**Topic**

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

![Diagram showing network nodes with "Flood!" and "... then compute" annotations]
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 Setting

Nodes compute their forwarding table in the same distributed setting as for distance vector:

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/links may fail, messages may be lost
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)
Phase 1: 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.
Phase 2: 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!
Forwarding Table

Source Tree for E (from Dijkstra)

E’s Forwarding Table

<table>
<thead>
<tr>
<th>To</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>C</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</td>
</tr>
</tbody>
</table>
Handling Changes

- On change, flood updated LSPs, and re-compute routes
  - E.g., nodes adjacent to failed link or node initiate

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

![Graph with nodes and labels](image-url)

Failure! G XXXX
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 ...
Link-State Complications

- Things that can go wrong:
  - Seq. number reaches max, or is corrupted
  - Node crashes and loses seq. number
  - Network partitions then heals

- Strategy:
  - Include age on LSPs and forget old information that is not refreshed

- Much of the complexity is due to handling corner cases (as usual!)
## DV/LS Comparison

<table>
<thead>
<tr>
<th>Goal</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>
IS-IS and OSPF Protocols

- Widely used in large enterprise and ISP networks
  - IS-IS = Intermediate System to Intermediate System
  - OSPF = Open Shortest Path First

- Link-state protocol with many added features
  - E.g., “Areas” for scalability
Topic

• How to route with multiple parties, each with their own routing policies
  – This is Internet-wide BGP routing
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
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 symmetric 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
Routing with BGP (Border Gateway Protocol)

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

![Diagram showing BGP routing between ISPs A and B through an IXP](image)

- Prefixes:
  - Prefix A1
  - Prefix A2
  - Prefix B1
  - Prefix F1
  - Prefix F2 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 ...