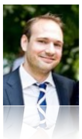


Operating Systems and Networks

Network Lecture 5:

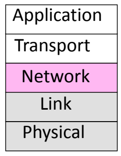
Network Layer (1)

Dr. Raphael Reischuk
Network Security Group
ETH Zürich

May 12, 2016

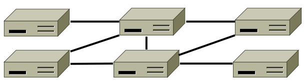
Where we are in the Course?

- Starting the **Network Layer**
  - Builds on the **link layer**
    Routers send **packets** over multiple networks



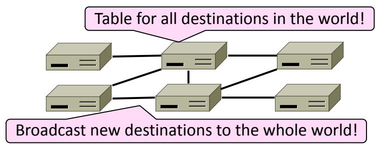
Why do we need a Network layer?

- We can already build networks with links and switches and send frames between hosts …



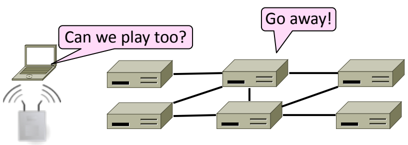
Shortcomings of Switches

1. Don't scale to large networks
   - Blow up of routing table, broadcast



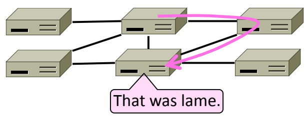
Shortcomings of Switches (2)

2. Don't work across more than one link layer technology
   - Hosts on Ethernet + 3G + 802.11 …



Shortcomings of Switches (3)

3. Don't give much traffic control
   - Want to plan routes / bandwidth



Network Layer Approach

- **Scaling**
  - Hierarchy, in the form of prefixes
- **Heterogeneity**
  - IP for internetworking
- **Bandwidth Control**
  - Lowest-cost routing
  - QoS (Quality of Service)

Topics

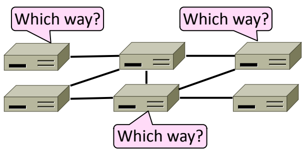
- Network service models
  - Datagrams (packets), virtual circuits
- IP (Internet Protocol)
  - Internetworking
  - Forwarding (Longest Matching Prefix)
  - Helpers: ARP and DHCP
  - Fragmentation and MTU discovery
  - Errors: ICMP (traceroute)
- IPv6, the future of IP
- NAT, a "middlebox"
- Routing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

This time (all items above Routing algorithms)

Next time (Routing algorithms)

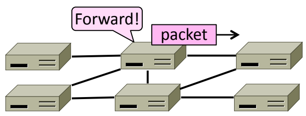
Routing vs. Forwarding

- **Routing** is the process of deciding in which direction to send traffic
  - Network wide (global) and expensive



Routing vs. Forwarding (2)

- **Forwarding** is the process of sending a packet on its way
  - Node process (local) and fast

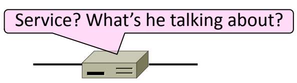


Our Plan

- **Forwarding** (this time)
  - What routers do with packets
- **Routing** (next time)
  - Logically this comes first
  - But ignore it for now

Network Services (§5.1)

- **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)

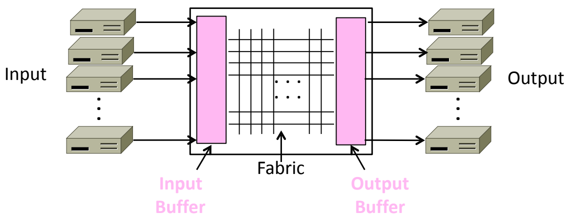
- **Virtual circuits**, or connection-oriented service
  - Like a telephone call

Store-and-Forward Packet Switching

- 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

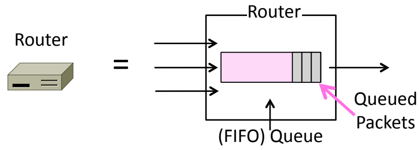
Store-and-Forward (2)

- Switching element has internal buffering for contention



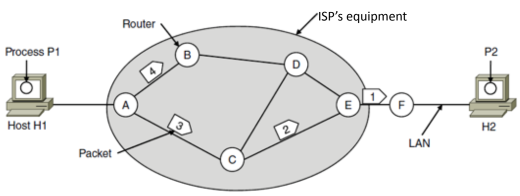
Store-and-Forward (3)

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



Datagram Model

- Packets contain a **destination address**; each router uses it to forward each packet, possibly on different paths



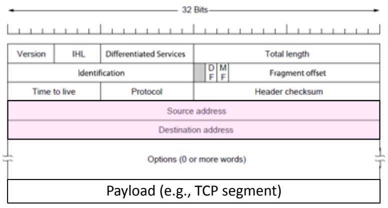
Datagram Model (2)

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

A's table (initially) Dest.	A's table (initially) Line	A's table (later) Dest.	A's table (later) Line	C's Table Dest.	C's Table Line	E's Table Dest.	E's Table Line
A		A		A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C		C	C
D	B	D	B	D	E	D	D
E	C	E	**B**	E	E	E	
F	C	F	**B**	F	E	F	F

IP (Internet Protocol)

- Network layer of the Internet, uses datagrams (next)
  – IPv4 carries 32 bit addresses on each packet (often 1.5 KB)

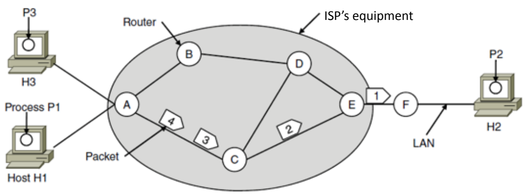


Virtual Circuit Model

- 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

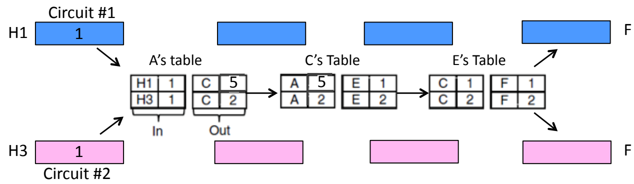
Virtual Circuits (2)

- Packets only contain a **short label** to identify the circuit
  – Labels have **no global meaning**, only unique for a link



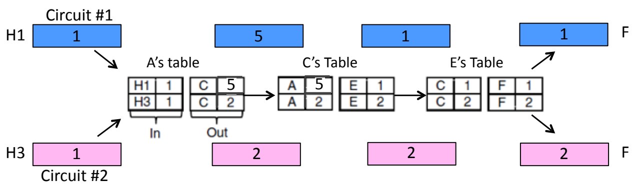
Virtual Circuits (3)

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



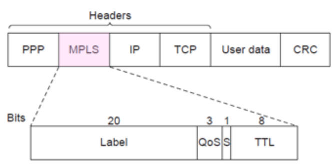
Virtual Circuits (4)

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



MPLS (Multi-Protocol Label Switching, §5.6.5)

- 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



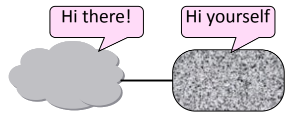
Datagrams vs Virtual Circuits

- Complementary strengths

Issue	Datagrams	Virtual Circuits
Setup phase	Not needed	Required
Router state	Per destination	Per connection
Addresses	Packet carries full address	Packet carries short label
Routing	Per packet	Per circuit
Failures	Easier to mask	Difficult to mask
Quality of service	Difficult to add	Easier to add

Internetworking (§5.5, 5.6.1)

- How do we connect different networks together?
  - This is called internetworking
  - We'll look at how IP does it

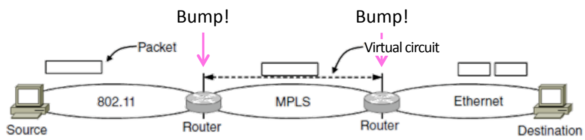


How Networks May Differ

- Basically, in a lot of ways:
  - Service model (datagrams, virtual circuits)
  - Addressing (what kind)
  - QoS (priorities, no priorities)
  - Packet sizes
  - Security (whether encrypted, source authenticated, verified forwarding)
- Internetworking hides the differences with a common protocol

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



Internetworking – Cerf and Kahn

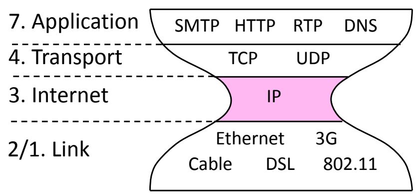
- Pioneered by **Cerf** and **Kahn**, the "fathers of the Internet"
  - In 1974, later led to TCP/IP
- Tackled the problems of interconnecting networks
  - Instead of mandating a single network technology

Vint Cerf Bob Kahn

© 2009 IEEE © 2009 IEEE

Internet Reference Model

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

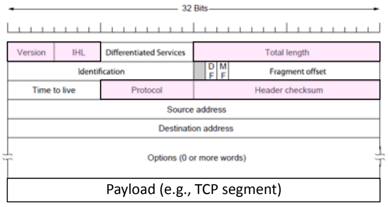


IP as a Lowest Common Denominator

- 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

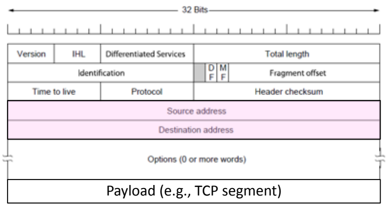
IPv4 (1)

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



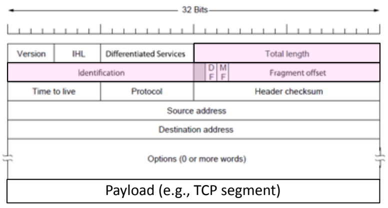
IPv4 (2)

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



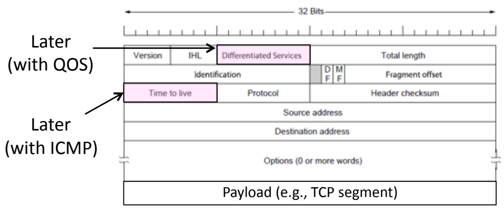
IPv4 (3)

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



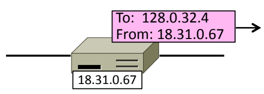
IPv4 (4)

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



IP Prefixes (§5.6.1-5.6.2)

- What do **IP addresses** look like?
  - And IP prefixes (blocks of addresses)
  - (This is IPv4; we'll cover IPv6 later)



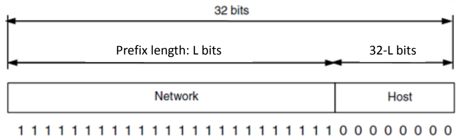
IP Addresses

- 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

```
8 bits   8 bits   8 bits   8 bits
aaaaaaaabbbbbbbbccccccccdddddddd  ↔  A.B.C.D
00010010|00011111|00000000|00000001  ↔
```


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



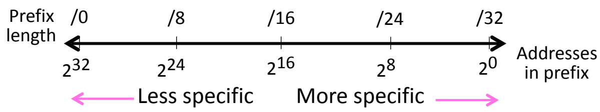
IP Prefixes (2)

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

```
00010010|00011111|00000000|xxxxxxxx  ↔  18.31.0.0/24
10000000|00001101|xxxxxxxx|xxxxxxxx  ↔  128.13.0.0/16
```

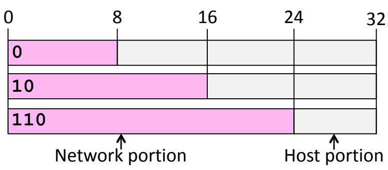

IP Prefixes (3)

- **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



IP Address Classes – Historical

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



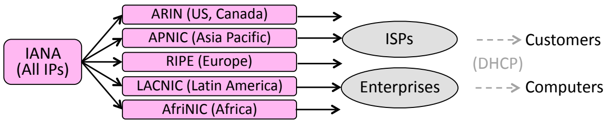
Class A, 2^{24} addresses /8
Class B, 2^{16} addresses /18
Class C, 2^{8} addresses /24

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

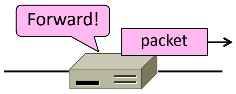
Allocating Public IP Addresses

- 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)



IP Forwarding (§5.6.1-5.6.2)

- How do routers **forward** packets?
  - We'll look at how IP does it
  - (We'll cover routing later)



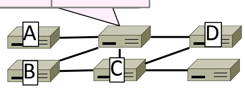
Recap

- We want the network layer to:
  - Scale to large networks — This lecture
    - Using addresses with hierarchy
  - Support diverse technologies — More later
    - Internetworking with IP
  - Use link bandwidth well — Next time
    - Lowest-cost routing

IP Forwarding

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

Prefix	Next Hop
192.24.0.0/18	D
192.24.12.0/22	B



Longest Matching Prefix

- 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)

Prefix	Next Hop
192.24. 0 .0/18	D
192.24.12.0/22	B

192.24.6.0 →
192.24.14.32 →
192.24.54.0 →

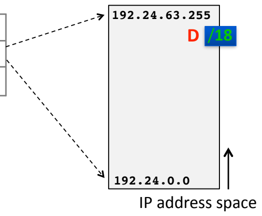
Longest Matching Prefix (2)

Prefix	Next Hop
192.24. 0 .0/18	D
192.24.12.0/22	B

192.24.6.0 →

192.24.14.32 →

192.24.54.0 →



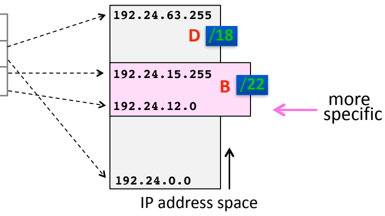
Longest Matching Prefix (2)

Prefix	Next Hop
192.24. 0 .0/18	D
192.24.12.0/22	B

192.24.6.0 →

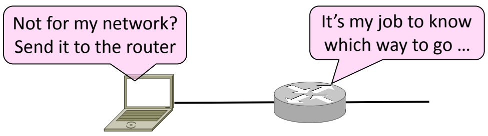
192.24.14.32 →

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



Host Forwarding Table

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

Prefix	Next Hop
My network prefix	Send direct to that IP
0.0.0.0/0	Send to my router

Flexibility of Longest Matching Prefix

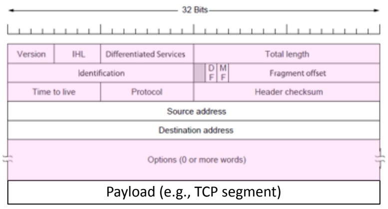
- Can provide **default behavior**, with **less specific** prefixes
  - To send traffic going outside an organization to a border router
- Can provide **special-case behavior**, with **more specific** prefixes
  - For performance, economics, security, …

Performance of Longest Matching Prefix

- Hierarchical addresses result in a compact table
  - Less specific prefixes **reduce table size**
- Finding longest match is more complex than table lookup
  - Was a concern for fast routers, but not an issue in practice these days

Other Aspects of Forwarding

- It's not all about addresses …



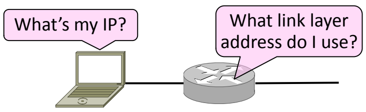
Other Aspects (2)

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

Coming later

Helping IP with ARP, DHCP (§5.6.4)

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

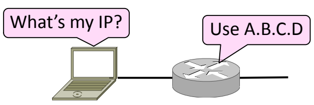


Getting IP Addresses

- Problem:
  - A node wakes up for the first time …
  - What's 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: Dynamic Host Configuration Protocol)
   - Shifts burden from users to IT folks



DHCP

- From 1993, very widely used
- DHCP **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

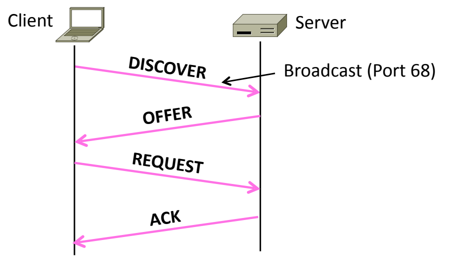
- DHCP is a client-server application
  - Uses UDP ports 67, 68

DHCP
UDP
IP
Ethernet

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

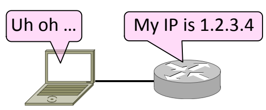


DHCP Messages (2)

- To **renew** an existing lease, an abbreviated sequence is used:
  - REQUEST, followed by ACK
- Protocol also supports replicated servers for reliability

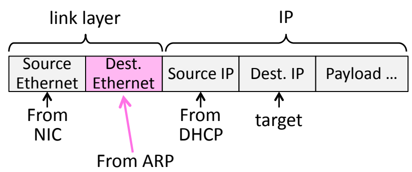
Sending an IP Packet

- 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?



ARP (Address Resolution Protocol)

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

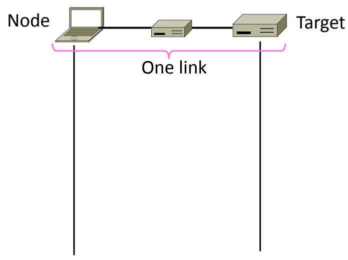


ARP Protocol Stack

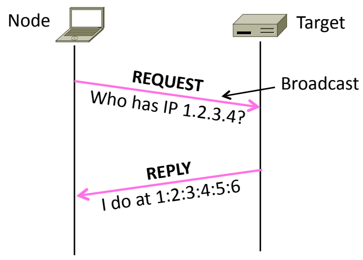
- ARP sits right on top of link layer
  - No servers, just asks node with target IP to identify itself
  - Uses **broadcast** to reach all nodes

ARP
Ethernet

ARP Messages



ARP Messages

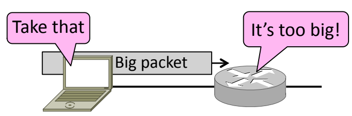


Discovery Protocols

- 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 (§5.5.5)

- 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

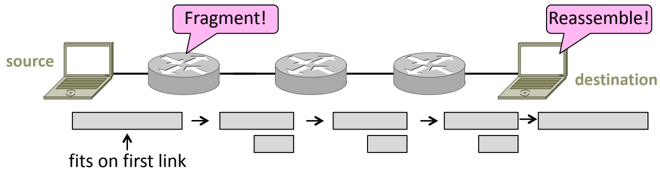
- Different networks have different maximum packet sizes or MTUs
  - MTU = **Maximum Transmission Unit**
  - E.g., Ethernet 1.5K, WiFi 2.3K
- Prefer large packets for efficiency
  - But what size is too large?
  - Difficult … because node does not know the complete network path

Packet Size Solutions

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

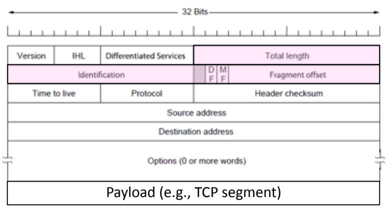
IPv4 Fragmentation

- 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

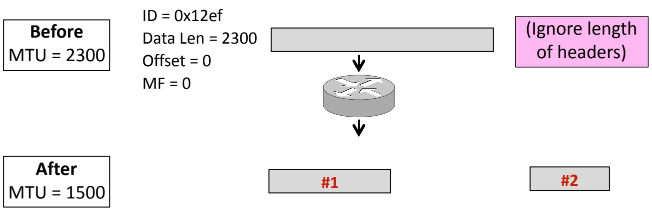


MF = more fragments
DF = don't fragment!

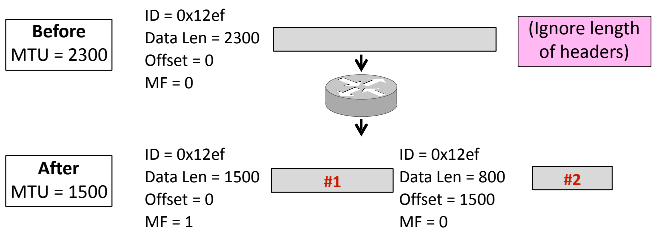
IPv4 Fragmentation Procedure

- 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** on all pieces, except last
- Receiving hosts reassembles pieces:
  - **Identification field** links pieces together, MF tells receiver when it has all pieces

IPv4 Fragmentation



IPv4 Fragmentation



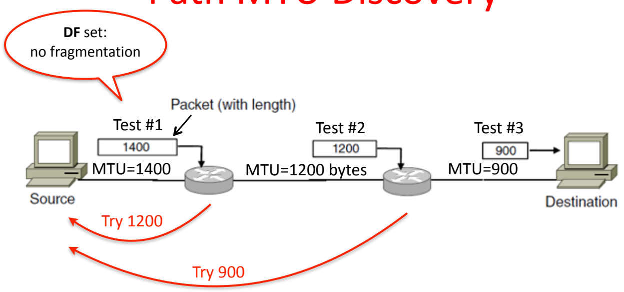
IPv4 Fragmentation

- It works!
  - Does it allow repeated fragmentation?
- But fragmentation is undesirable
  - **More work** for routers and hosts
  - Magnifies severity of packet loss (retransmit entire packet)
  - Security vulnerabilities too (hiding content easy)

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

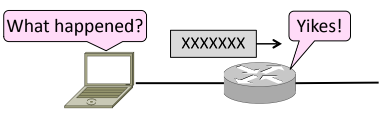


Path MTU Discovery (4)

- Process may seem involved
  - But **usually quick** to find right size
- Path MTU depends on the path, so can change over time
  - Search is ongoing
- Implemented with ICMP (next)
  - Set **DF** bit in IP header to get feedback messages

Error Handling with ICMP (§5.6.4)

- What happens when something goes wrong during forwarding?
  - Need to be able to find the problem

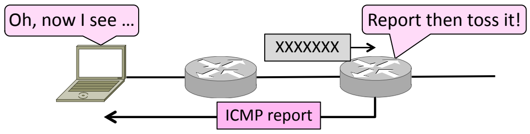


Internet Control Message Protocol

- 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

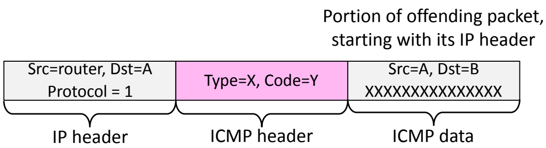


ICMP Message Format

- 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

- 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



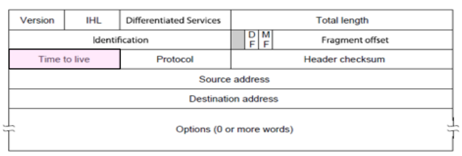
Example ICMP Messages

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

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

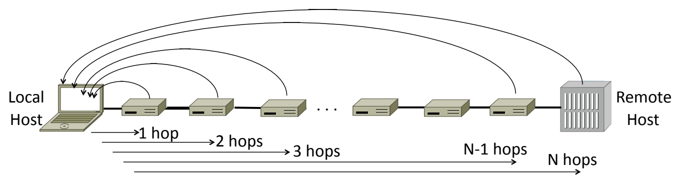
Traceroute

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



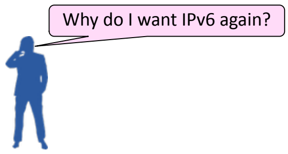
Traceroute (2)

- Traceroute repurposes TTL and ICMP functionality
  - Sends **probe packets** increasing TTL starting from 1
  - ICMP errors **identify routers** on the path



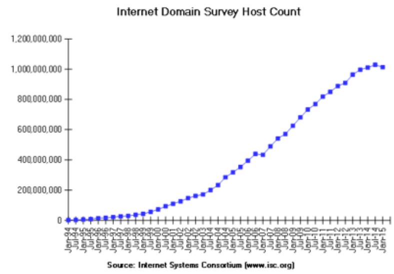
IP Version 6 (§5.6.3)

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



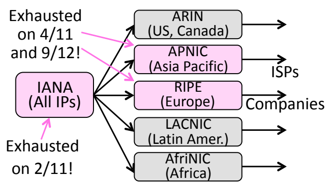
Internet Growth

- At least 1.1 billion Internet hosts and likely to grow further with IoT and mobile devices
- And we're using 32-bit addresses!
  —> 4.3 Billion addresses



The End of New IPv4 Addresses

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



End of the world ? 12/21/12?

IP Version 6 to the Rescue

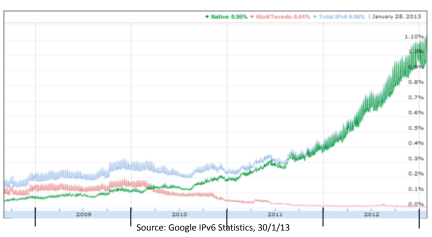
- Effort started by the IETF in 1994
  - Much larger addresses (128 bits)
  - Many sundry improvements

2^96 times larger!
3,4·10^38

- 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

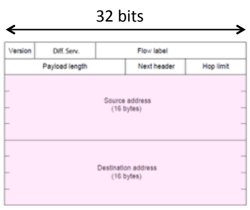


Source: Google IPv6 Statistics, 30/1/13

Time for growth!

IPv6

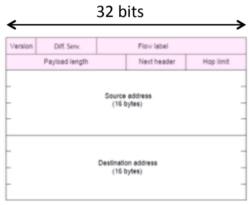
- Features large addresses
  - 128 bits, most of header
- New notation
  - 8 groups of 4 hex digits (16 bits)
  - Omit leading zeros, groups of zeros



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)

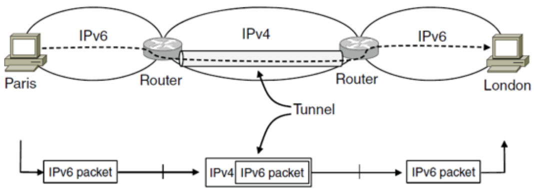


IPv6 Transition

- 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)

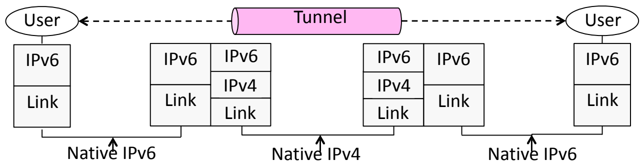
Tunneling

- Native IPv6 islands connected via IPv4
  - Tunnel carries IPv6 packets across IPv4 network



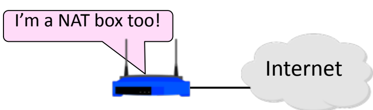
Tunneling (2)

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



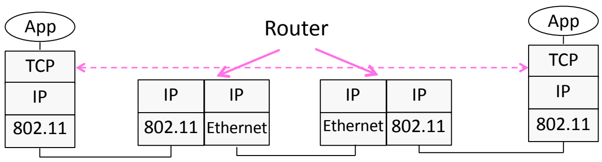
Network Address Translation (§5.6.2)

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



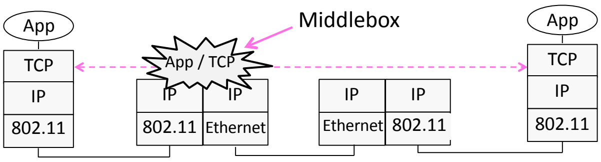
Layering Review

- 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)

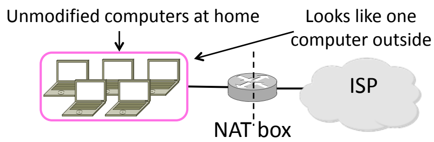
- Advantages
  - A possible rapid deployment path when there is no other option
  - Control over many hosts
- Disadvantages
  - Breaking layering interferes with connectivity; strange side effects
  - Poor vantage point for many tasks

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

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

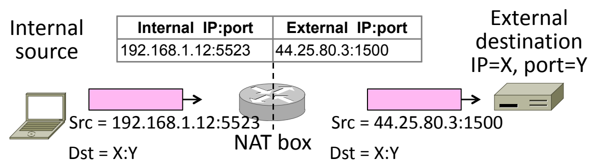
Internal IP:port (What host thinks)	External IP:port (What ISP thinks)
192.168.1.12 : 5523	44.25.80.3 : 1500
192.168.1.13 : 1234	44.25.80.3 : 1501
192.168.2.20 : 1234	44.25.80.3 : 1502

- 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

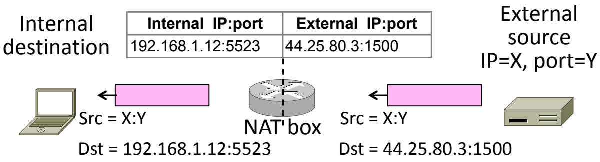
Internal IP:port	External IP:port
192.168.1.12:5523	44.25.80.3:1500



How NAT Works (3)

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

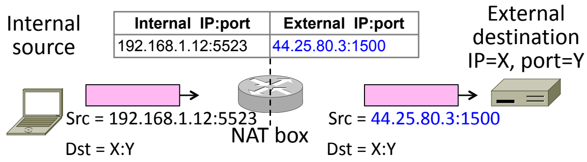
Internal IP:port	External IP:port
192.168.1.12:5523	44.25.80.3:1500



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

Internal IP:port	External IP:port
192.168.1.12:5523	44.25.80.3:1500



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