IP datagram format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version</td>
<td>Number</td>
</tr>
<tr>
<td>header length (bytes)</td>
<td>“type” of data</td>
</tr>
<tr>
<td>max number remaining</td>
<td>Max number of remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>time to live</td>
<td>Max number of remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>upper layer</td>
<td>Upper layer protocol to deliver payload to</td>
</tr>
<tr>
<td>upper layer protocol</td>
<td>Options (if any)</td>
</tr>
<tr>
<td>32 bit source IP address</td>
<td>32 bit source IP address</td>
</tr>
<tr>
<td>32 bit destination IP address</td>
<td>32 bit destination IP address</td>
</tr>
<tr>
<td>data</td>
<td>Data</td>
</tr>
<tr>
<td>(variable length,</td>
<td>Data</td>
</tr>
<tr>
<td>typically a TCP segment)</td>
<td>Data</td>
</tr>
</tbody>
</table>

**how much overhead with TCP?**

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

**E.g. timestamp, record route taken, specify list of routers to visit.**
IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments

fragmentation: in: one large datagram out: 3 smaller datagrams
reassembly
IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>185</td>
</tr>
<tr>
<td>1040</td>
<td>x</td>
<td>0</td>
<td>370</td>
</tr>
</tbody>
</table>
IPv4 Addressing
IP Addressing: introduction

- **IP address**: 32-bit identifier for host, router **interface**
- **interface**: connection between host/router and physical link
  - router’s typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface

Example IP addresses:
- 223.1.1.1 = 11011111 00000001 00000001 00000001
  - 223
  - 1
  - 1
  - 1
Subnets

- **IP address:**
  - subnet part (high order bits)
  - host part (low order bits)

- **What's a subnet?**
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router

![Network diagram with 3 subnets](image.png)

Network consisting of 3 subnets
Why subnet?

- Isolate traffic.
  - Security.
  - Network resource utilization.
  - Contention.
Subnets

Recipe

- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a subnet.

Subnet mask: /24

Denotes that the leftmost 24 bits define subnet address. All devices on the same subnet have same subnet address.
**IP addressing: CIDR**

**CIDR: Classless InterDomain Routing**
- subnet portion of address of arbitrary length
- address format: `a.b.c.d/x`, where `x` is # bits in subnet portion of address

```
11001000 00010111 00010000 00000000
```

200.23.16.0/23

The `x` bits are also called **network prefix**.
IP addresses: how to get one?

Q: How does a host get IP address?

- hard-coded by system admin in a file ("static IP address")
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
  - Static IP address

- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"
DHCP: Dynamic Host Configuration Protocol

**Goal:** allow host to *dynamically* obtain its IP address from network when joining

- Can renew its *lease* on address in use
- Allows reuse of addresses (only hold address while connected)
- Support for mobile users who want to join network (more shortly)

**DHCP overview:**

- host broadcasts “*DHCP discover*” msg
- host requests IP address: “*DHCP request*” msg
- DHCP server sends address: “*DHCP ack*” msg
DHCP client-server scenario

Arriving DHCP client needs address in this network.
DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover
src: 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
Lifetime: 3600 secs

DHCP request
src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP ACK
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs
DHCP: more than IP address

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

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)
DHCP: example

- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
  - DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
  - Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
  - Ethernet demuxed to IP demuxed, UDP demuxed to DHCP
DHCP: example

- DCP server formulates DHCP ACK containing client’s IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame 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
**IP addresses: how to get one?**

**Q:** How does **network** get subnet part of IP addr?

**A:** gets allocated portion of its provider ISP’s address space

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>11001000 00010111 00010000 00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>11001000 00010111 00010010 00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>11001000 00010111 00010100 00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>...</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>Organization 7</td>
<td>11001000 00010111 00011110 00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>
Hierarchical addressing allows efficient advertisement of routing information:

- **Organization 0**: 200.23.16.0/23
- **Organization 1**: 200.23.18.0/23
- **Organization 2**: 200.23.20.0/23
- **Organization 7**: 200.23.30.0/23

**Fly-By-Night-ISP**:
- “Send me anything with addresses beginning 200.23.16.0/20”

**ISPs-R-Us**:
- “Send me anything with addresses beginning 199.31.0.0/16”

Internet
Q: How does an ISP get block of addresses?
A: ICANN: Internet Corporation for Assigned Names and Numbers
   - allocates addresses
   - manages DNS
   - assigns domain names, resolves disputes
NAT: Network Address Translation

All datagrams leaving local network have **same** single source NAT IP address: 138.76.29.7, different source port numbers.

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual).
NAT: Network Address Translation

- Motivation: local network uses just one IP address as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).
NAT: Network Address Translation

Implementation: NAT router must:

○ **outgoing datagrams:** replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  
  ... remote clients/servers will respond using (NAT IP address, new port #) as destination addr.

○ **remember (in NAT translation table)** every (source IP address, port #) to (NAT IP address, new port #) translation pair

○ **incoming datagrams:** replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table
NAT: Network Address Translation

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

3: Reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

NAT translation table

<table>
<thead>
<tr>
<th>WAN side addr</th>
<th>LAN side addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.76.29.7, 5001</td>
<td>10.0.0.1, 3345</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NAT: Network Address Translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!

- NAT is controversial:
  - Routers should only process up to layer 3
  - Violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - Address shortage should instead be solved by IPv6
NAT traversal problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can’t use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7

- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000
NAT traversal problem

- solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration
NAT traversal problem

- solution 3: relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections