Routing
- How do routers process IP packets
- Forwarding lookup algorithms
- Assigned reading
  - [D+97] Small Forwarding Tables for Fast Routing Lookups
  - [BV01] Scalable Packet Classification

Outline
- IP address allocation
- Alternative methods for packet forwarding
- IP route lookup
- Variable prefix match algorithms

IP Address Problem (1991)
- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most domains
  - Very few class A – IANA (Internet Assigned Numbers Authority) very careful about giving
  - Class B – greatest problem
    - Sparsely populated – but people refuse to give it back
Solution 1 – CIDR

- Assign multiple class C addresses
- Assign consecutive blocks
- RFC1338 – Classless Inter-Domain Routing (CIDR)

Classless Inter-Domain Routing

- Do not use classes to determine network ID
- Assign any range of addresses to network
  - Use common part of address as network number
    - e.g., addresses 192.4.16 - 196.4.31 have the first 20 bits in common. Thus, we use this as the network number
    - netmask is /20, /xx is valid for almost any xx
- Enables more efficient usage of address space (and router tables)

Solution 2 - NAT

- Network Address Translation (NAT)
- Alternate solution to address space
  - Kludge (but useful)
- Sits between your network and the Internet
- Translates local network layer addresses to global IP addresses
- Has a pool of global IP addresses (less than number of hosts on your network)
NAT Illustration

Operation: Source (S) wants to talk to Destination (D):
- Create S_g-S_p mapping
- Replace S_p with S_g for outgoing packets
- Replace S_g with S_p for incoming packets
- D & S can be just IP addresses or IP addresses + port #'s

Solution 3 - IPv6
- Scale – addresses are 128bit
- Header size?
- Simplification
  - Removes infrequently used parts of header
  - 40byte fixed size vs. 20+ byte variable
- IPv6 removes checksum
  - Relies on upper layer protocols to provide integrity
- IPv6 eliminates fragmentation
  - Requires path MTU discovery
  - Requires 1280 byte MTU
- Flows
  - Help soft state systems
  - Maps well onto TCP connection or stream of UDP packets on host-port pair

IPv6 Header

Summary: Internet Architecture
- Packet-switched datagram network
- IP is the “compatibility layer”
  - Hourglass architecture
  - All hosts and routers run IP
- Stateless architecture
  - no per flow state inside network
Summary: Minimalist Approach

- Dumb network
  - IP provide minimal functionalities to support connectivity
    - Addressing, forwarding, routing
- Smart end system
  - Transport layer or application performs more sophisticated functionalities
    - Flow control, error control, congestion control
- Advantages
  - Accommodate heterogeneous technologies (Ethernet, modem, satellite, wireless)
  - Support diverse applications (telnet, ftp, Web, X windows)
  - Decentralized network administration

Summary: IP Design

- Relatively simple design
  - Some parts not so useful (TOS, options)
- Beginning to show age
  - Unclear what the solution will be → probably IPv6

Outline

- IP address allocation
- Alternative methods for packet forwarding
- IP route lookup
- Variable prefix match algorithms

Techniques for Forwarding Packets

- Source routing
  - Packet carries path
- Table of virtual circuits
  - Connection routed through network to setup state
  - Packets forwarded using connection state
- Table of global addresses (IP)
  - Routers keep next hop for destination
  - Packets carry destination address
Source Routing

- List entire path in packet
  - Driving directions (north 3 hops, east, etc.)
- Router processing
  - Examine first step in directions
  - Strip first step from packet
  - Forward to step just stripped off

Virtual Circuits/Tag Switching

- Connection setup phase
  - Use other means to route setup request
  - Each router allocates flow ID on local link
  - Creates mapping of inbound flow ID/port to outbound flow ID/port
  - Each packet carries connection ID
    - Sent from source with 1st hop connection ID
- Router processing
  - Lookup flow ID – simple table lookup
  - Replace flow ID with outgoing flow ID
  - Forward to output port

Source Routing

- Advantages
  - Switches can be very simple and fast
- Disadvantages
  - Variable (unbounded) header size
  - Sources must know or discover topology (e.g., failures)
- Typical use
  - Ad-hoc networks (DSR)
  - Machine room networks (Myrinet)
Virtual Circuits

- Advantages
  - More efficient lookup (simple table lookup)
  - More flexible (different path for each flow)
  - Can reserve bandwidth at connection setup
  - Easier for hardware implementations
- Disadvantages
  - Still need to route connection setup request
  - More complex failure recovery – must recreate connection state
- Typical uses
  - ATM – combined with fixed sized cells
  - MPLS – tag switching for IP networks

Disadvantages

- Still need to route connection setup request
- More complex failure recovery – must recreate connection state

Typical uses

- ATM – combined with fixed sized cells
- MPLS – tag switching for IP networks

Virtual Circuits

IP Datagrams on Virtual Circuits

- Challenge – when to setup connections
  - At bootup time – permanent virtual circuits (PVC)
    - Large number of circuits
  - For every packet transmission
    - Connection setup is expensive
  - For every connection
    - What is a connection?
    - How to route connectionless traffic?

Traffic pattern

- Few long lived flows
- Flow – set of data packets from source to destination
- Large percentage of packet traffic
- Improving forwarding performance by using virtual circuits for these flows
- Other traffic uses normal IP forwarding

Global Addresses (IP)

- Each packet has destination address
- Each switch has forwarding table of destination → next hop
  - At v and x: destination → east
  - At w and y: destination → south
  - At z: destination → north
- Distributed routing algorithm for calculating forwarding tables
Global Address Example

Router Table Size
- One entry for every host on the Internet
  - 100M entries, doubling every year
- One entry for every LAN
  - Every host on LAN shares prefix
  - Still too many, doubling every year
- One entry for every organization
  - Every host in organization shares prefix
  - Requires careful address allocation

Global Addresses
- Advantages
  - Stateless – simple error recovery
- Disadvantages
  - Every switch knows about every destination
    - Potentially large tables
  - All packets to destination take same route

Summary

<table>
<thead>
<tr>
<th>Source Routing</th>
<th>Global Addresses</th>
<th>Virtual Circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header Size</td>
<td>Worst</td>
<td>OK – Large address</td>
</tr>
<tr>
<td>Router Table Size</td>
<td>None</td>
<td>Number of hosts (prefixes)</td>
</tr>
<tr>
<td>Forward Overhead</td>
<td>Best</td>
<td>Prefix matching</td>
</tr>
<tr>
<td>Setup Overhead</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Error Recovery</td>
<td>Tell all hosts</td>
<td>Tell all routers</td>
</tr>
</tbody>
</table>
Outline

- IP address allocation
- Alternative methods for packet forwarding
- IP route lookup
- Variable prefix match algorithms

Original IP Route Lookup

- Address classes
  - A: 0 | 7 bit network | 24 bit host (16M each)
  - B: 10 | 14 bit network | 16 bit host (64K)
  - C: 110 | 21 bit network | 8 bit host (255)
- Address would specify prefix for forwarding table
  - Simple lookup

Original IP Route Lookup – Example

- www.cmu.edu address 128.2.11.43
  - Class B address – class + network is 128.2
  - Lookup 128.2 in forwarding table
  - Prefix – part of address that really matters for routing
- Forwarding table contains
  - List of class+network entries
  - A few fixed prefix lengths (8/16/24)
- Large tables
  - 2 Million class C networks
  - 32 bits does not give enough space encode network location information inside address – i.e., create a structured hierarchy

CIDR Revisited

- Supernets
  - Assign adjacent net addresses to same org
  - Classless routing (CIDR)
- How does this help routing table?
  - Combine routing table entries whenever all nodes with same prefix share same hop
CIDR Example

- Network provider is allocated 8 class C chunks, 201.10.0.0 to 201.10.7.255
  - Allocation uses 3 bits of class C space
  - Remaining 21 bits are network number, written as 201.10.0.0/21
- Replaces 8 class C routing entries with 1 combined entry
  - Routing protocols carry prefix with destination network address
  - Longest prefix match for forwarding

CIDR Illustration

Provider is given 201.10.0.0/21

CIDR Shortcomings

- Multi-homing
- Customer selecting a new provider

Routing to the Network

- Packet to 10.1.1.3 arrives
- Path is R2 – R1 – H1 – H2
### Routing Within the Subnet

- **Packet to 10.1.1.3**
- **Matches 10.1.0.0/23**

Routing table at R2

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>lo0</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>provider</td>
<td>10.1.16.1</td>
</tr>
<tr>
<td>10.1.0.2/24</td>
<td>10.1.8.1</td>
<td>10.1.8.1</td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>10.1.2.2</td>
<td>10.1.2.2</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>10.1.2.2</td>
<td>10.1.2.2</td>
</tr>
</tbody>
</table>

- **Packet to 10.1.1.3**
- **Matches 10.1.1.1/31**

Routing table at H1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>lo0</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>provider</td>
<td>10.1.16.1</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>10.1.1.2</td>
<td>10.1.1.2</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>10.1.1.2</td>
<td>10.1.1.2</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>10.1.1.2</td>
<td>10.1.1.2</td>
</tr>
</tbody>
</table>

- **Packet to 10.1.1.3**
- **Direct route**
  - **Longest prefix match**

Routing table at H1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>lo0</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>provider</td>
<td>10.1.16.1</td>
</tr>
<tr>
<td>10.1.0.2/24</td>
<td>10.1.1.2</td>
<td>10.1.1.2</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>10.1.1.2</td>
<td>10.1.1.2</td>
</tr>
</tbody>
</table>

### Outline

- **IP address allocation**
- **Alternative methods for packet forwarding**
- **IP route lookup**
- **Variable prefix match algorithms**
Compressed Trie Using Sample Database

Compressed Trie

Sample Database
- P1 = 10*
- P2 = 111*
- P3 = 11001*
- P4 = 1*
- P5 = 0*
- P6 = 1000*
- P7 = 100000*
- P8 = 1000000*

Skip Count vs. Path Compression

- Removing one way branches ensures # of trie nodes is at most twice # of prefixes
- Using a skip count requires exact match at end and backtracking on failure → path compression simpler

How To Do Variable Prefix Match
- Traditional method – Patricia Tree
  - Arrange route entries into a series of bit tests
- Worst case = 32 bit tests
  - Problem: memory speed is a bottleneck

Speeding up Prefix Match (P+98)
- Cut prefix tree at 16 bit depth
  - 64K bit mask
  - Bit = 1 if tree continues below cut (root head)
  - Bit = 1 if leaf at depth 16 or less (genuine head)
  - Bit = 0 if part of range covered by leaf
Prefix Tree

Prefix Tree

Speeding up Prefix Match (P+98)

- Each 1 corresponds to either a route or a subtree
  - Keep array of routes/pointers to subtree
  - Need index into array – how to count # of 1s
  - Keep running count to 16bit word in base index + code word (6 bits)
  - Need to count 1s in last 16bit word
    - Clever tricks
  - Subtrees are handled separately

Speeding up Prefix Match (P+98)

- Scaling issues
  - How would it handle IPv6
- Update issues
- Other possibilities
  - Why were the cuts done at 16/24/32 bits?
  - Improve data structure by shuffling bits
### Speeding up Prefix Match - Alternatives

- **Route caches**
  - Temporal locality
  - Many packets to same destination
- **Other algorithms**
  - Waldvogel – Sigcomm 97
    - Binary search on prefixes
    - Works well for larger addresses
  - Bremler-Barr – Sigcomm 99
    - Clue = prefix length matched at previous hop
    - Why is this useful?

### Binary Search on Ranges

- Problem: Slow search \( \log_2 N + 1 = 20 \) for a million prefixes and update \( O(n) \).

- Encode each prefix as range and place all range endpoints in binary search table or tree. Need two next hops per entry for > and = case. [Lampson, Srinivasan, Varghese]

### Speeding up Prefix Match - Alternatives

- **Content addressable memory (CAM)**
  - Hardware based route lookup
  - Input = tag, output = value associated with tag
  - Requires exact match with tag
    - Multiple cycles (1 per prefix searched) with single CAM
    - Multiple CAMs (1 per prefix) searched in parallel
  - Ternary CAM
    - 0,1,don’t care values in tag match
    - Priority (i.e. longest prefix) by order of entries in CAM

### Next Lecture: Routers & Routing

- **High-speed router architecture**
- **Intro to routing protocols**
- **Assigned reading**
  - [McK97] A Fast Switched Backplane for a Gigabit Switched Router
  - RIP/OSPF