Network Layer 2: Internet Protocol

Telematics group
University of Göttingen, Germany

Overview
- IP addressing
- IP forwarding
- Looking into IP datagrams
  - IP header
  - IP fragmentation and reassembly
- ICMP
- DNS, ARP, DHCP
- NAT
- IPv6
- Multicast


The Internet Network layer

Host, router network layer functions:

- Transport layer: TCP, UDP
- Routing protocol: path selection (RIP, OSPF, BGP)
- IP protocol: addressing conventions, datagram format, packet handling conventions
- ICMP protocol: error reporting, router "signaling"

Link layer

Physical layer

Router Architecture Overview

Two key router functions:
- run routing algorithms/protocol (RIP, OSPF, BGP)
- switching datagrams from incoming to outgoing link
Input Port Functions

Decentralized switching:
- given datagram dest., lookup output port using routing table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Queuing

- Fabric slower than input ports combined → queueing may occur at input queues
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
- Queuing delay and loss due to input buffer overflow!

Three types of switching fabrics

Switching Via Memory

First generation routers:
- Packet copied by system’s (single) CPU
- Speed limited by memory bandwidth (2 bus crossings per datagram)

Modern routers:
- Input port processor performs lookup, copy into memory
- Cisco Catalyst 8500
Switching Via a Bus

- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

Switching Via An Interconnection Network

- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network

Output Ports

- Buffering required when datagrams arrive from fabric faster than the transmission rate
- Scheduling discipline chooses among queued datagrams for transmission

Output port queueing

- buffering when arrival rate via switch exceeds output line speed
- queueing (delay) and loss due to output port buffer overflow!
IP Addressing: Introduction

- IP address: 32-bit identifier for host, router interface
- Interface: connection between host/router and physical link
  - Router's typically have multiple interfaces
  - Host may have multiple interfaces
  - IP addresses associated with each interface

- 3.1.1 = 11011111 00000001 00000001 00000001

IP Addressing

- IP address:
  - Network part (high order bits)
  - Host part (low order bits)
- What's a network?
  - (from IP address perspective)
  - Device interfaces with same network part of IP address
  - Can physically reach each other without intervening router

Network consisting of 3 IP networks
(for IP addresses starting with 223, first 24 bits are network addresses)

IP Addresses

given notion of "network", let's re-examine IP addresses:
"class-full" addressing:

<table>
<thead>
<tr>
<th>Class</th>
<th>Network</th>
<th>Host</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>128.0.0.0 to 127.255.255</td>
<td>0.0.0.0 to 255.255.255</td>
<td>128.0.0.0 to 127.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>192.0.0.0 to 191.255.255.255</td>
<td>0.0.0.0 to 255.255.255.255</td>
<td>192.0.0.0 to 191.255.255.255</td>
</tr>
<tr>
<td>C</td>
<td>224.0.0.0 to 239.255.255.255</td>
<td>0.0.0.0 to 255.255.255.255</td>
<td>224.0.0.0 to 239.255.255.255</td>
</tr>
</tbody>
</table>

32 bits
IP addressing: CIDR

- Classful addressing:
  - Inefficient use of address space, address space exhaustion
  - E.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network
- CIDR: Classless InterDomain Routing
  - Network portion of address of arbitrary length
  - Address format: a.b.c.d/x, where x is # bits in network portion of address

\[
\begin{array}{cccc}
11001000 & 00010111 & 00010000 & 00000000 \\
11001000 & 00010111 & 00010000 & 00000000 \\
11001000 & 00010111 & 00010000 & 00000000 \\
11001000 & 00010111 & 00010000 & 00000000 \\
11001000 & 00010111 & 00010000 & 00000000 \\
\end{array}
\]

200.23.16.0/23

IP addresses: how to get one?

Q: How does host get IP address?

A: Hard-coded by system admin in a file

- Wintel: control-panel->network->configuration->tcp/ip->properties
- UNIX: /etc/rc.config

DHCP: Dynamic Host Configuration Protocol:
  - Dynamically get address from server
  - "Plug-and-play" (more shortly)

Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:

ISP's block

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>Organization 0</th>
<th>Organization 1</th>
<th>Organization 2</th>
<th>Organization 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000</td>
<td>11001000</td>
<td>11001000</td>
<td>11001000</td>
<td></td>
</tr>
<tr>
<td>00010111</td>
<td>00010111</td>
<td>00010111</td>
<td>00010111</td>
<td></td>
</tr>
<tr>
<td>00010000</td>
<td>00010000</td>
<td>00010000</td>
<td>00010000</td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td>200.23.16.0/23</td>
<td>200.23.18.0/23</td>
<td>200.23.20.0/23</td>
<td>200.23.30.0/23</td>
<td></td>
</tr>
</tbody>
</table>

Hierarchical addressing: route aggregation

- "Send me anything with addresses beginning 199.31.0.0/16"
- "Send me anything with addresses beginning 200.23.0.0/23"
- "Send me anything with addresses beginning 199.31.0.0/16"
Hierarchical addressing: more specific routes

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

"Send me anything with addresses beginning 200.23.16.0/20"

200.23.16.0/23

200.23.18.0/23

200.23.30.0/23

IPA-Us

Organization 1

Organization 3

Organization 7

Fly-By-Night-ISP

Organization 0

"Send me anything with addresses beginning 200.23.18.0/23"

Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23

Organizations:
- ISP
- Internet
- Organization
- ISP
- Organization

IP addressing: the last word...

Q: How does an ISP get block of addresses?
A: ICANN: Internet Corporation for Assigned Names and Numbers
- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

Getting a datagram from source to dest.

Starting at A, send IP datagram addressed to B:
- look up net. address of B in forwarding table
- find B is on same net. as A
- link layer will send datagram directly to B inside link layer frame
- B and A are directly connected

IP forwarding: getting a datagram from source to dest.

IP datagram:
- datagram remains unchanged, as it travels from source to destination
- addr fields of interest here

Forwarding table in A
- Dest. Net.
- next router
- Nhops
- misc
- fields
- IP addr
- IP addr
- data
- data
- data
- data
- data
- data

Forwarding table in B
- Dest. Net.
- next router
- Nhops
- misc
- fields
- IP addr
- IP addr
- data
- data
- data
- data
- data
- data

IP forwarding fields:
- source
- dest
- next router
- nhops
- data
- misc
Getting a datagram from source to dest.

Starting at A, dest. E:
• look up network address of E in forwarding table
  – A, E not directly attached
  – routing table: next hop router to E is 223.1.1.4
• link layer sends datagram to router 223.1.1.4 inside link-layer frame
• datagram arrives at 223.1.1.4
• continued....

Arriving at 223.1.4, destined for 223.1.2.2
• look up network address of E in router’s forwarding table
• E on same network as router’s interface 223.1.2.9
  – router, E directly attached
• link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
• datagram arrives at 223.1.2.2!!! (hooray!)

IP datagram format

- IP protocol version
- header length (in bytes)
- “type” of data
- max number remaining hops (decremented at each router)
- upper layer protocol to deliver payload to
- how much overhead with TCP?
  - 20 bytes of TCP
  - 20 bytes of IP
  - 40 bytes = app layer overhead

- 32-bit source IP address
- 32-bit destination IP address
- 16-bit identifier
- fragments
- time to live
- internet checksum
- options (if any)

- total datagram length (in bytes)
- for fragmentation/reassembly

- E.g. timestamp, record route taken, specify list of routers to visit.

IP Fragmentation & Reassembly

- network links have MTU (max transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
  - one datagram becomes several datagrams
  - “reassembled” only at final destination
- IP header bits used to identify, order related fragments

misc fields
223.1.1
223.1.2
223.1.3
223.1.4
223.1.2.3
223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.9
223.1.2.1
223.1.3.2
223.1.3.1
223.1.3.27
223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.9
223.1.2.2
223.1.2.1
223.1.3.2
223.1.3.1
223.1.3.27
223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.9
223.1.2.2
223.1.2.1
IP Fragmentation and Reassembly

Example

- A 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams

<table>
<thead>
<tr>
<th>Length</th>
<th>ID</th>
<th>Offset</th>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11500</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>11500</td>
<td>x</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>11500</td>
<td>x</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>11500</td>
<td>x</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>11500</td>
<td>x</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>11500</td>
<td>x</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>11500</td>
<td>x</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
<tr>
<td>11500</td>
<td>x</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
</tbody>
</table>

ICMP: Internet Control Message Protocol

- used by hosts, routers, gateways to communicate
- network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/echo reply (used by ping)
- network-layer “above” IP:
  - ICMP messages carried in IP datagrams
  - Type, code plus first 8 bytes of IP datagram causing error
  - Type: 0 - echo reply (ping)
  - Code: 10 - router discovery
  - Code: 11 - TTL expired
  - Code: 12 - bad IP header

Mapping Computer Names to IP addresses

The Domain Naming System (DNS)

Names are hierarchical and belong to a domain:
- e.g. elaine17.stanford.edu
- Common domain names: .com, .edu, .gov, .org, .net, .uk (or other country-specific domain).
- Top-level names are assigned by the Internet Corporation for Assigned Names and Numbers (ICANN).
- A unique name is assigned to each organization.

DNS Client-Server Model

- DNS maintains a hierarchical, distributed database of names.
- Servers are arranged in a hierarchy.
- Each domain has a “root” server.
- An application needing an IP address is a DNS client.

Example:

- On MediaRaum machines, try “host www.mit.edu” or “nslookup www.mit.edu”
- Question: Why does “host ar-goettingen1.g-win.dfn.de” return multiple IP addresses?
An example of names and addresses

Mapping the path between two hosts

```
fu@s2:~$ host s2
s2.ifi.informatik.uni-goettingen.de has address 134.76.81.12
fu@s2:~$ traceroute www.informatik.tu-clausthal.de
traceroute to www.informatik.tu-clausthal.de (139.174.100.42), 30 hops max, 38 byte packets
1 134.76.81.254 (134.76.81.254) 0.623 ms 0.949 ms 0.914 ms
2 gr-theo1-sued1.gwdg.de (134.76.249.145) 0.470 ms 0.450 ms 0.410 ms
3 gr-gwdg1-theo1.gwdg.de (134.76.249.105) 0.532 ms 0.488 ms 0.488 ms
4 c12012-int.gwdg.de (134.76.249.201) 0.452 ms 0.459 ms 0.442 ms
5 ar-goettingen1.g-win.dfn.de (188.1.46.193) 1.137 ms 0.961 ms 0.675 ms
6 ar-goettingen2.g-win.dfn.de (188.1.89.130) 1.309 ms 1.085 ms 0.779 ms
7 7200vxr.rz.tu-clausthal.de (139.174.251.12) 2.487 ms 3.000 ms 3.554 ms
8 r-vlan-tuc.rz.tu-clausthal.de (139.174.253.254) 4.439 ms 3.876 ms 3.474 ms
9 www.informatik.tu-clausthal.de (139.174.100.42) 3.043 ms 5.446 ms 6.197 ms
```

As of Nov 7th, 2002.

Q:

- How does a host know the MAC addresses of other hosts in its local-area network?
- How does a router determine the MAC address from the IP address of a host?

Need a "translator" for IP addresses to/from MAC addresses

Address translation can be static or dynamic

- Static translation requires a lot of bookkeeping work from sysadmin

Address Resolution Protocol (ARP)

- Dynamic address translation
- If destination IP address is not in local ARP cache, broadcast query for that address, at most one host will reply
- ARP table entries are cached for up to 15mins, unless if refreshed
- ARP uses layer-2 broadcasting: what about ATM LANs?

Address Resolution: ARP

- How does a host know the MAC addresses of other hosts in its local-area network?
- How does a router determine the MAC address from the IP address of a host?

Need a "translator" for IP addresses to/from MAC addresses

Address translation can be static or dynamic

- Static translation requires a lot of bookkeeping work from sysadmin

Address Resolution Protocol (ARP)

- Dynamic address translation
- If destination IP address is not in local ARP cache, broadcast query for that address, at most one host will reply
- ARP table entries are cached for up to 15mins, unless if refreshed
- ARP uses layer-2 broadcasting: what about ATM LANs?

ARP packet header

- Example: Ethernet to IP translation

```
<table>
<thead>
<tr>
<th>Hardware type = 1</th>
<th>ProtocolType = 0x0800</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware type = 1</td>
<td>ProtocolType = 0x0800</td>
</tr>
<tr>
<td>SourceHardwareAddr</td>
<td>TargetHardwareAddr</td>
</tr>
<tr>
<td>SourceProtocolAddr</td>
<td>TargetProtocolAddr</td>
</tr>
</tbody>
</table>
```

Configuration of IP addresses

- How is a host allocated a certain IP address?
- Need to satisfy uniqueness requirement
- Need to provide same network address with all other hosts in same network
- Manual configuration: `ipconfig`
  - Requires sysadmin privileges, and it is error-prone
- Dynamic configuration of IP addresses?
  - Simpler to manage, and also more efficient usage of the IP address space
DHCP: Dynamic Host Configuration Protocol

**Goal:** allow host to dynamically obtain its IP address from network server when it joins network

Can renew its lease on address in use

Support for mobile users who want to join network (more shortly)

**DHCP overview:**
- host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario

DHCP server: 223.1.2.5 arriving client

DHCP discover
src: 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddrr: 0.0.0.0
transaction ID: 654

DHCP offer
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
Lifetime: 3600 secs

DHCP request
src: 0.0.0.0, 68
dest: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP ACK
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

NAT: Network Address Translation

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

Datagrams leaving local network have same single source NAT IP address: 138.76.29.7, different source port numbers
NAT: Network Address Translation

- **Motivation:**
  - Local network uses just one IP address as far as outside world is concerned:
    - No need to be allocated range of addresses from ISP.
    - Just one IP address is used for all devices.
  - Can change addresses of devices in local network without notifying outside world.
  - Can change ISP without changing addresses of devices in local network.
  - Devices inside local net not explicitly addressable, visible by outside world (a security plus).

**Implementation:**

- Outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #).
  - Remote clients/servers will respond using (NAT IP address, new port #) as destination addr.
- Remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair.
- Incoming datagrams: replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table.

10.0.0.1
10.0.0.2
10.0.0.3
S: 10.0.0.1, 3345
D: 128.119.40.186, 80

10.0.0.4
138.76.29.7

1: Host 10.0.0.1 sends datagram to 128.119.40.186, 80
2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table.
3: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345
4: Reply arrives dest address: 138.76.29.7, 5001

- **16-bit port-number field:**
  - 60,000 simultaneous connections with a single LAN-side address!
- **NAT is controversial:**
  - Routers should only process up to layer 3
  - Violates end-to-end argument
  - NAT possibility must be taken into account by app designers, eg, P2P applications
  - Address shortage should instead be solved by IPv6
IPv6

- Initial motivation: 32-bit address space completely allocated by 2008.
- Additional motivation:
  - header format helps speed processing/forwarding
  - new "anycast" address: route to "best" of several replicated servers
- IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed

IPv6 Header (Cont)

- Priority: identify priority among datagrams in flow
- Flow Label: identify datagrams in same "flow." (concept of "flow" not well defined)
- Next header: identify upper layer protocol for data

Other Changes from IPv4

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

Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- Two proposed approaches:
  - **Dual Stack**: some routers with dual stack (v6, v4) can “translate” between formats
  - **Tunneling**: IPv6 carried as payload in IPv4 datagram among IPv4 routers
Dual Stack Approach

Flow: X
Src: A
Dest: F
data

A-to-B:
IPv6

B-to-C:
IPv4

Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast

Network multicast

- Router actively participate in multicast, making copies of packets as needed and forwarding towards multicast receivers

Tunneling

Logical view:

Physical view:

Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast
Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast

Application-layer multicast
- end systems involved in multicast copy and forward unicast datagrams among themselves

Internet Multicast Service Model

multicast group concept: use of indirection
- hosts addresses IP datagram to multicast group
- routers forward multicast datagrams to hosts that have "joined" that multicast group

Multicast groups
- class D Internet addresses reserved for multicast:
  1110 Multicast Group ID
- host group semantics:
  - anyone can "join" (receive) multicast group
  - anyone can send to multicast group
  - no network-layer identification to hosts of members
- needed: infrastructure to deliver mcast-addressed datagrams to all hosts that have joined that multicast group

Joining a mcast group: two-step process
- local: host informs local mcast router of desire to join group: IGMP (Internet Group Management Protocol)
- wide area: local router interacts with other routers to receive mcast datagram flow
  - many protocols (e.g., DVMRP, MOSPF, PIM)
IGMP: Internet Group Management Protocol

- **host**: sends IGMP report when application joins mcast group
  - IP_ADD_MEMBERSHIP socket option
  - host need not explicitly "unjoin" group when leaving
- **router**: sends IGMP query at regular intervals
  - host belonging to a mcast group must reply to query

**Multicast Routing**

- **Goal**: find a tree (or trees) connecting routers having local mcast group members
  - tree: not all paths between routers used
- **Approaches**:
  - **source-based**: different tree from each sender to receivers
  - **shared tree**: same tree used by all group members

**Approaches for building mcast trees**

- **Approaches**:
  - **source-based tree**: one tree per source
    - shortest path trees
    - reverse path forwarding
  - **group-shared tree**: group uses one tree
    - minimal spanning (Steiner)
    - center-based trees
  - details for further study

**Shortest Path Tree**

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra’s algorithm

**Shared tree**

**Source-based trees**

**Legend**

- Source
- Router with attached group member
- Link used for forwarding, i indicates order link added by algorithm
Reverse Path Forwarding

- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

  if (mcast datagram received on incoming link on shortest path back to center)
  then flood datagram onto all outgoing links
  else ignore datagram

Reverse Path Forwarding: example

- result is a source-specific reverse SPT
  - may be a bad choice with asymmetric links

Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - “prune” msgs sent upstream by router with no downstream group members

Shared-Tree: Steiner Tree

- Steiner Tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave
**Center-based trees**

- single delivery tree shared by all
- one router identified as “center” of tree
- to join:
  - edge router sends unicast `join-msg` addressed to center router
  - `join-msg` "processed" by intermediate routers and forwarded towards center
  - `join-msg` either hits existing tree branch for this center, or arrives at center
  - path taken by `join-msg` becomes new branch of tree for this router

**Center-based trees: an example**

Suppose R6 chosen as center:

- R1, R2, R3
- R4
- R5
- R6, R7

**Internet Multicasting Routing: DVMRP**

- **DVMRP**: distance vector multicast routing protocol, RFC1075
- **flood and prune**: reverse path forwarding, source-based tree
  - RPF tree based on DVMRP’s own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

**DVMRP: continued...**

- **soft state**: DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: relearn or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- **odds and ends**
  - commonly implemented in commercial routers
  - Mbone routing done using DVMRP
**Tunneling**

**Q:** How to connect "islands" of multicast routers in a "sea" of unicast routers?

- mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router
- receiving mcast router unencapsulates to get mcast datagram

---

**PIM: Protocol Independent Multicast**

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios:
  - **Dense:**
    - group members densely packed, in "close" proximity.
    - bandwidth more plentiful
  - **Sparse:**
    - # networks with group members small wrt # interconnected networks
    - group members "widely dispersed"
    - bandwidth not plentiful

---

**Consequences of Sparse-Dense Dichotomy:**

**Dense**
- group membership by routers assumed until routers explicitly prune
- data-driven construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing profligate

**Sparse**
- no membership until routers explicitly join
- receiver-driven construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing conservative

---

**PIM- Dense Mode**

- flood-and-prune RPF, similar to DVMRP but
  - underlying unicast protocol provides RPF info for incoming datagram
  - less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
  - has protocol mechanism for router to detect it is a leaf-node router
PIM - Sparse Mode

- center-based approach
- router sends join msg to rendezvous point (RP)
  - intermediate routers update state and forward join
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths

sender(s):
- unicast data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send stop msg if no attached receivers
  - "no one is listening!"