Recap: Basic Routing Protocols

- Distance vector protocols
  - Basic DV protocol
    - take away: use monotonicity as a technique to understand liveness
  - DSDV, EIGRP
    - take away: use global invariants to understand safety
- Link state protocols
  - OSPF, IS-IS

Recap: Routing in the Internet

Gateway routers participate in intradomain to learn internal routes.

Recap: BGP

- The de facto Inter-AS standard protocol
- A Path Vector protocol:
  - Derived from Distance Vector protocol
  - Send the entire path (i.e., a sequence of ASNs)
- Why path vector:
  - Path vector prevents counting-to-infinity problem
  - Path vector allows an AS to define local policies on the selection of AS paths for each destination

Recap: BGP Routing Decision Process
BGP Example 1

AS A (OSPF)

AS B (OSPF in intra routing)

AS C

AS D

i → b: I can reach hosts in D; my path: BCD

a1 → i: I can reach hosts in D; my path: AD

a2 → a1: I can reach hosts in D; path: D

Export to E: i → e: I can reach hosts in D; path: IAD

Selection policy:
- Low local pref
- Shortest AS Path
- Prefer eBGP (hot potato)

b → i2: I can reach hosts in D; my path: BCD

i2 → i: I can reach hosts in D; path: BCD

Choose BCD using i2

Observing BGP Paths

- Using one of the looking glass servers:
  http://www.bgp4.as/looking-glasses

Routing: Example 3

AS A (OSPF)

AS B (OSPF in intra routing)

AS C

AS D

i → b: I can reach hosts in D; my path: BCD

a1 → i: I can reach hosts in D; my path: AD

a2 → a1: I can reach hosts in D; path: D

Export to E: i → e: I can reach hosts in D; path: IBCD

Selection policy:
- Low local pref A
- Shortest AS Path
- Prefer iBGP (cold potato)

b → i2: I can reach hosts in D; my path: BCD

i2 → i: I can reach hosts in D; path: BCD

Choose BCD using i2

Summary of Interdomain Routing

- Key features
  - Hierarchical routing
  - Policy routing

Benefits of Internet Hierarchical Routing

- ASes have flexibility to choose their own intra-AS routing protocols
  - allows autonomy
- Only a small # of routers (gateways) from each AS in the inter-AS level
  - improves scalability
- Inter-AS route represented as a list of ASNs instead of detailed routers
  - improves scalability/privacy
Issue 1: Hierarchical Routing May Pay a Price for Path Quality

Issue 2: BGP Instability

The BAD GADGET example:
- 0 is the destination
- the route selection policy of each AS is to prefer its counter clockwise neighbor

Policy interaction causes routing instability!

Understanding Instability: P-Graph

- Nodes in P-graph are feasible paths
- Edges represent priority (low to high)
  - A directed edge from path \( P_2 \) to \( P_1 \)
    - intuition: to let \( N_1 \) choose \( N_1 \times P_1 \), \( P_1 \) must be chosen and exported to \( N_1 \)
  - A directed edge from a lower ranked path to a higher ranked path
    - intuition: the higher ranked path should be considered first

Partial Order Graph and BGP Convergence

- If the P-graph has no loop, then BGP policy converges.
  - intuition: choose the path node from the partial order graph with no out-going edge to non-fixed path nodes, fix the path node, eliminate all no longer feasible; continue
- Example: suppose we swap the order of 30 and 320

Internet Economy: Two Types of Business Relationship

- Customer provider relationship
  - a provider is an AS that connects the customer to the rest of the Internet
  - customer pays the provider for the transit service
  - e.g., Yale is a customer of AT&T and QWEST
- Peer-to-peer relationship
  - mutually agree to exchange traffic between their respective customers
  - there is no payment between peers

Question

Do we often see path instability in the Internet?

Preview: the current Internet ISP economy implies no loop in P-graph!
Implication of Business Relationship: Route Selection Policies

- Route selection (ranking) policy:
  - The typical route selection policy is to prefer customers over peers/providers to reach a destination, i.e., Customer > Peer/Provider.

Example: Typical Export → No-Valley Routing

- A advertises path to C, but not P2.
- A learns path to C, P1, P2.
- A advertises path to C, but not P1.

Typical Export Policies Imply Patterns of Routes

- Invariant 1 of valid BGP routes (with labels representing business relationship)

  P C P C P C Dest

  Reasoning: only route learned from customer is sent to provider; thus after a PC, it is always PC to the destination.

- Invariant 2 of valid BGP routes (with labels representing business relationship)

  CP CP/PP Dest

  Reasoning: routes learned from peer or provider are sent to only customers; thus all relationship before is CP.
Stability of BGP Routing

- Suppose
  1. there is no loop formed by provider-customer relationship in the Internet
  2. each AS uses typical route selection policy: \( C > E/P \)
  3. each AS uses the typical export policies

- Then BGP policy routing always converges.

Case 1: A Link is PC

Proof by contradiction. Assume a loop in P-graph. Consider a fixed link in the loop.

Case 2: Link is CP/PP

Summary: BGP Policy Routing

- Advantage
  - satisfies real demand

- Issue
  - policy dispute can lead to instability
    - current Internet economy provides a stability framework, but if the framework changes, we may see instability

- Comment:
  - a policy routing framework is a preference (ranking) aggregation framework
  - a fundamental negative result is Arrow’s Impossibility Theorem (http://en.wikipedia.org/wiki/Arrow’s_impossibility_theorem)

Routing: Remaining Issue

IP Addressing Scheme: Requirements

- We need an address to uniquely identify each destination

- Routing scalability needs flexibility in aggregation of destination addresses
  - we should be able to aggregate a set of destinations as a single routing unit

- Preview: the unit of routing in the Internet is a network—the destinations in the routing protocols are networks, not individual IP addresses
**IP Address: An IP Address Identifies an Interface**

- IPv4 address: 32-bit identifier for an interface
- interface:
  - routers typically have multiple interfaces
  - host may have multiple interfaces

```
%sbin/ifconfig -a
```

- 223.1.1.1 223.1.1.2 223.1.1.3 223.1.1.4 223.1.1.5 223.1.1.6 223.1.1.7 223.1.1.8 223.1.1.9

**IP Addressing**

- IP address:
  - network part
  - host part

- What's a network? (from IP address perspective)
  - can be routed together (depend on the routing protocol)

```
223.1.1.1 223.1.1.2 223.1.1.3 223.1.1.4 223.1.1.5 223.1.1.6 223.1.1.7 223.1.1.8 223.1.1.9
```

**Specifying Network Address in IP**

"classful" addressing in the original IP design:

<table>
<thead>
<tr>
<th>Class</th>
<th>Address</th>
<th>Subnets</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100.0.0.0 to 107.255.255.255</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>108.0.0.0 to 109.255.255.255</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>110.0.0.0 to 110.255.255.255</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>111.0.0.0 to 111.255.255.255</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>112.0.0.0 to 114.255.255.255</td>
<td></td>
</tr>
</tbody>
</table>

Problem of classful addressing?

**IP Addressing: CIDR**

- Static classful addressing:
  - inefficient use of address space, address space exhaustion
  - e.g., a class A net allocated enough addresses for 16 million hosts; a class B address may also be too big
  - not flexible for aggregation

- 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

```
200.23.16.0/23
```

**CIDR Address Aggregation**

- Q: how to do IP addr lookup: at S?

**Routing Table Size of BGP**

(number of globally advertised, aggregated networks)

Internet Growth
(http://www.caida.org/research/topology/as_core_network/historical.xml)
Routing Table Prefix Length Distr.

IP Addressing: How to Get One?

Q: How does an ISP get its block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers
   - allocates addresses
   - manages DNS
   - assigns domain names, resolves disputes

Example:
   `%whois -h whois.arin.net 130.132.1.1` to check the organization who owns an address

IP addresses: How to Get One?

Q: How does a host get an IP address?

- Static configured
  - Wintel: control-panel->network->configuration->tcp/ip->properties
  - Unix:
    - `%/sbin/ifconfig eth0 inet 192.168.0.10 netmask 255.255.255.0`

- DHCP: Dynamic Host Configuration Protocol:
  - dynamically get address from a server
  - “plug-and-play”

Outline

- Admin. and recap
- BGP
- IP addressing
  - IP forwarding

What A Router Looks Like

Basic Router Structure

Cisco GSR 12416

Juniper M160

switching fabric

run routing algorithms/protocol (RIP, OSPF, BGP)
Input Port Functions

physical layer:
bit-level reception
data link layer:
e.g., Ethernet
network layer:
lookup output port using forwarding table

For more details: http://www.cisco.com/warp/public/63/arch12000-swfabric.html

IP Datagram Format

- IP protocol version: 32 bits
- header length (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

IP protocol
head type
16 bit identifier
32 bit source IP address
32 bit destination IP address
Internet checksum
time to live
fragment offset
IP protocol version number
header length (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

data (variable length, typically a TCP or UDP segment)

ICMP: Internet Control Message Protocol

- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
  - ICMP message: type, code plus first 8 bytes of IP datagram causing error

ICMP message body

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>host unreachable</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>dest unreachable</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>port unreachable</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>dest unknown</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>source quench ( congests control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>

traceroute is developed by a clever use of ICMP
Data Forwarding: Steps

- If no error, look up packet destination address in forwarding table:
  - if datagram for a host on directly attached network, it is the job of the link layer now
  - otherwise,
    - lookup: find next-hop router using longest-prefix matching, and its outgoing interface
    - if needed, do fragmentation
    - forward packet to outgoing interface (to the next hop neighbor)

Try `netstat –rn` to see the forwarding table.

Forwarding Look up using Patricia Trie

Default: -

Example 1 (same network): A→B

Forwarding table in A

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>Nhops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1/24</td>
<td>223.1.1.4</td>
<td>1</td>
</tr>
<tr>
<td>223.1.2/24</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3/24</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

Routing table: next hop router to dest is 223.1.1.4
Link layer sends datagram to router 223.1.1.4 inside a link-layer frame
The dest. of the link layer frame is 223.1.1.4

Example 2 (Different Networks): A→E

Forwarding table in A

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>Nhops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1/24</td>
<td>223.1.1.4</td>
<td>1</td>
</tr>
<tr>
<td>223.1.2/24</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3/24</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

Routing table: next hop router to dest is 223.1.1.4
Link layer sends datagram to router 223.1.1.4 inside a link-layer frame
The dest. of the link layer frame is 223.1.1.4

Arriving at 223.1.1.4, destined for 223.1.2.2

- look up dest address 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!)
Backup: Switching Fabric

Switching: Low End

Switching Via An Interconnection Network
- Overcome bus bandwidth limitations
- Fragmenting datagram into fixed length cells, switch cells through the fabric.
- Crossbar, Banyan networks, and others
- Cisco 12416: switches 320 Gbps (upgradeable to 1.28 Tbps) with 16 slots (each 10G full-duplex) through the crossbar interconnection network

New Potential Bottleneck: Output Ports
- Due to output port contention and head-of-the-Line (HOL) blocking (i.e., queued datagram at front of queue prevents others in queue from moving forward)

Head-of-Line Blocking Limits Thruput
- Due to output port contention and HOL blocking, the stable throughput is only around $2 \times \sqrt{2} = 0.986$ of line speed!

Buffering required when datagrams arrive from fabric faster than the transmission rate

Queueing (delay) and loss due to output port buffer overflow!

Scheduling and queue/ buffer management choose among queued datagrams for transmission

Output Ports
**Backup: IP Multicast**

**IP Fragmentation & Reassembly**
- Network links have MTU (max transfer size) - largest possible link-level frame.
- Different link types, different MTUs, e.g., Ethernet MTU is 1500 bytes.
- Large IP datagram divided (“fragmented”)
  - One datagram becomes several datagrams.
  - “reassembled” only at final destination.
  - IP header bits used to identify, order related fragments.

**IP Fragmentation and Reassembly**

### Example
- 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams

<table>
<thead>
<tr>
<th>ID</th>
<th>offset</th>
<th>fragflag</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>0</td>
<td>4000</td>
</tr>
<tr>
<td>x</td>
<td>1480</td>
<td>0</td>
<td>1500</td>
</tr>
<tr>
<td>x</td>
<td>2960</td>
<td>0</td>
<td>1040</td>
</tr>
</tbody>
</table>

**IP Multicast: Service Model**
- Multicast group concept: use of indirection.
- A group is identified by a location-independent logical address (class D IP address: prefix 1110).
- Open group model
  - Anyone can send packets to the “logical” group address.
  - Anyone can join a group and receive packets.
- Normal, best-effort delivery semantics of IP.

Needed infrastructure to deliver multicast-addressed datagrams to all hosts that have joined that multicast group.

**Multicast Across LANs**
- **Goal**: find a tree (or trees) connecting routers having local multicast group members.
  - **Source-based**: different tree from sender to each receiver.
  - Distance-vector multicast routing protocol (DVMRP).
  - Protocol-independent multicast-dense mode (PIM-DM).
  - Core-Based Tree (CBT).
  - Protocol-independent multicast-sparse mode (PIM-SM).

**Source Tree: Reverse Path Flooding (RPF)**
- A router x forwards a packet from source (S) iff it arrives via neighbor y, and y is on the shortest path from x back to S.
- A packet is replicated to all but the incoming interface.

```
4 1 5 2 3 6
shared tree
```

```
4 1 5 2 3 6
source-based trees
```
Reverse Path Forwarding: Improvement

- Basic idea: forward a packet from S only on child links for S
- A child link of router x for source S
  - a link that has x as parent on the shortest path from the link to S
  - a child x notifies its parent y (through the routing protocol) that it has selected y as its parent

Reverse Path Forwarding: Pruning

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

Pruning

- Prune (Source, Group) at a leaf router if no members
  - send No-Membership Report (NMR) up tree
- If all children of router R prune (S,G)
  - propagate prune for (S,G) to its parent
- What do you do when a member of a group (re)joins?
  - send a Graft message to upstream parent
- How to deal with failures?
  - prune dropped
  - flow is reinstated
  - downstream routers re-prune
- Note: again a soft-state approach

Implementation of Source Trees in the Internet

- Multicast OSFP (MOSFP)
  - Membership is part of the link state distribution; calculate source specific, pre-pruned trees
- Reverse Path Forwarding
  - Distance Vector Multicast Routing Protocol (DVMRP)
  - Protocol Independent Multicast – Dense Mode (PIM-DM)
    - very similar to DVMRP
  - Difference: PIM uses any unicast routing algorithm to determine the path from a router to the source; DVMRP uses distance vector
  - Question: the state requirement of Reverse Path Forwarding

Building a Shared Tree

- Steiner Tree: minimum cost tree connecting all routers with attached group members
- A Steiner tree is not a spanning tree because you do not need to connect all nodes in the network
- Problem is NP-hard
- Excellent heuristics exists
- Not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

Center (Core) based Shared Tree

- Single delivery tree shared by all
- One router identified as “center” of tree
- Tree construction is receiver-based
  - 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
- A sender unicasts a packet to center
- The packet is distributed on the tree when it hits the tree
**Example: M3 Joins**
- Group members: M1, M2
- Discussion: what is property of the constructed tree?

**Example: M1 Sends Data**
- Group members: M1, M2, M3
- M1 sends data

**Shared Tree Protocols in the Internet**
- Core Based Tree
- Protocol Independent Multicast (PIM)
- Sparse mode
- The catch: how do you know the center?
  - session announcement

**Mbone: Tunneling**
- Q: How to connect “islands” of multicast routers in a “sea” of unicast routers?
- physical topology
- logical topology
- multicast datagram encapsulated inside “normal” (non-multicast-addressed) datagram
- normal IP datagram sent thru “tunnel” via regular IP unicast to receiving multicast router
- receiving multicast router unencapsulates to get multicast datagram

**Backup: NAT**
- A local network uses just one public IP address as far as outside world is concerned
- Each device on the local network is assigned a private IP address
- All datagrams leaving local network have same single source NAT IP address: 138.76.29.7, different source port numbers
- Data with source or destination in this network have 192.168.1/24 address for source, destination (as usual)
Private IP

NAT: Network Address Translation

Implementation: NAT router must:
- 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

Network Address Translation: Advantages
- No need to be allocated range of addresses from ISP: - just one public IP address is used for all devices
- 16-bit port-number field allows 60,000 simultaneous connections with a single LAN-side address!
- can change ISP without changing addresses of devices in local network
- can change addresses of devices in local network without notifying outside world
- Devices inside local net not explicitly addressable, visible by outside world (a security plus)

Network Address Translation: Problems
- If both hosts are behind NAT, they will have difficulty establishing connection
- NAT is controversial:
  - routers should process up to only layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by having more addresses --- IPv6!