Introduction to Computer Networks

Routing Overview
Routing versus Forwarding

• **Forwarding** is the process of sending a packet on its way

• **Routing** is the process of deciding in which direction to send traffic
Improving on the Spanning Tree

• Spanning tree provides basic connectivity
  – e.g., some path $B \rightarrow C$

• Routing uses all links to find “best” paths
  – e.g., use BC, BE, and CE

![Diagram of network with spanning tree and routing examples.](attachment:image_url)
Perspective on Bandwidth Allocation

- Routing allocates network bandwidth adapting to failures; other mechanisms used at other timescales

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Timescale / Adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load-sensitive routing</td>
<td>Seconds / Traffic hotspots</td>
</tr>
<tr>
<td>Routing</td>
<td>Minutes / Equipment failures</td>
</tr>
<tr>
<td>Traffic Engineering</td>
<td>Hours / Network load</td>
</tr>
<tr>
<td>Provisioning</td>
<td>Months / Network customers</td>
</tr>
</tbody>
</table>
Goals of Routing Algorithms

- We want several properties of any routing scheme:

<table>
<thead>
<tr>
<th>Property</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>Finds paths that work</td>
</tr>
<tr>
<td>Efficient paths</td>
<td>Uses network bandwidth well</td>
</tr>
<tr>
<td>Fair paths</td>
<td>Doesn’t starve any nodes</td>
</tr>
<tr>
<td>Fast convergence</td>
<td>Recovers quickly after changes</td>
</tr>
<tr>
<td>Scalability</td>
<td>Works well as network grows large</td>
</tr>
</tbody>
</table>
Rules of Routing Algorithms

• Decentralized, distributed setting
  – All nodes are alike; no controller
  – Nodes only know what they learn by exchanging messages with neighbors
  – Nodes operate concurrently
  – May be node/link/message failures

Who’s there?
Propose a Routing Algorithms

- Suppose you were to design a routing algorithm, how would you go about it?
Introduction to Computer Networks

Shortest Path Routing
(§5.2.1-5.2.2)
Topic

• Defining “best” paths with link costs
  – These are shortest path routes

Best?
What are “Best” paths anyhow?

- Many possibilities:
  - Latency, avoid circuitous paths
  - Bandwidth, avoid slow links
  - Money, avoid expensive links
  - Hops, to reduce switching

- But only consider topology
  - Ignore workload, e.g., hotspots
Shortest Paths

We’ll approximate “best” by a cost function that captures the factors
  – Often call lowest “shortest”

1. Assign each link a cost (distance)
2. Define best path between each pair of nodes as the path that has the lowest total cost (or is shortest)
3. Pick randomly to any break ties
Shortest Paths (2)

- Find the shortest path $A \rightarrow E$
- All links are bidirectional, with equal costs in each direction
  - Can extend model to unequal costs if needed
Shortest Paths (3)

- ABCE is a shortest path
- \( \text{dist}(ABCE) = 4 + 2 + 1 = 7 \)

- This is less than:
  - \( \text{dist}(ABE) = 8 \)
  - \( \text{dist}(ABFE) = 9 \)
  - \( \text{dist}(AE) = 10 \)
  - \( \text{dist}(ABCDE) = 10 \)
Shortest Paths (4)

• Optimality property:
  – Subpaths of shortest paths are also shortest paths
• ABCE is a shortest path
  \(\rightarrow\) So are ABC, AB, BCE, BC, CE
Sink Trees

- Sink tree for a destination is the union of all shortest paths towards the destination
  - Similarly source tree
Sink Trees (2)

• Implications:
  – Only need to use destination to follow shortest paths
  – Each node only need to send to the next hop

• Forwarding table at a node
  – Lists next hop for each destination
  – Routing table may know more
Dijkstra’s Algorithm

Algorithm:

• Mark all nodes tentative, set distances from source to 0 (zero) for source, and \( \infty \) (infinity) for all other nodes

• While tentative nodes remain:
  – Extract N, the one with lowest distance
  – Add link to N to the shortest path tree
  – Relax the distances of neighbors of N by lowering any better distance estimates
Dijkstra’s Algorithm (2)

• Initialization

We’ll compute shortest paths to/from A
Dijkstra’s Algorithm (3)

• Relax around A
Dijkstra’s Algorithm (4)

- Relax around B

![Graph showing the algorithm in action](image-url)
Dijkstra’s Algorithm (5)

- Relax around C
Dijkstra Comments

• Dynamic programming algorithm; leverages optimality property

• Runtime depends on efficiency of extracting min-cost node

• Gives us complete information on the shortest paths to/from one node
  – But requires complete topology
Introduction to Computer Networks

Distance Vector Routing
(§5.2.4)
Topic

• How to compute shortest paths in a distributed network
  – The Distance Vector (DV) approach
Distance Vector Setting

Each node computes its forwarding table in a distributed setting:

1. Nodes know only the cost to their neighbors; not the topology
2. Nodes can talk only to their neighbors using messages
3. All nodes run the same algorithm concurrently
4. Nodes and links may fail, messages may be lost
Distance Vector Algorithm

Each node maintains a vector of distances to all destinations

1. Initialize vector with 0 (zero) cost to self, $\infty$ (infinity) to other destinations
2. Periodically send vector to neighbors
3. Update vector for each destination by selecting the shortest distance heard, after adding cost of neighbor link
   - Use the best neighbor for forwarding
Distance Vector (2)

- Consider from the point of view of node A
  - Can only talk to nodes B and E

<table>
<thead>
<tr>
<th>To</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
</tr>
<tr>
<td>F</td>
<td>∞</td>
</tr>
<tr>
<td>G</td>
<td>∞</td>
</tr>
<tr>
<td>H</td>
<td>∞</td>
</tr>
</tbody>
</table>

Initial vector
Distance Vector (3)

- First exchange with B, E; learn best 1-hop routes

<table>
<thead>
<tr>
<th>To</th>
<th>B says</th>
<th>E says</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>G</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>H</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>B +4</th>
<th>E +10</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
<td>10</td>
</tr>
<tr>
<td>F</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>G</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>H</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A's Cost</th>
<th>A's Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>∞</td>
<td>--</td>
</tr>
<tr>
<td>∞</td>
<td>--</td>
</tr>
<tr>
<td>∞</td>
<td>10</td>
</tr>
<tr>
<td>∞</td>
<td>--</td>
</tr>
<tr>
<td>∞</td>
<td>--</td>
</tr>
</tbody>
</table>

Learned better route

To B says E says
A 0 --
B 4 B
C ∞ --
D ∞ --
E 10 E
F ∞ --
G ∞ --
H ∞ --
Distance Vector (4)

- Second exchange; learn best 2-hop routes

<table>
<thead>
<tr>
<th>To</th>
<th>B says</th>
<th>E says</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>∞</td>
</tr>
<tr>
<td>H</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A's Cost</th>
<th>A's Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>E</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
</tr>
</tbody>
</table>

![Graph showing network distances and updates]
Distance Vector (4)

- Third exchange; learn best 3-hop routes

<table>
<thead>
<tr>
<th>To</th>
<th>B says</th>
<th>E says</th>
<th>B +4</th>
<th>E +10</th>
<th>A's Cost</th>
<th>A's Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>18</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>13</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>11</td>
<td>6</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>10</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>12</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>16</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>H</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>14</td>
<td>9</td>
<td>B</td>
</tr>
</tbody>
</table>
Distance Vector (5)

- Subsequent exchanges; converged

<table>
<thead>
<tr>
<th>To</th>
<th>B says</th>
<th>E says</th>
<th>B +4</th>
<th>E +10</th>
<th>A's Cost</th>
<th>A's Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>17</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>13</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>11</td>
<td>6</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>10</td>
<td>8</td>
<td>B</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>12</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>16</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>H</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>14</td>
<td>9</td>
<td>B</td>
</tr>
</tbody>
</table>
Distance Vector Dynamics

• Adding routes:
  – News travels one hop per exchange

• Removing routes
  – When a node fails, no more exchanges, other nodes forget

• But partitions (unreachable nodes in divided network) are a problem
  – “Count to infinity” scenario
Dynamics (2)

- Good news travels quickly, bad news slowly (inferred)

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```

Desired convergence

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
```

“Count to infinity” scenario
Dynamics (3)

• Various heuristics to address
  – e.g., “Split horizon, poison reverse” (Don’t send route back to where you learned it from.)

• But none are very effective
  – Link state now favored in practice
  – Except when very resource-limited
Introduction to Computer Networks

Link State Routing (§5.2.5)
Topic

• How to compute shortest paths in a distributed network
  – The Link-State (LS) approach
Link-State Routing

• One of two approaches to routing
  – Trades more computation than distance vector for better dynamics

• Widely used in practice
  – Used in Internet/ARPANET from 1979
  – Modern networks use OSPF and IS-IS
Link-State Algorithm

Proceeds in two phases:

1. Nodes flood topology in the form of link state packets
   - Each node learns full topology

2. Each node computes its own forwarding table
   - By running Dijkstra (or equivalent)
Topology Dissemination

- Each node floods link state packet (LSP) that describes their portion of the topology

Node E’s LSP flooded to A, B, C, D, and F
Route Computation

• Each node has full topology
  – By combining all LSPs

• Each node simply runs Dijkstra
  – Some replicated computation, but finds required routes directly
  – Compile forwarding table from sink/source tree
  – That’s it folks!
Handling Changes

- Nodes adjacent to failed link or node will notice
  - Flood updated LSP with less connectivity

<table>
<thead>
<tr>
<th>Seq. #</th>
<th>B’s LSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Seq. #</th>
<th>F’s LSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>4</td>
</tr>
</tbody>
</table>

Failure!  
G xxxxx
Handling Changes (2)

• Link failure
  – Both nodes notice, send updated LSPs
  – Link is removed from topology

• Node failure
  – All neighbors notice a link has failed
  – Failed node can’t update its own LSP
  – But it is OK: all links to node removed
Handling Changes (3)

• Addition of a link or node
  – Add LSP of new node to topology
  – Old LSPs are updated with new link

• Additions are the easy case ...
## DV/LS Comparison

<table>
<thead>
<tr>
<th>Property</th>
<th>Distance Vector</th>
<th>Link-State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>Distributed Bellman-Ford</td>
<td>Replicated Dijkstra</td>
</tr>
<tr>
<td>Efficient paths</td>
<td>Approx. with shortest paths</td>
<td>Approx. with shortest paths</td>
</tr>
<tr>
<td>Fair paths</td>
<td>Approx. with shortest paths</td>
<td>Approx. with shortest paths</td>
</tr>
<tr>
<td>Fast convergence</td>
<td>Slow – many exchanges</td>
<td>Fast – flood and compute</td>
</tr>
<tr>
<td>Scalability</td>
<td>Excellent – storage/compute</td>
<td>Moderate – storage/compute</td>
</tr>
</tbody>
</table>
Introduction to Computer Networks

Equal-Cost Multi-Path Routing
(§5.2.1)

Computer Science & Engineering
UNIVERSITY of WASHINGTON
Topic

• More on shortest path routes
  – Allow multiple shortest paths

Use ABCE and ABE from A→E
Multipath Routing

- Allow multiple routing paths from node to destination be used at once
  - Topology has them for redundancy
  - Using them can improve performance

- Questions:
  - How do we find multiple paths?
  - How do we send traffic along them?
**Equal-Cost Multipath Routes**

- **One form of multipath routing**
- **Extends shortest path model**
  - Keep set if there are ties
- **Consider A → E**
  - $ABE = 4 + 4 = 8$
  - $ABCE = 4 + 2 + 2 = 8$
  - $ABCDE = 4 + 2 + 1 + 1 = 8$
  - Use them all!
Source “Trees”

- With ECMP, source/sink “tree” is a directed acyclic graph (DAG)
  - Each node has set of next hops
  - Still a compact representation
Source “Trees” (2)

- Find the source “tree” for E
  - Procedure is Dijkstra, simply remember set of next hops
  - Compile forwarding table similarly, may have set of next hops

- Straightforward to extend DV too
  - Just remember set of neighbors
Source “Trees” (3)

Source Tree for E

E’s Forwarding Table

New for ECMP

<table>
<thead>
<tr>
<th>Node</th>
<th>Next hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B, C, D</td>
</tr>
<tr>
<td>B</td>
<td>B, C, D</td>
</tr>
<tr>
<td>C</td>
<td>C, D</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>--</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>F</td>
</tr>
<tr>
<td>H</td>
<td>C, D</td>
</tr>
</tbody>
</table>
ECMP Forwarding

- Could randomly pick a next hop for each packet based on destination
  - Balances load, but adds jitter

- Instead, try to send packets from a given source/destination pair on the same path
  - Source/destination pair is called a flow
  - Hash flow identifier to next hop
  - No jitter within flow, but less balanced
ECMP Forwarding (2)

Multipath routes from F to H

E’s Forwarding Choices

<table>
<thead>
<tr>
<th>Flow</th>
<th>Possible next hops</th>
<th>Example choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>F → H</td>
<td>C, D</td>
<td>D</td>
</tr>
<tr>
<td>F → C</td>
<td>C, D</td>
<td>D</td>
</tr>
<tr>
<td>E → H</td>
<td>C, D</td>
<td>C</td>
</tr>
<tr>
<td>E → C</td>
<td>C, D</td>
<td>C</td>
</tr>
</tbody>
</table>

Use both paths to get to one destination
Topic: Hierarchical Routing

• How to scale routing with hierarchy in the form of regions
  – Route to regions, not individual nodes
Internet Growth

- At least a billion Internet hosts and growing ...
Internet Routing Growth

- Internet growth translates into routing table growth (even using prefixes) ...
Impact of Routing Growth

1. Forwarding tables grow
   - Larger router memories, may increase lookup time

2. Routing messages grow
   - Need to keeps all nodes informed of larger topology

3. Routing computation grows
   - Shortest path calculations grow faster than the size of the network
Techniques to Scale Routing

1. IP prefixes
   - Route to blocks of hosts

2. Network hierarchy
   - Route to network regions

3. IP prefix aggregation
   - Combine, and split, prefixes

Last week
This topic
Next topic
Hierarchical Routing

• Introduce a larger routing unit
  – IP prefix (hosts) ← from one host
  – Region, e.g., ISP network

• Route first to the region, then to the IP prefix within the region
  – Hide details within a region from outside of the region
Hierarchical Routing (2)

Full table for 1A

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Line</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1B</td>
<td>1B</td>
<td>1</td>
</tr>
<tr>
<td>1C</td>
<td>1C</td>
<td>1</td>
</tr>
<tr>
<td>2A</td>
<td>1B</td>
<td>2</td>
</tr>
<tr>
<td>2B</td>
<td>1B</td>
<td>3</td>
</tr>
<tr>
<td>2C</td>
<td>1B</td>
<td>3</td>
</tr>
<tr>
<td>2D</td>
<td>1B</td>
<td>4</td>
</tr>
<tr>
<td>3A</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>3B</td>
<td>1C</td>
<td>2</td>
</tr>
<tr>
<td>4A</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>4B</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>4C</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>5A</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>5B</td>
<td>1C</td>
<td>5</td>
</tr>
<tr>
<td>5C</td>
<td>1B</td>
<td>5</td>
</tr>
<tr>
<td>5D</td>
<td>1C</td>
<td>6</td>
</tr>
<tr>
<td>5E</td>
<td>1C</td>
<td>5</td>
</tr>
</tbody>
</table>

Hierarchical table for 1A

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Line</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1B</td>
<td>1B</td>
<td>1</td>
</tr>
<tr>
<td>1C</td>
<td>1C</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1B</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1C</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1C</td>
<td>4</td>
</tr>
</tbody>
</table>
Hierarchical Routing (3)
Hierarchical Routing (4)

- Penalty is longer paths

1C is best route to region 5, except for destination 5C
Observations

• Outside a region, nodes have **one route** to all hosts within the region
  – This gives savings in table size, messages and computation

• However, each node may have a **different route** to an outside region
  – Routing decisions are still made by individual nodes; there is no single decision made by a region
Introduction to Computer Networks

IP Prefix Aggregation and Subnets (§5.6.2)
Prefixes and Hierarchy

- IP prefixes already help to scale routing, but we can go further
  - We can use a less specific (larger) IP prefix as a name for a region

<table>
<thead>
<tr>
<th>Region</th>
<th>IP1 /18</th>
<th>IP2 /18</th>
<th>IP3 /17</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

I’m the whole region

IP /16
Subnets and Aggregation

1. Subnets
   - Internally split one large prefix into multiple smaller ones

2. Aggregation
   - Externally join multiple smaller prefixes into one large prefix
Subnets

- Internally split up one IP prefix

One prefix sent to rest of Internet

32K addresses

16K
8K
4K
Aggregation

- Externally join multiple separate IP prefixes

One prefix sent to rest of Internet

Rest of Internet

ISP
Announcements

• Proxy project due today
• TOR project will be released on Wednesday
  – Part 1: design due in a week
  – Part 2: two more weeks for coding
• Homework 3 will be released next week; due in a week
Introduction to Computer Networks

Routing with Policy (BGP) (§5.6.7)
Structure of the Internet

- Networks (ISPs, CDNs, etc.) group hosts as IP prefixes
- Networks are richly interconnected, often using IXPs
Internet-wide Routing Issues

• Two problems beyond routing within an individual network

1. Scaling to very large networks
   – Techniques of IP prefixes, hierarchy, prefix aggregation

2. Incorporating policy decisions
   – Letting different parties choose their routes to suit their own needs

Yikes!
Effects of Independent Parties

- Each party selects routes to suit its own interests
  - e.g., shortest path in ISP

- What path will be chosen for A2 $\Rightarrow$ B1 and B1 $\Rightarrow$ A2?
  - What is the best path?
Effects of Independent Parties (2)

- Selected paths are longer than overall shortest path
  - And asymmetric too!
- This is a consequence of independent goals and decisions, not hierarchy
Routing Policies

• Capture the goals of different parties – could be anything
  – E.g., Internet2 only carries non-commercial traffic

• Common policies we’ll look at:
  – ISPs give TRANSIT service to customers
  – ISPs give PEER service to each other
Routing Policies – Transit

• One party (customer) gets TRANSIT service from another party (ISP)
  – ISP accepts traffic for customer from the rest of Internet
  – ISP sends traffic from customer to the rest of Internet
  – Customer pays ISP for the privilege
Routing Policies – Peer

• Both party (ISPs in example) get PEER service from each other
  – Each ISP accepts traffic from the other ISP only for their customers
  – ISPs do not carry traffic to the rest of the Internet for each other
  – ISPs don’t pay each other
Routes on the Internet

• Assume that you have transit & peering links
• Questions:
  – What strategy should ISPs use to select paths?
  – What do the resulting end-to-end paths look like?
Routing with BGP (Border Gateway Protocol)

- BGP is the **interdomain** routing protocol used in the Internet
  - Path vector, a kind of distance vector

```
ISP A
Prefix A1
Prefix A2

IXP
Prefix F1

ISP B
Prefix B1
Net F
Prefix F1 via ISP B, Net F at IXP
```
Routing with BGP (2)

- Different parties like ISPs are called AS (Autonomous Systems)
- Border routers of ASes announce BGP routes to each other
- Route announcements contain an IP prefix, path vector, next hop
  - Path vector is list of ASes on the way to the prefix; list is to find loops
- Route announcements move in the opposite direction to traffic
Routing with BGP (3)
Routing with BGP (4)

Policy is implemented in two ways:

1. Border routers of ISP announce paths only to other parties who may use those paths
   - Filter out paths others can’t use

2. Border routers of ISP select the best path of the ones they hear in any, non-shortest way
Routing with BGP (5)

- **TRANSIT**: AS1 says [B, (AS1, AS3)], [C, (AS1, AS4)] to AS2
Routing with BGP (6)

- **CUSTOMER (other side of TRANSIT):** AS2 says \([A, (AS2)]\) to AS1
Routing with BGP (7)

- **PEER**: AS2 says \([A, (AS2)]\) to AS3, AS3 says \([B, (AS3)]\) to AS2
Routing with BGP (8)

- AS2 hears two routes to B (via AS1, AS3) and chooses AS3 (Free!)
BGP Thoughts

• Much more beyond basics to explore!

• Policy is a substantial factor
  – Can we even be independent decisions will be sensible overall?

• Other important factors:
  – Convergence effects
  – How well it scales
  – Integration with intradomain routing
  – And more ...