What is routing?

- **forwarding** – moving packets between ports
  - Look up destination address in forwarding table
  - Find *out-port* or \( \langle \text{out-port}, \text{MAC addr} \rangle \) pair

- **Routing** is process of populating forwarding table
  - Routers exchange messages about nets they can reach
  - Goal: Find optimal route for every destination
  - …or maybe good route, or just any route (depending on scale)
Routing algorithm properties

• **Static vs. dynamic**
  - Static: routes change slowly over time
  - Dynamic: automatically adjust to quickly changing network conditions

• **Global vs. decentralized**
  - Global: All routers have complete topology
  - Decentralized: Only know neighbors & what they tell you

• **Intra-domain vs. Inter-domain routing**
  - Intra-: All routers under same administrative control
  - Intra-: Scale to ~100 networks (e.g., campus like Stanford)
  - Inter-: Decentralized, scale to Internet
Optimality

- View network as a graph
- Assign cost to each edge
  - Can be based on latency, b/w, utilization, queue length, …
- Problem: Find lowest cost path between two nodes
  - Must be computed in distributed way
Distance Vector

- *Local* routing algorithm
- Each node maintains a set of triples
  - \((\text{Destination}, \text{Cost}, \text{NextHop})\)
- Exchange updates w. directly connected neighbors
  - periodically (on the order of several seconds to minutes)
  - whenever table changes (called triggered update)
- Each update is a list of pairs:
  - \((\text{Destination}, \text{Cost})\)
- Update local table if receive a “better” route
  - smaller cost
  - from newly connected/available neighbor
- Refresh existing routes; delete if they time out
DV Example

B’s routing table:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>A</td>
</tr>
</tbody>
</table>
Adapting to failures

- F detects that link to G has failed
- F sets distance to G to infinity and sends update to A
- A sets distance to G to infinity since it uses F to reach G
- A receives periodic update from C with 2-hop path to G
- A sets distance to G to 3 and sends update to F
- F decides it can reach G in 4 hops via A
Danger: Loops

- link from A to E fails
- A advertises distance of infinity to E
- B and C advertise a distance of 2 to E
- B decides it can reach E in 3 hops; advertises this to A
- A decides it can reach E in 4 hops; advertises this to C
- C decides that it can reach E in 5 hops…
How to avoid loops

• Consider small value (e.g., 16) to be infinity
  - Will quickly decide node is unavailable

• Split horizon
  - When sending updates to node $A$, don’t include destinations you route to through $A$

• Split horizon with poison reverse
  - When sending updates to node $A$, explicitly include very high cost ("poison") for destinations you route to through $A$

• Note: Latter two only help between two nodes
  - Can still get loop with three nodes involved
  - Might need to delay advertising routes after changes, but will affect convergence time
Link State

- **Strategy**
  - Send to all nodes (not just neighbors)
  - Send only information about directly connected links (not entire routing table)

- **Link State Packet (LSP)**
  - ID of the node that created the LSP
  - Cost of link to each directly connected neighbor
  - Sequence number (SEQNO)
  - Time-to-live (TTL) for this packet
Reliable flooding

- Store most recent LSP from each node
- Forward LSP to all nodes but one that sent it
- Generate new LSP periodically
  - Increment SEQNO
- Start SEQNO at 0 when reboot
  - If you hear your own packet w. SEQNO = \( n \), set your next SEQNO to \( n + 1 \)
- Decrement TTL of each stored LSP
  - discard when TTL = 0
Calculating best path

• Dijkstra’s shortest path algorithm

• Let:
  - \( N \) denote set of nodes in the graph
  - \( l(i, j) \) denotes non-negative cost (weight) for edge \((i, j)\)
  - \( s \) denotes yourself (node computing paths)

• Initialize variables
  - \( M \leftarrow \{s\} \) (set of nodes “incorporated” so far)
  - \( C_n \leftarrow l(s, n) \) (cost of the path from \( s \) to \( n \))
  - \( R_n \leftarrow \perp \) (next hop on path to \( n \))
Dijkstra’s algorithm

• While $N \neq M$
  - Let $w \in (N - M)$ be node with lowest $C_w$
  - $M \leftarrow M \cup \{w\}$
  - Foreach $n \in (N - M)$, if $C_w + l(w, n) < C_n$
    then $C_n \leftarrow C_w + l(w, n)$, $R_n \leftarrow w$

• Example: $D (D, 0, ⊥)(C, 2, C)(B, 5, C')(A, 10, C)$
Distance Vector vs. Link State

• # of messages
  - DV: convergence time varies, but $\Omega(d)$ where $d$ is # of neighbors of node
  - LS: $O(n \cdot d)$ for $n$ nodes in system

• Computation
  - DV: Could count all the way to $\infty$ if loop
  - LS: $O(n^2)$

• Robustness – what happens with malfunctioning router?
  - DV: Node can advertise incorrect path cost
  - DV: Costs used by others, errors propagate through net
  - LS: Node can advertise incorrect link cost
Metrics

- **Original ARPANET metric**
  - measures number of packets enqueued on each link
  - took neither latency nor bandwidth into consideration

- **New ARPANET metric**
  - stamp each incoming packet with its arrival time (AT)
  - record departure time (DT)
  - when link-level ACK arrives, compute
    \[ \text{Delay} = (DT - AT) + \text{Transmit} + \text{Latency} \]
  - if timeout, reset DT to departure time for retransmission
  - link cost = average delay over some time period

- **Fine Tuning**
  - compressed dynamic range
  - replaced Delay with link utilization

- **Today: policy often trumps performance** [more later]
Intradomain routing protocols

• RIP (routing information protocol)
  - Fairly simple implementation of DV

• OSPF (open shortest path first)
  - LS-based protocol
  - Adds notion of areas for scalability
  - Area 0 is “backbone” area (includes all boundary routers)
  - Traffic between two areas must always go through area 0
  - Only need to know how to route exactly within area
  - Else, just route to appropriate area
  - (Virtual links can allow distant routers to be in area 0)
Figure 4.40  ♦ Hierarchically structured OSPF AS with four areas
Scaling issues

• Every router must be able to forward based on any destination IP address
  - Given address, it needs to know “next hop” (table)
  - Naïve: Have an entry for each address
  - There would be $10^8$ entries!

• Solution: Entry covers range of addresses
  - Can’t do this if addresses are assigned randomly! (e.g., Ethernet addresses)
  - This is why address aggregation is important
  - Addresses allocation should be based on network structure

• What is structure of the Internet?
The Internet, 1990

- Hierarchical structure w. single backbone
Address allocation, 1990

- **Hierarchical IP addresses**
  - Class A (8-bit prefix), B (16-bit), C (24-bit)

- **Subnetting** adds another level within organizations
  - Subnet masks define variable partition of host part
  - Subnets visible only within site
The Internet, today

- Multiple “backbones”
Address allocation, today

- **Class system makes inefficient use of addresses**
  - class C with 2 hosts \(\frac{2}{255} = 0.78\%\) efficient
  - class B with 256 hosts \(\frac{256}{65535} = 0.39\%\) efficient
  - Causes shortage of IP addresses (esp. class B)
  - Makes address authorities reluctant to give out class Bs

- **Still Too Many Networks**
  - routing tables do not scale
  - route propagation protocols do not scale
Supernetting

• Assign block of contiguous network numbers to nearby networks

• Called CIDR: Classless Inter-Domain Routing

• Represent blocks with a single pair
  \((\text{first network address}, \text{count})\)

• Restrict block sizes to powers of 2
  - Represent length of network in bits w. slash
  - E.g.: 128.96.34.0/25 means netmask has 25 1 bits, followed by 7 0 bits, or \(0xffffffff80 = 255.255.255.128\)
  - E.g.: 128.96.33.0/24 means netmask 255.255.255.0

• All routers must understand CIDR addressing
Route Propagation

- For each destination address, must either:
  1. Have prefix mapped to next hop in forwarding table, or
  2. know “smarter router”—default for unknown prefixes
     - Using longest prefix match, default is prefix 0.0.0.0/0
     - Hosts use local router as default, local routers use site edge routers, edge routers use core routers

- **Core routers know everything—no default**

- **Manage using notion of Autonomous System (AS)**

- **Two-level route propagation hierarchy**
  - interior gateway protocol (each AS selects its own)
  - exterior gateway protocol (Internet-wide standard)
Autonomous systems

• Correspond to an administrative domain
  - Internet is not a single network
  - ASes reflect organization of the Internet
  - E.g., Stanford, large company, etc.

• Goals:
  - ASes want to choose their own local routing algorithm
  - ASes want to set policies about non-local routing

• Each AS assigned unique 16-bit number
Types of traffic & AS

• *Local traffic* – packets with src or dst in local AS

• *Transit traffic* – passes through an AS

• *Stub AS*
  - Connects to only a single other AS

• *Multihomed AS*
  - Connects to multiple ASes
  - Carries no transit traffic

• *Transit AS*
  - Connects to multiple ASes and carries transit traffic
Customers/provider relationship

- Smaller ASes (companies) are stub or multihomed
  - Purchase connectivity from one or more regional ISPs

- Regional ISPs typically purchase connectivity from one or more global (a.k.a tier-1) ISPs

- Each such connection has two roles:
  - Customer: smaller AS paying for connectivity
  - Provider: larger AS being paid for connectivity

- Other possibility: ISP-to-ISP connection
Transit vs. peering relationships

• **Customer-provider relationship called transit**
  - Provider allows customer to route to (nearly) all destinations in its routing tables
  - Nearly always involves payment from customer to provider

• **Two ASes may decide to have a peering relationship**
  - Allow each another to route to some of the destinations in their routing tables
  - Typically these are an ISP’s own customers (to whom they provide transit)
  - Usually no money changes hands, so long as traffic ratio is narrower than, e.g., 4:1
Financial Motives: Peering and Transit

- Peering relationship often between competing ISPs
- Tier 1s peer with one another to reach all prefixes
  - Tier 1 often defined as ISP that doesn’t buy transit
- Incentives to peer:
  - Typically, two ISPs notice their own direct customers originate a lot of traffic for the other
  - Each can avoid paying transit costs to others for this traffic; shunt it directly to one another
  - Often better performance (shorter latency, lower loss rate) as avoid transit via another provider
  - Easier than stealing one another’s customers
Financial Motives (continued)

- **Disincentives to peer:**
  - Economic disincentive: transit lets ISP charge customer; peering typically doesn’t
  - Contracts must be renegotiated often
  - Need to agree on how to handle asymmetric traffic loads between peers

- **Sometimes ISPs play chicken**
  - E.g., Cogent & Level 3 both nearly Tier 1, and peered
  - Cogent probably sent a lot more traffic to Level 3 than vice versa
  - Level 3 cut off peering arrangement, asked Cogent to buy transit
  - Neither ISP’s customers could reach each other
  - But Cogent super low-cost provider, so hurt them less
  - Level 3 backed down bug probably extracted some concessions (details secret, but maybe carried cogent’s traffic less far)
Exterior gateway protocol: BGP-4

- **Goal:** Share connectivity information across ASes
  - Don’t strive for “optimal” routes—too hard
  - Different ASes may have different notions of cost
  - May have policies that dictate suboptimal routes

- **Used by two types of routers:**
  - *edge* routers, connecting organization to world
  - *core* routers, making up backbone

- **Within ASes, use any routing protocol (e.g., OSPF)**
  - But backbones would have to propagate too many prefixes
  - So use OSPF only for local prefixes
  - *internal* BGP (iBGP) variant used within AS to propagate information about prefixes learned through external BGP
• Routers use external BGP (eBGP) across ASes
  - In picture eBGP sessions correspond to external links
  - Note: Does not have to be this way [more later]

• Use iBGP within organization
When to propagate announced routes?

- **If AS A advertises route for destination D to AS B**
  - Means A will forward all traffic from A to D—costs bandwidth
  - Strong motivation for ISP to control which routes it advertises

- **When peering**
  - Only let peer AS send to specific your own customers

- **When selling transit**
  - Propagate prefixes advertised by your paying customers
  - If you hear prefix advertised by multiple ASes, favor advertisements from own customer
    (want to send lots of traffic to your customer so buys fast link)

- **When buying transit**
  - Only propagate transit provider’s routes to paying customers
    (not peer ASes)
What routing algorithm should BGP use?

- **Constraints:**
  - Scaling
  - Autonomy (policy and privacy)

- **Link-state?**
  - Requires sharing of complete network information
  - Information exchanges don’t scale
  - Can’t express policy

- **Distance Vector?**
  - Scales and retains privacy
  - Can’t implement policy
  - Can’t avoid loops if shortest paths not taken
Path Vector Protocol

• Distance vector algorithm with extra information
  - For each route, store the complete path (ASes)
  - No extra computation, just extra storage

• Advantages:
  - Can make policy choices based on set of ASes in path
  - Can easily avoid loops

• In addition, separate speaker & gateway roles
  - speaker talks BGP protocol to other ASes
  - gateways are routers that border other ASes
  - Can have more gateways than speakers
  - Speaker can reach gateways over local network
BGP Example

- **Speaker for AS2 advertises reachability to P and Q**
  - network 128.96, 192.4.153, 192.4.32, and 192.4.3, can be reached directly from AS2

- **Speaker for backbone advertises**
  - networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path (AS1, AS2).

- **Speaker can withdraw previously advertised paths**
Basic BGP Messages

- **Open:**
  - Establishes BGP session (uses TCP port #179)

- **Notification:**
  - Report unusual conditions (message header error, ...)

- **Announce:** Inform neighbor of new routes, as
  
  IP prefix: [Attribute 0] [Attribute1] [ ... ]

- **Withdraw:** Inform neighbor of newly inactive routes

- **Keepalive:**
  - Inform neighbor that connection is still viable
Attributes of BGP routes

- **AS path**
- **Origin**
  - Who originated the announcement?
  - IGP, EGP, or “incomplete” (for static routes)
- **Multi-Exit Discriminator (MED)**
  - Used if ASes $A$ & $B$ connect at multiple points [next slide]
- **Local preference**
  - Used in iBGP to select (or give preference to) a particular exit for a particular prefix
Multi-Exit Discriminators (MEDs)

- **Imagine provider** $P$ **and customer** $C$
  - Both have big networks
  - Peer with each other in both Boston and San Francisco

- **Someone in Boston sends a packet to** $C$’s server
  - $P$ can give the packet to $C$ in Boston (requires the packet to go over $C$’s cross-country link)
  - or $P$ can carry traffic to San Francisco, and give it to $C$ there

- **$C$ would rather get the packet in San Francisco**
  - Save bandwidth on its cross-country link
  - Expresses this preference to $P$ using MED

- **ISPs need not honor MEDs from neighbors**
  - Might honor MEDs from paying customers
  - For all else, use hot-potato routing (get packet off your network as cheaply as possible)
Synthesizing forwarding table from BGP

- Given multiple advertisements for same prefix …Which should be used for forwarding table?
  - Consider different attributes in order of decreasing priority

<table>
<thead>
<tr>
<th>Priority</th>
<th>Rule</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LOCAL PREF</td>
<td>Highest LOCAL PREF (e.g., prefer transit customer routes over peer and provider routes)</td>
</tr>
<tr>
<td>2</td>
<td>AS PATH</td>
<td>Shortest AS PATH length</td>
</tr>
<tr>
<td>3</td>
<td>MED</td>
<td>Lowest MED</td>
</tr>
<tr>
<td>4</td>
<td>eBGP&gt;iBGP</td>
<td>Prefer routes learned over eBGP vs. over iBGP</td>
</tr>
<tr>
<td>5</td>
<td>IGP path</td>
<td>“Nearest” egress router</td>
</tr>
<tr>
<td>6</td>
<td>Router ID</td>
<td>Smallest router IP address</td>
</tr>
</tbody>
</table>