Chapter 4
Network Layer: The Data Plane

A note on the use of these Powerpoint slides:
We’re making these slides freely available to all (faculty, students, readers). They’re in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we’d like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2016
© J.F Kurose and K.W. Ross, All Rights Reserved
Chapter 4: outline

4.1 Overview of Network layer
   - data plane
   - control plane

4.2 What’s inside a router

4.3 IP: Internet Protocol
   - datagram format
   - fragmentation
   - IPv4 addressing
   - network address translation
   - IPv6

4.4 Generalized Forward and SDN
   - match
   - action
   - OpenFlow examples of match-plus-action in action


**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 or two interfaces (e.g., wired Ethernet, wireless 802.11)

- **IP addresses associated with each interface**

```plaintext
223.1.1.1 = 11011111 00000001 00000001 00000001
```

Network Layer: Data Plane  4-3
Q: how are interfaces actually connected?
A: we’ll learn about that in chapter 5, 6.

A: wired Ethernet interfaces connected by Ethernet switches

For now: don’t need to worry about how one interface is connected to another (with no intervening router)

A: wireless WiFi interfaces connected by WiFi base station
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 consisting of 3 subnets

Network Layer: Data Plane  4-5
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

Network Layer: Data Plane  4-6
Subnets

how many?
CPDR: 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
IP addresses: how to get one?

Q: How does a host get IP 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 as server
  - “plug-and-play”
DHCP: Dynamic Host Configuration Protocol

goal: allow host to dynamically obtain its IP address from network server when it joins network
  • 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 network (more shortly)

DHCP overview:
  • host broadcasts “DHCP discover” msg [optional]
  • DHCP server responds with “DHCP offer” msg [optional]
  • host requests IP address: “DHCP request” msg
  • DHCP server sends address: “DHCP ack” msg
DHCP client-server scenario

DHCP server

arriving DHCP client needs address in this network

Network Layer: Data Plane 4-11
DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover

Broadcast: is there a DHCP server out there?

DHCP offer

Broadcast: I’m a DHCP server! Here’s an IP address you can use

DHCP request

Broadcast: OK. I’ll take that IP address!

DHCP ACK

Broadcast: OK. You’ve got that IP address!
DHCP: more than IP addresses

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: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

Network Layer: Data Plane 4-14
- 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

DHCP: example
DHCP: Wireshark output (home LAN)

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: **0x6b3a11b7**
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
  Length: 7; Value: 010016D323688A;
  Hardware type: Ethernet
  Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
**Option: (55) Parameter Request List**
  Length: 11; Value: 010F03062C2E2F1F21F92B
  1 = Subnet Mask; 15 = Domain Name
  3 = Router; 6 = Domain Name Server
  44 = NetBIOS over TCP/IP Name Server

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: **0x6b3a11b7**
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP ACK**
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
  Length: 12; Value: 445747E2445749F244574092;
  IP Address: 68.87.71.226;
  IP Address: 68.87.73.242;
  IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."
**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: route aggregation

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”

Network Layer: Data Plane 4-18
Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1

Send me anything with addresses beginning 200.23.16.0/20

Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23
Q: how does an ISP get block of addresses?
A: ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/
  • allocates addresses
  • manages DNS
  • assigns domain names, resolves disputes
NAT: network address translation

- **rest of Internet**
- **local network (e.g., home network)**
- **10.0.0/24**

**138.76.29.7**

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

**10.0.0/24**

*datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)*
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)
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

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>
</tbody>
</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

* Check out the online interactive exercises for more examples: [http://gaia.cs.umass.edu/kurose_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)
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
• address shortage should be solved by IPv6
• violates end-to-end argument
  • NAT possibility must be taken into account by app designers, e.g., P2P applications
• NAT traversal: what if client wants to connect to server behind NAT?
Chapter 4: outline

4.1 Overview of Network layer
   • data plane
   • control plane

4.2 What’s inside a router

4.3 IP: Internet Protocol
   • datagram format
   • fragmentation
   • IPv4 addressing
   • network address translation
   • IPv6

4.4 Generalized Forward and SDN
   • match
   • action
   • OpenFlow examples of match-plus-action in action
IPv6: motivation

- **Initial motivation:** 32-bit address space soon to be completely allocated.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

**IPv6 datagram format:**
- fixed-length 40 byte header
- no fragmentation allowed
IPv6 datagram format

**priority:** identify priority among datagrams in flow

**flow Label:** identify datagrams in same “flow.”
(concept of “flow” not well defined).

**next header:** identify upper layer protocol for data

<table>
<thead>
<tr>
<th>ver</th>
<th>pri</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>payload len</td>
</tr>
<tr>
<td></td>
<td></td>
<td>next hdr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hop limit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>source address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(128 bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>destination address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(128 bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data</td>
</tr>
</tbody>
</table>

32 bits
Other changes from IPv4

- **checksum**: removed entirely to reduce processing time at each hop
- **options**: allowed, but outside of header, indicated by “Next Header” field
- **ICMPv6**: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions
Transition from IPv4 to IPv6

- Not all routers can be upgraded simultaneously
  - No “flag days”
  - How will network operate with mixed IPv4 and IPv6 routers?
- *Tunneling*: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers
Tunneling

**logical view:**
- A (IPv6)
- B (IPv6)
- E (IPv6)
- F (IPv6)

**physical view:**
- A (IPv6)
- B (IPv6)
- C (IPv4)
- D (IPv4)
- E (IPv6)
- F (IPv6)

IPv4 tunnel connecting IPv6 routers
**Tunneling**

**logical view:**

IPv6

IPv6

IPv6

IPv6

**Physical view:**

IPv6

IPv6

IPv4

IPv4

IPv6

IPv6

- Flow: X
  - src: A
  - dest: F
- Data

**A-to-B:** IPv6

**B-to-C:** IPv6 inside IPv4

**E-to-F:** IPv6

**Network Layer: Data Plane 4-32**
IPv6: adoption

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable

Long (long!) time for deployment, use
- 20 years and counting!
- think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
- Why?
Chapter 4: outline

4.1 Overview of Network layer
   • data plane
   • control plane

4.2 What’s inside a router

4.3 IP: Internet Protocol
   • datagram format
   • fragmentation
   • IPv4 addressing
   • network address translation
   • IPv6

4.4 Generalized Forward and SDN
   • match
   • action
   • OpenFlow examples of match-plus-action in action
Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller.
OpenFlow data plane abstraction

- **flow**: defined by header fields
- **generalized forwarding**: simple packet-handling rules
  - **Pattern**: match values in packet header fields
  - **Actions**: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - **Priority**: disambiguate overlapping patterns
  - **Counters**: #bytes and #packets

*Flow table in a router (computed and distributed by controller) define router’s match+action rules*
OpenFlow data plane abstraction

- **flow**: defined by header fields
- **generalized forwarding**: simple packet-handling rules
  - **Pattern**: match values in packet header fields
  - **Actions**: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - **Priority**: disambiguate overlapping patterns
  - **Counters**: #bytes and #packets

1. src=1.2.*.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*.*, dest=3.4.*.*.* → forward(2)
3. src=10.1.2.3, dest=.*.*.*.* → send to controller

* : wildcard
# OpenFlow: Flow Table Entries

**Rule**
- Action
- Stats

**Packet + byte counters**

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>VLAN ID</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
</tr>
</thead>
</table>

- **Link layer**
- **Network layer**
- **Transport layer**
### Destination-based forwarding:

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>port6</td>
</tr>
</tbody>
</table>

**IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6**

### Firewall:

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>drop</td>
</tr>
</tbody>
</table>

**do not forward (block) all datagrams destined to TCP port 22**

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>drop</td>
</tr>
</tbody>
</table>

**do not forward (block) all datagrams sent by host 128.119.1.1**
## Examples

### Destination-based layer 2 (switch) forwarding:

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>22:A7:23:11:E1:02</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port3</td>
</tr>
</tbody>
</table>

*layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6*
OpenFlow abstraction

- **match+action**: unifies different kinds of devices

- **Router**
  - **match**: longest destination IP prefix
  - **action**: forward out a link

- **Switch**
  - **match**: destination MAC address
  - **action**: forward or flood

- **Firewall**
  - **match**: IP addresses and TCP/UDP port numbers
  - **action**: permit or deny

- **NAT**
  - **match**: IP address and port
  - **action**: rewrite address and port
**OpenFlow example**

**Example:** datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2.

<table>
<thead>
<tr>
<th>match</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Src = 10.3.<em>.</em></td>
<td>forward(3)</td>
</tr>
<tr>
<td>IP Dst = 10.2.<em>.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>match</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ingress port = 1</td>
<td>forward(4)</td>
</tr>
<tr>
<td>IP Src = 10.3.<em>.</em></td>
<td></td>
</tr>
<tr>
<td>IP Dst = 10.2.<em>.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>match</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ingress port = 2</td>
<td>forward(3)</td>
</tr>
<tr>
<td>IP Dst = 10.2.0.3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>match</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ingress port = 2</td>
<td>forward(4)</td>
</tr>
<tr>
<td>IP Dst = 10.2.0.4</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 4: done!

4.1 Overview of Network layer: data plane and control plane
4.2 What’s inside a router
4.3 IP: Internet Protocol
   • datagram format
   • fragmentation
   • IPv4 addressing
   • NAT
   • IPv6
4.4 Generalized Forward and SDN
   • match plus action
   • OpenFlow example

**Question:** how do forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

**Answer:** by the control plane (next chapter)