Filling in the gaps we need to make for IP forwarding work in practice

- Getting IP addresses (DHCP)
- Mapping IP to link addresses (ARP)

What’s my IP?

What link layer address do I use?
Getting IP Addresses

• Problem:
  – A node wakes up for the first time …
  – What is its IP address? What’s the IP address of its router? Etc.
  – At least Ethernet address is on NIC
Getting IP Addresses (2)

1. Manual configuration (old days)
   - Can’t be factory set, depends on use

2. A protocol for automatically configuring addresses (DHCP)
   - Shifts burden from users to IT folk
DHCP

• DHCP (Dynamic Host Configuration Protocol), from 1993, widely used

• It leases IP address to nodes

• Provides other parameters too
  – Network prefix
  – Address of local router
  – DNS server, time server, etc.
DHCP Protocol Stack

- DHCP is a client-server application
  - Uses UDP ports 67, 68
DHCP Addressing

• Bootstrap issue:
  – How does node send a message to DHCP server before it is configured?

• Answer:
  – Node sends broadcast messages that delivered to all nodes on the network
  – Broadcast address is all 1s
  – IP (32 bit): 255.255.255.255
  – Ethernet (48 bit): ff:ff:ff:ff:ff:ff
DHCP Messages

Client

One link

Server
DHCP Messages (2)

Client

DISCOVER

OFFER

REQUEST

ACK

Server

Broadcast
DHCP Messages (3)

• To renew an existing lease, an abbreviated sequence is used:
  – REQUEST, followed by ACK

• Protocol also supports replicated servers for reliability
Sending an IP Packet

Problem:
- A node needs Link layer addresses to send a frame over the local link
- How does it get the destination link address from a destination IP address?

Uh oh ...

My IP is 1.2.3.4
**ARP (Address Resolution Protocol)**

- Node uses to map a local IP address to its Link layer addresses

| Link layer | Source | Dest. | Source | Dest. | Payload ...
|------------|--------|-------|--------|-------|-------------
| From NIC   | Ethernet | Ethernet | IP     | IP    |             
| From DHCP  |         |       | From DHCP |       |             
| From ARP   |         |       |         |       |             

From DHCP is used to obtain the MAC address associated with the IP address of the sending node. From ARP is used to resolve the IP address of the destination node to its MAC address. From NIC is the physical connection to the network interface card.
ARP Protocol Stack

- ARP sits right on top of link layer
  - No servers, just asks node with target IP to identify itself
  - Uses broadcast to reach all nodes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ARP</td>
<td>Ethernet</td>
</tr>
</tbody>
</table>

CSE 461 University of Washington
ARP Messages

Node

Target

One link
ARP Messages (2)

Node

REQUEST
Who has IP 1.2.3.4?

Target

Broadcast

REPLY
I do at 1:2:3:4:5:6
Discovery Protocols

• Help nodes find each other
  – There are more of them!
    • E.g., zeroconf, Bonjour

• Often involve broadcast
  – Since nodes aren’t introduced
  – Very handy glue
Other Aspects of Forwarding

- It’s not all about addresses ...

<table>
<thead>
<tr>
<th>Version</th>
<th>IHL</th>
<th>Differentiated Services</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>M</td>
</tr>
<tr>
<td>Identification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time to live</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source address

Destination address

Options (0 or more words)

Payload (e.g., TCP segment)
Other Aspects (2)

- Decrement TTL value
  - Protects against loops
- Checks header checksum
  - To add reliability
- Fragment large packets
  - Split to fit it on next link
- Send congestion signals
  - Warns hosts of congestion
- Generates error messages
  - To help manage network
- Handle various options

Coming later
Topic

• How do we connect networks with different maximum packet sizes?
  – Need to split up packets, or discover the largest size to use
Packet Size Problem

• Different networks have different maximum packet sizes
  – Or MTU (Maximum Transmission Unit)
  – E.g., Ethernet 1.5K, WiFi 2.3K

• Prefer large packets for efficiency
  – But what size is too large?
  – Difficult because node does not know complete network path
Packet Size Solutions

- **Fragmentation (now)**
  - Split up large packets in the network if they are too big to send
  - Classic method, dated

- **Discovery (next)**
  - Find the largest packet that fits on the network path and use it
  - IP uses today instead of fragmentation
IPv4 Fragmentation

- Routers fragment packets that are too large to forward
- Receiving host reassembles to reduce load on routers
IPv4 Fragmentation Fields

- Header fields used to handle packet size differences
  - Identification, Fragment offset, MF/DF control bits
IPv4 Fragmentation Procedure

• Routers split a packet that is too large:
  – Typically break into large pieces
  – Copy IP header to pieces
  – Adjust length on pieces
  – Set offset to indicate position
  – Set MF (More Fragments) on all pieces except last

• Receiving hosts reassembles the pieces:
  – Identification field links pieces together, MF tells receiver when it has all pieces
IPv4 Fragmentation (2)

Before
MTU = 2300
ID = 0x12ef
Data Len = 2300
Offset = 0
MF = 0

After
MTU = 1500
ID = Data Len = Offset = MF =

(Ignore length of headers)
IPv4 Fragmentation (3)

**Before**
MTU = 2300

ID = 0x12ef
Data Len = 2300
Offset = 0
MF = 0

**After**
MTU = 1500

ID = 0x12ef
Data Len = 1500
Offset = 0
MF = 1

ID = 0x12ef
Data Len = 800
Offset = 1500
MF = 0
IPv4 Fragmentation (4)

• It works!
  – Allows repeated fragmentation

• But fragmentation is undesirable
  – More work for routers, hosts
  – Tends to magnify loss rate
  – Security vulnerabilities too
Path MTU Discovery

• Discover the MTU that will fit
  – So we can avoid fragmentation
  – The method in use today

• Host tests path with large packet
  – Routers provide feedback if too large; they tell host what size would have fit
Path MTU Discovery (2)
Path MTU Discovery (3)

- Test #1: MTU = 1400 bytes
- Test #2: MTU = 1200 bytes
- Test #3: MTU = 900 bytes

MTU=1400
MTU=1200 bytes
MTU=900

Try 1200
Try 900
Path MTU Discovery (4)

• Process may seem involved
  – But usually quick to find right size
• Path MTU depends on the path and so can change over time
  – Search is ongoing
• Implemented with ICMP (next)
  – Set DF (Don’t Fragment) bit in IP header to get feedback messages
Topic

• What happens when something goes wrong during forwarding?
  – Need to be able to find the problem
Internet Control Message Protocol

• ICMP is a companion protocol to IP
  – They are implemented together
  – Sits on top of IP (IP Protocol=1)

• Provides error report and testing
  – Error is at router while forwarding
  – Also testing that hosts can use
ICMP Errors

- When router encounters an error while forwarding:
  - It sends an ICMP error report back to the IP source address
  - It discards the problematic packet; host needs to rectify

Oh, now I see ...

Report then toss it!

ICMP report
ICMP Message Format

- Each ICMP message has a Type, Code, and Checksum
- Often carry the start of the offending packet as payload
- Each message is carried in an IP packet
ICMP Message Format (2)

- Each ICMP message has a Type, Code, and Checksum
- Often carry the start of the offending packet as payload
- Each message is carried in an IP packet
# Example ICMP Messages

<table>
<thead>
<tr>
<th>Name</th>
<th>Type / Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dest. Unreachable (Net or Host)</td>
<td>3 / 0 or 1</td>
<td>Lack of connectivity</td>
</tr>
<tr>
<td>Dest. Unreachable (Fragment)</td>
<td>3 / 4</td>
<td>Path MTU Discovery</td>
</tr>
<tr>
<td>Time Exceeded (Transit)</td>
<td>11 / 0</td>
<td>Traceroute</td>
</tr>
<tr>
<td>Echo Request or Reply</td>
<td>8 or 0 / 0</td>
<td>Ping</td>
</tr>
</tbody>
</table>

Testing, not a forwarding error: Host sends Echo Request, and destination responds with an Echo Reply
Traceroute

- IP header contains TTL (Time to live) field
  - Decremented every router hop, with ICMP error if it hits zero
  - Protects against forwarding loops
Traceroute (2)

• Traceroute repurposes TTL and ICMP functionality
  – Sends probe packets increasing TTL starting from 1
  – ICMP errors identify routers on the path
IP version 6, the future of IPv4 that is now (still) being deployed

Why do I want IPv6 again?
Internet Growth

- At least a billion Internet hosts and growing ...
- And we’re using 32-bit addresses!
The End of New IPv4 Addresses

- Now running on leftover blocks held by the regional registries; much tighter allocation policies

Exhausted on 4/11 and 9/12!
IANA (All IPs)

Exhausted on 2/11!

ARIN (US, Canada)
APNIC (Asia Pacific)
RIPE (Europe)
LACNIC (Latin Amer.)
AfriNIC (Africa)

End of the world? 12/21/12?
IP Version 6 to the Rescue

• Effort started by the IETF in 1994
  – Much larger addresses (128 bits)
  – Many sundry improvements

• Became an IETF standard in 1998
  – Nothing much happened for a decade
  – Hampered by deployment issues, and a lack of adoption incentives
  – Big push ~2011 as exhaustion looms
IPv6 Deployment

Percentage of users accessing Google via IPv6

Time for growth!

Source: Google IPv6 Statistics, 30/1/13
IPv6

• Features large addresses
  – 128 bits, most of header

• New notation
  – 8 groups of 4 hex digits (16 bits)
  – Omit leading zeros, groups of zeros

Ex: 2001:0db8:0000:0000:0000:ff00:0042:8329 →
IPv6 (2)

- Lots of other, smaller changes
  - Streamlined header processing
  - Flow label to group of packets
  - Better fit with “advanced” features (mobility, multicasting, security)
IPv6 Transition

• The Big Problem:
  – How to deploy IPv6?
  – Fundamentally incompatible with IPv4

• Dozens of approaches proposed
  – Dual stack (speak IPv4 and IPv6)
  – Translators (convert packets)
  – Tunnels (carry IPv6 over IPv4) »
Tunneling

- Native IPv6 islands connected via IPv4
  - Tunnel carries IPv6 packets across IPv4 network
Tunneling (2)

- Tunnel acts as a single link across IPv4 network
Tunneling (3)

- Tunnel acts as a single link across IPv4 network
  - Difficulty is to set up tunnel endpoints and routing
Topic

• What is NAT (Network Address Translation)? How does it work?
  – NAT is widely used at the edges of the network, e.g., homes

I’m a NAT box too!
Layering Review

• Remember how layering is meant to work?
  – “Routers don’t look beyond the IP header.” Well ...
Middleboxes

• Sit “inside the network” but perform “more than IP” processing on packets to add new functionality
  – NAT box, Firewall / Intrusion Detection System
Middleboxes (2)

• Advantages
  – A possible rapid deployment path when there is no other option
  – Control over many hosts (IT)

• Disadvantages
  – Breaking layering interferes with connectivity; strange side effects
  – Poor vantage point for many tasks
NAT (Network Address Translation) Box

• NAT box connects an internal network to an external network
  – Many internal hosts are connected using few external addresses
  – Middlebox that “translates addresses”

• Motivated by IP address scarcity
  – Controversial at first, now accepted
NAT (2)

- Common scenario:
  - Home computers use “private” IP addresses
  - NAT (in AP/firewall) connects home to ISP using a single external IP address
How NAT Works

• Keeps an internal/external table
  – Typically uses IP address + TCP port
  – This is address and port translation

<table>
<thead>
<tr>
<th>Internal IP:port</th>
<th>External IP : port</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.12 : 5523</td>
<td>44.25.80.3 : 1500</td>
</tr>
<tr>
<td>192.168.1.13 : 1234</td>
<td>44.25.80.3 : 1501</td>
</tr>
<tr>
<td>192.168.2.20 : 1234</td>
<td>44.25.80.3 : 1502</td>
</tr>
</tbody>
</table>

• Need ports to make mapping 1-1 since there are fewer external IPs
How NAT Works (2)

- Internal → External:
  - Look up and rewrite Source IP/port

<table>
<thead>
<tr>
<th>Internal IP:port</th>
<th>External IP : port</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.12 : 5523</td>
<td>44.25.80.3 : 1500</td>
</tr>
</tbody>
</table>

Internal source

<table>
<thead>
<tr>
<th>Src = 192.168.1.12</th>
<th>Dst = 5523</th>
</tr>
</thead>
</table>

NAT box

<table>
<thead>
<tr>
<th>Src = 44.25.80.3</th>
<th>Dst = 1500</th>
</tr>
</thead>
</table>

External destination

IP=X, port=Y
How NAT Works (3)

- External → Internal
  - Look up and rewrite Destination IP/port

<table>
<thead>
<tr>
<th>Internal IP:port</th>
<th>External IP : port</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.12 : 5523</td>
<td>44.25.80.3 : 1500</td>
</tr>
</tbody>
</table>

External source IP=X, port=Y

Internal destination

Src = Dst =

NAT box

Src = Dst =
How NAT Works (4)

• Need to enter translations in the table for it to work
  – Create external name when host makes a TCP connection

<table>
<thead>
<tr>
<th>Internal source</th>
<th>Internal IP:port</th>
<th>External IP:port</th>
<th>External destination IP=X, port=Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src =</td>
<td>192.168.1.12 : 5523</td>
<td></td>
<td>Src = Dst =</td>
</tr>
</tbody>
</table>
NAT Downsides

- Connectivity has been broken!
  - Can only send incoming packets after an outgoing connection is set up
  - Difficult to run servers or peer-to-peer apps (Skype) at home

- Doesn’t work so well when there are no connections (UDP apps)

- Breaks apps that unwisely expose their IP addresses (FTP)
NAT Upsides

• Relieves much IP address pressure
  – Many home hosts behind NATs

• Easy to deploy
  – Rapidly, and by you alone

• Useful functionality
  – Firewall, helps with privacy

• Kinks will get worked out eventually
  – “NAT Traversal” for incoming traffic