Overview

• Routing
• Distance Vector Routing
• Prim’s Spanning Tree Algorithm
Objective

• Understand how routing in the Internet is performed
• Learn about two essential routing algorithms that operate on graphs

Routing – Intro

• Shortest path routing
• Centralized
  • Each node has full “view” of network
  • Each node calculates shortest path using routing algorithm
  • “Link state algorithm”
  • (Exchange of link information always decentralized)

Image Source: https://www.kaspersky.com/blog/amazing-internet-maps/10441/
Routing - Intro

• Distributed approach
  • Each node computes best path without full view
  • Shortest path computed as link information is exchanged
  • “Distance vector algorithm”

Distance Vector Algorithm

• Features
  • Distributed
  • Iterative
  • Asynchronous
  • Each node reports local view
    • Cost to neighbors
    • Routes to others via neighbors
Distance Vector Algorithm

- Each node picks the best option
  - Bellman-Ford equation: 
    \[ d_x(y) = \min_v \{ c(x, v) + d_v(y) \} \]
- Information is exchanged as distance vector
  - Shortest distance to all nodes as seen locally
  - With enough exchanges, routing converges

Distance Vector Algorithm

- Bellman-Ford equation (dynamic programming)
  - \( d_x(y) := \text{cost of least cost path from } x \text{ to } y \)
  - Then \( d_x(y) = \min_v \{ c(x, v) + d_v(y) \} \)
Distance Vector Example

• Each node picks the best option
  • Bellman-Ford equation:
    \[ d_x(y) = \min_v \{ c(x,v) + d_v(y) \} \]
  • Information is exchanged as distance vector
    • Shortest distance to all nodes as seen locally
    • With enough exchanges, routing converges

• Clearly, \( d_V(z) = 5 \), \( d_X(z) = 3 \), \( d_W(z) = 3 \)

• B-F equation says:
  \[
  d_u(z) = \min \{ c(u,v) + d_v(z), \ c(u,x) + d_x(z), \ c(u,w) + d_w(z) \ }
  = \min \{ 2 + 5, \ 1 + 3, \ 5 + 3 \} = 4
  \]

node achieving minimum is next hop in shortest path, used in forwarding table
Distance Vector Algorithm

- \( D_x(y) = \) estimate of least cost from \( x \) to \( y \)
  - \( x \) maintains distance vector \( D_x = [D_x(y) : y \in N] \)

- Node \( x \):
  - Knows cost to each neighbor \( v \): \( c(x,v) \)
  - Maintains its neighbor’s distance vector. For each neighbor it maintains \( D_v = [D_v(y) : y \in N] \)

Distance Vector Algorithm

Key idea:

- From time to time, each node sends its own distance vector estimate to neighbors
- when \( x \) receives new DV estimate from neighbor, it updates its own DV using B-F equation:

\[
D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N
\]

- under minor, natural conditions, the estimate \( D_x(y) \) converge to the actual least cost \( d(x,y) \)
Distance Vector Algorithm
Iterative, asynchronous:
• Each local iteration caused by:
  • Local link cost change
  • DV update message from neighbor

Distributed:
• Each node notifies neighbor only when its DV changes
  • neighbors then notify their neighbors if necessary

\[
D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}
\]
\[
= \min\{2+0, 7+1\} = 2
\]
Distance Vector Algorithm

• Example:
  • Two updates
  • First, node 2
  • Then, node 3
Comparison – LS versus DV

Message complexity:

• **LS**: with $n$ nodes, $E$ links, $O(nE)$ msgs sent
• **DV**: exchange between neighbors only:
  • Convergence time varies

Speed of convergence:

• **LS**: $O(n^2)$ algorithm requires $O(nE)$ msgs
• **DV**: convergence time varies
  • Routing loops

Robustness: What happens if router malfunctions?

• **LS**:
  • Node can advertise incorrect link cost
  • Each node only computes its own table
• **DV**:
  • DV node can advertise incorrect path cost
  • Errors propagate through network

Prim’s Spanning Tree Algorithm

• Consider problem that games designers and Internet radio providers face
• Efficiently transfer information to anyone who may be listening
  • Gaming: game state (e.g. player positions) known at each player
  • Radio: client receive data to play content
Prim’s Spanning Tree Algorithm

Brute Force Approaches

- Keep list of all listeners and send individual messages
  - Four copies need to be sent in example on previous slide
- How many times would each router handle same message
  - Gaming: game state (e.g. player positions) known at each player
  - Radio: client receive data to play content
Broadcast Routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient

In-network Duplication

- **flooding:** when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- **controlled flooding:** node only broadcasts pkt if it hasn’t broadcast same packet before
  - node keeps track of packet ids already broadcasted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- **Spanning tree:** no redundant packets received by any node
Spanning Tree

• First, construct a spanning tree
• Nodes then forward/make copies only along spanning tree

(a) broadcast initiated at A
(b) broadcast initiated at D

Spanning Tree Creation

• Center node
• Each node sends unicast join message to center node
  • message forwarded until it arrives at a node already belonging to spanning tree

(a) stepwise construction of spanning tree (center: E)
(b) constructed spanning tree
Spanning Tree Example

Next Steps

• Lecture and discussion on 10/28
• Exam on 10/28