Computer Networks

Lecture 1: Network Layer

Adrian Perrig
Network Security Group
ETH Zürich
Welcome to Second Half of the Course!

• During the remainder of the semester, we will look at remaining 3 layers: network, link, and physical layer

• We will also look at advanced topics
  – Transport protocols: QUIC with BBR congestion control
  – Multipath communication
  – Security and privacy: attacks, anonymous communication
  – Next-generation network architecture: SCION
Lecture Style

• Student interaction is encouraged!
  – Please ask questions if something is unclear
  – Please point out any errors that you spot
  – Please focus on lecture instead of Facebook, Twitter, etc.
    • Please turn off your phone and other devices during class
Minds open...

... Laptops closed and cell phones / ipads off
Computers

They were designed to make people more productive, and then someone invented the internet.
Problem with Being Online During Class

• It takes the mind a few minutes to reach a state of deep concentration
  – Reaching deep concentration needs to be trained, many people have lost this ability due to constant / frequent interruptions

• Knowing that one cannot be interrupted increases concentration. Said another way: Expectation of interruption can prevent deep concentration
  – Try to be off-line when studying networking!

• Open laptop / iPad / cell phone used for surfing can also disrupt / interrupt people sitting behind you
Slide Credits

• Slides adapted from slide deck by David Wetherall
• Some slides by Laurent Vanbever (Jennifer Rexford, Scott Shenker)
Where we are in the Course

• **Starting the Network Layer!**
  – Builds on the link layer, provides service to transport layer
  – **Routers** send **packets** over multiple networks

<table>
<thead>
<tr>
<th>Application</th>
<th>Transport</th>
<th>Network</th>
<th>Link</th>
<th>Physical</th>
</tr>
</thead>
</table>
Why do we need a Network layer?

- How do packets reach the destination?
  - E.g., HTTP, DNS messages, how can they reach server / client?
Not like this...
Network Layer Challenges

• Scale to a global Internet
  – Hierarchy, in the form of prefixes

• Heterogeneity
  – IP for internetworking

• Bandwidth Control
  – Lowest-cost routing
  – QoS (Quality of Service)
Topics

- Network service models
  - Datagrams (packets), virtual circuits
- IP (Internet Protocol)
  - Internetworking
  - Forwarding (Longest Matching Prefix)
  - Helpers: ARP and DHCP
  - Fragmentation and MTU discovery
  - Errors: ICMP (traceroute!)
- IPv6, the future of IP
- NAT, a “middlebox”
- Routing algorithms

TODAY!

Next Time
Routing vs. Forwarding

- **Routing** is the process of deciding in which direction to send traffic (*control plane* of network)
  - Network wide (global) and expensive
Routing vs. Forwarding (2)

- **Forwarding** is the process of sending a packet on its way (data plane of network)
  - Node process (local) and fast
Our Plan

• **Forwarding** this time
  – What routers do with packets

• **Routing** next time
  – Logically this comes first
  – But ignore it for now
Network Service Model

- What kind of service does the Network layer provide to the Transport layer?
  - How is it implemented at routers?
Two Network Service Models

• Datagrams, or connectionless service
  – Like postal letters
  – (This one is IP)

• Virtual circuits, or connection-oriented service
  – Like a telephone call
Store-and-Forward Packet Switching (§1.3.1)

• Both models are implemented with store-and-forward packet switching
  – Routers receive a complete packet, storing it temporarily if necessary before forwarding it onwards
  – We use statistical multiplexing to share link bandwidth over time
Store-and-Forward (2)

- Switching element has internal buffering for contention
Store-and-Forward (3)

- Simplified view with per-port output buffering
  - Buffer is typically a FIFO (First In First Out) queue
  - If full, packets are discarded (congestion)
Datagram Model

- Packets contain a destination address; each router uses it to forward each packet, possibly on different paths
Datagram Model (2)

- Each router has a forwarding table keyed by address
  - Gives next hop for each destination address; may change
**IP (Internet Protocol)**

- Network layer of the Internet, uses datagrams
  - IPv4 carries 32 bit src / dst addresses in each packet
Virtual Circuit Model (§1.3.2)

• Three phases:
  1. Connection establishment, circuit is set up
     • Path is chosen, circuit information stored in routers
  2. Data transfer, circuit is used
     • Packets are forwarded along the path
  3. Connection teardown, circuit is deleted
     • Circuit information is removed from routers

• Just like a telephone circuit, but virtual in the sense that no bandwidth need be reserved; statistical sharing of links
Virtual Circuits (2)

- Packets only contain a short label to identify the circuit
  - Labels don’t have any global meaning, only unique for a link
Virtual Circuits (3)

- Each router has a forwarding table keyed by circuit
  - Gives output line and next label to place on packet

[Diagram showing circuits and tables]
Virtual Circuits (4)

- Each router has a forwarding table keyed by circuit
  - Gives output line and next label to place on packet
MPLS: Multi-Protocol Label Switching (§6.5.1)

- A virtual-circuit like technology widely used by ISPs
  - ISP sets up circuits inside their backbone ahead of time
  - ISP adds MPLS label to IP packet at ingress, undoes at egress
Datagrams vs Virtual Circuits

- Complementary strengths

<table>
<thead>
<tr>
<th>Issue</th>
<th>Datagrams</th>
<th>Virtual Circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup phase</td>
<td>Not needed</td>
<td>Required</td>
</tr>
<tr>
<td>Router state</td>
<td>Per destination</td>
<td>Per connection</td>
</tr>
<tr>
<td>Addresses</td>
<td>Packet carries full address</td>
<td>Packet carries short label</td>
</tr>
<tr>
<td>Routing</td>
<td>Per packet</td>
<td>Per circuit</td>
</tr>
<tr>
<td>Failures</td>
<td>Easier to mask</td>
<td>Difficult to mask</td>
</tr>
<tr>
<td>Quality of service</td>
<td>Difficult to add</td>
<td>Easier to add</td>
</tr>
</tbody>
</table>
Internetworking

• How do we connect different networks together?
  – This is called **internetworking**
  – We’ll look at how IP does it
How Networks May Differ

• Basically, in a lot of ways:
  – Service model (datagrams, VCs)
  – Addressing (what kind)
  – QoS (priorities, no priorities)
  – Packet sizes
  – Security (whether encrypted, source authenticated, verified forwarding)

• Internetworking hides the differences with a common protocol (Uh oh.)
Connecting Datagram and VC networks

• An example to show that it’s not so easy
  – Need to map destination address to a VC and vice-versa
  – A bit of a “road bump”, e.g., might have to set up a VC

[Diagram showing network nodes and connections with labels: Source, 802.11, Router, MPLS, Virtual circuit, Destination]
Internetworking – Cerf and Kahn

• Pioneered by Cerf and Kahn, the “fathers of the Internet”
  – In 1974, later led to TCP/IP

• Tackled the problems of interconnecting networks
  – Instead of mandating a single network technology
Internet Reference Model

- IP is the “narrow waist” of the Internet
  - Supports many different links below and apps above

- 2/1. Link
  - Ethernet, DSL, 802.11

- 3. Internet
  - IP

- 4. Transport
  - TCP, UDP

- 7. Application
  - SMTP, HTTP, RTP, DNS
IP as a Lowest Common Denominator

• Suppose only some networks support QoS or security etc.
  – Difficult for internetwork to support

• Pushes IP to be a “lowest common denominator” protocol
  – Asks little of lower-layer networks
  – Gives little as a higher layer service
IPv4 (Internet Protocol) (§4.3.1)

- Various fields to meet straightforward needs
  - Version, Header (IHL) and Total length, Protocol, and Header Checksum

![IPv4 header diagram]

Payload (e.g., TCP segment)
IPv4 (2)

- Network layer of the Internet, uses datagrams
  - Provides a layer of addressing above link addresses (next)
IPv4 (3)

- Some fields to handle packet size differences (later)
  - Identification, Fragment offset, Fragment control bits

![IPv4 Packet Format Diagram](image-url)
IPv4 (4)

- Other fields to meet other needs (explained later)
  - Differentiated Services, Time to live (TTL)

Quality of Service (QoS)
Later, with ICMP

Payload (e.g., TCP segment)
IP Prefixes (§4.3.3)

• What do IP addresses look like?
  – And IP prefixes, or blocks of addresses
  – (This is IPv4; we’ll cover IPv6 later.)

To: 128.0.32.4
From: 18.31.0.67

18.31.0.67
IP Addresses

• IPv4 uses 32-bit addresses
  – Later we’ll see IPv6, which uses 128-bit addresses

• Written in “dotted quad” notation
  – Four 8-bit numbers separated by dots
  
  \[
  \begin{align*}
  &8 \text{ bits} \quad 8 \text{ bits} \quad 8 \text{ bits} \quad 8 \text{ bits} \\
  &\text{aaaaaaaabbbbbbbccccccccddddddddd} \leftrightarrow \text{A.B.C.D} \\
  &\text{00010010001111110000000000000001} \leftrightarrow
  \end{align*}
  \]
IP Prefixes – Modern

- Addresses are allocated in blocks called **prefixes**
  - Addresses in an L-bit prefix have the same top L bits
  - There are $2^{32-L}$ addresses aligned on $2^{32-L}$ boundary
IP Prefixes (2)

- Written in “IP address/length” notation
  - Address is lowest address in the prefix, length is prefix bits
  - E.g., 128.13.0.0/16 is 128.13.0.0 to 128.13.255.255
  - So a /24 ("slash 24") is 256 addresses, and a /32 is one address

000100100001111100000000xxxxxxxxx ↔

| | | | ↔ 128.13.0.0/16 |
IP Prefixes (3)

- **More specific prefix**
  - Has longer prefix, hence a smaller number of IP addresses

- **Less specific prefix**
  - Has shorter prefix, hence a larger number of IP addresses

### Prefix length

<table>
<thead>
<tr>
<th>Prefix length</th>
<th>/0</th>
<th>/8</th>
<th>/16</th>
<th>/24</th>
<th>/32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addresses in prefix</td>
<td>$2^{32}$</td>
<td>$2^{24}$</td>
<td>$2^{16}$</td>
<td>$2^{8}$</td>
<td>$2^0$</td>
</tr>
</tbody>
</table>

Less specific —> More specific
Example Address Allocation

A /24 prefix means that we have 8 bits left for host addresses, enough for 256 hosts?

82.130.102.0 /24

<table>
<thead>
<tr>
<th>prefix part</th>
<th>host part</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>01010010.10000010.01100110.</td>
<td>00000000</td>
<td>82.130.102.0</td>
</tr>
<tr>
<td>01010010.10000010.01100110.</td>
<td>00000001</td>
<td>82.130.102.1</td>
</tr>
<tr>
<td>01010010.10000010.01100110.</td>
<td>00000010</td>
<td>82.130.102.2</td>
</tr>
<tr>
<td>01010010.10000010.01100110.</td>
<td>11111110</td>
<td>82.130.102.254</td>
</tr>
<tr>
<td>01010010.10000010.01100110.</td>
<td>11111111</td>
<td>82.130.102.255</td>
</tr>
</tbody>
</table>
In practice, the first and last IP address of a prefix are not usable, so a /24 has therefore only **254 addresses** that can be allocated to hosts.

<table>
<thead>
<tr>
<th>Prefix part</th>
<th>Host part</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>01010010.10000010.01100110.</td>
<td>00000000</td>
<td>82.130.102.0</td>
</tr>
<tr>
<td>01010010.10000010.01100110.</td>
<td>11111111</td>
<td>82.130.102.255</td>
</tr>
</tbody>
</table>
Prefixes are also sometimes specified using an address and a mask

<table>
<thead>
<tr>
<th>Address</th>
<th>82.130.102.49</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01010010.10000010.01100110.00110001</td>
</tr>
<tr>
<td></td>
<td>11111111.11111111.11111111.00000000</td>
</tr>
<tr>
<td>Mask</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>
ANDing the address and the mask gives you the prefix

<table>
<thead>
<tr>
<th>Address</th>
<th>82.130.102.49</th>
</tr>
</thead>
<tbody>
<tr>
<td>01010010.10000010.01100110.00110001</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mask</th>
<th>255.255.255.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111.11111111.11111111.00000000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prefix</th>
<th>82.130.102.0 better written: 82.130.102/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>01010010.10000010.01100110.00000000</td>
<td></td>
</tr>
</tbody>
</table>
Given this IP prefix 82.130.0.0/17

**Compute**

- # of addressable hosts
- the prefix mask
- network address
- 1st host address
- last host address
- broadcast address
<table>
<thead>
<tr>
<th><strong>Given this IP prefix</strong></th>
<th>82.130.0.0/17</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compute</strong></td>
<td></td>
</tr>
<tr>
<td># of addressable hosts</td>
<td>32,766</td>
</tr>
<tr>
<td>the prefix mask</td>
<td>255.255.128.0</td>
</tr>
<tr>
<td>network address</td>
<td>82.130.0.0</td>
</tr>
<tr>
<td>1st host address</td>
<td>82.130.0.1</td>
</tr>
<tr>
<td>last host address</td>
<td>82.130.127.254</td>
</tr>
<tr>
<td>broadcast address</td>
<td>82.130.127.255</td>
</tr>
</tbody>
</table>
IP Address Classes – Historical

- Originally, IP addresses came in fixed size blocks with the class/size encoded in the high-order bits
  - They still exist, but the classes are now ignored

```
<table>
<thead>
<tr>
<th>Class</th>
<th>Network Portion</th>
<th>Host Portion</th>
<th>Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>0</td>
<td>24</td>
<td>2^{24}</td>
</tr>
<tr>
<td>Class B</td>
<td>10</td>
<td>16</td>
<td>2^{16}</td>
</tr>
<tr>
<td>Class C</td>
<td>110</td>
<td>8</td>
<td>2^{8}</td>
</tr>
</tbody>
</table>
```
Public / Private IP Addresses

• Public IP addresses, e.g., 18.31.0.1
  – Valid destination on the global Internet
  – Must be allocated to you before use
  – Now exhausted ... time for IPv6!

• Private IP addresses (RFC 1918)
  – Can be used freely within private networks (home, small company)
  – 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  – Need public IP address(es) and NAT to connect to global Internet
Allocating Public IP Addresses

- Follows a hierarchical process
  - Internet Assigned Numbers Authority (IANA) delegates to regional internet registries (RIRs)
  - RIRs delegate to companies in their region
  - Companies assign to their customers/computers (later, DHCP)

![Diagram showing the process of IP address allocation](image)
IP Forwarding

• How do routers forward packets?
  – We’ll look at how IP does it
  – (We’ll cover routing later)
Recap

• We want the network layer to:
  – Scale to large networks
    • Using addresses with hierarchy
  – Support diverse technologies
    • Internetworking with IP
  – Use link bandwidth well
    • Lowest-cost routing

This lecture

More later

Next time
IP Forwarding

- IP addresses on one network belong to the same prefix
- Node uses a table that lists the next hop for IP prefixes

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.24.0.0/18</td>
<td>D</td>
</tr>
<tr>
<td>192.24.12.0/22</td>
<td>B</td>
</tr>
</tbody>
</table>

![Diagram of IP forwarding with nodes A, B, C, and D, and network prefixes 192.24.0.0/18 and 192.24.12.0/22]
Longest Matching Prefix (§4.2.1)

• Prefixes in the table might overlap!
  – Combines hierarchy with flexibility

• **Longest matching prefix** forwarding rule:
  – For each packet, find the longest (most specific) prefix that contains the destination address
  – Forward the packet to the next hop router for that prefix
Longest Matching Prefix (2)

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.24.0.0/18</td>
<td>D</td>
</tr>
<tr>
<td>192.24.12.0/22</td>
<td>B</td>
</tr>
</tbody>
</table>

192.24.6.0 →
192.24.14.32 →
192.24.54.0 →
Host/Router Distinction

- In the Internet:
  - Routers do the routing, know which way to all destinations
  - Hosts send remote traffic (out of prefix) to nearest router

Not for my network? Send it to the router

It’s my job to know which way to go ...
Host Forwarding Table

• Give using longest matching prefix
  – 0.0.0.0/0 is a default route that catches all IP addresses

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>My network prefix</td>
<td>Send direct to that IP</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>Send to my router</td>
</tr>
</tbody>
</table>
Flexibility of Longest Matching Prefix

• Can provide default behavior, with less specific prefixes
  – To send traffic going outside an organization to a border router

• Can provide special case behavior, with more specific prefixes
  – For performance, economics, security, ...
Performance of Longest Matching Prefix

• Hierarchical addresses result in a compact table
  – Less specific prefixes reduce table size

• Finding longest match more complex than table lookup
  – Was a concern for fast routers, but not an issue in practice these days thanks to TCAM (next lecture)
  – However, TCAMs consume a lot of power ...
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>Identification</td>
<td>D</td>
<td>M</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>Time to live</td>
<td>Protocol</td>
<td>Header checksum</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
Helping IP with ARP, DHCP (§4.3.3)

- Filling in the gaps we need to make IP forwarding work in practice
  - Getting IP addresses (DHCP)
  - Mapping IP to link addresses (ARP)
Getting IP Addresses

• Problem:
  – A node wakes up for the first time ... 
  – 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 folks

What’s my IP?

Use A.B.C.D
DHCP (§4.3.3)

- DHCP (Dynamic Host Configuration Protocol), from 1993, very widely used
- It leases IP address to nodes
- Provides other parameters too
  - Network prefix
  - Address of local router (aka. "default gateway")
  - DNS server, time server, etc.
DHCP Protocol Stack

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

<table>
<thead>
<tr>
<th>DHCP</th>
<th>UDP</th>
<th>IP</th>
<th>Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DHCP Addressing

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

• Answer:
  – Node sends broadcast messages that are 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

DISCOVER

OFFER

REQUEST

ACK

Server

Broadcast
DHCP Messages (2)

• 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?

![Diagram of a node with IP 1.2.3.4 and a router with the text "Uh oh..."
and "My IP is 1.2.3.4"]
ARP: Address Resolution Protocol (§6.4.1)

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

<table>
<thead>
<tr>
<th>ARP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
</tr>
</tbody>
</table>
ARP Messages
ARP Messages (2)

Node

REQUEST
Who has IP 1.2.3.4?

Broadcast

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

Target
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
Packet Fragmentation (§4.3.2)

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

![Diagram showing a big packet and a router with speech bubbles saying "Take that" and "It’s too big!".](image-url)
Packet Size Problem

• Different networks have different maximum packet sizes or MTUs
  – 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
  – Split up large packets in the network if they are too big to send
  – Classic method (but dated) and slow for routers to fragment

• Discovery
  – Find the largest packet that fits on the network path and use it
  – IP uses this approach today instead of fragmentation
IPv4 Fragmentation (§4.3.2)

- 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

<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></td>
<td>Identification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>Time to live</td>
<td>Protocol</td>
<td>Header checksum</td>
<td></td>
</tr>
<tr>
<td>Source address</td>
<td>Destination address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (0 or more words)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Payload (e.g., TCP segment)
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 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)

• Important property: repeated fragmentation possible!

• But in practice, fragmentation is undesirable
  – More work for routers, hosts
  – Magnifies severity of packet loss (lost fragment prevents reception of entire packet)
  – Security vulnerabilities (Intrusion detection system evasion, DoS on server)
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)

MTU=1400
MTU=1200 bytes
MTU=900
Path MTU Discovery (4)

- Standard exists to convey MTU to host in ICMP message, but not implemented in practice
  - Hosts today use heuristics on which MTUs to try

- Path MTU depends on the path, 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
Error Handling with ICMP (§5.6)

• 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
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 (and for every second packet is kept in router buffer), 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 (§4.3.5)

- IP version 6, the future of IPv4 that is now (still) being deployed

Why do I want IPv6 again?
• At least 1.1 billion Internet hosts and likely to grow further with IoT and mobile devices
• 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 in 9/15 and 4/11 and 9/12!

Exhausted in 2/11!

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
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

→ 2001:db8::ff00:42:8329
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
  - Difficulty is to set up tunnel endpoints and routing
Network Address Translation (§4.3.4)

- 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

• 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, many times even desired
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**
- Src = 192.168.1.12:5523
- Dst = X:Y

**NAT box**

**External destination**
- IP=X, port=Y
- Src = 44.25.80.3:1500
- Dst = X: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

Src = X:Y
Dst = 192.168.1.12:5523

Src = X:Y
Dst = 44.25.80.3:1500
How NAT Works (4)

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

<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>

Src = 192.168.1.12:5523
Dst = X:Y

Src = 44.25.80.3:1500
Dst = X:Y
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 as ASCII within packet data (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