Routing

Dr. G. Omprakash

Assistant Professor, ECE, KLEF

Routing: Characteristics

- **Correctness**: The packets must reach their destination correctly
- **Simplicity**: The overhead in the routing process must be as low as possible
- **Robustness**: It is about the ability of the network to deliver packets in the face of localized failures and overloads
- **Stability**: The routing algorithms should be stable under all possible circumstances
    - No overload or underutilization during congestion
- **Fairness**: Every node in the network should get a fair chance of transmitting their packets
- **Optimality**: The routing algorithms should be optimal in terms of throughput and minimizing mean packet delays

Routing Algorithm Classification

- **Static Routing**
  - A single, permanent route is configured for each source–destination pair of nodes in the network
  - The route is computed in advance, offline, and downloaded to the routers when the network is booted
  - Do not base their routing decisions on the estimates of the current traffic or topology.
  - Also known as **Non-Adaptive** routing or **Fixed** routing
- **Dynamic Routing**
  - The routing decisions change as conditions on the network change
  - **Failure** and **Congestion**: conditions that influence routing decisions
  - Also known as **Adaptive** routing

Routing Algorithm Classification

- **Global**: All routers have complete topology, link cost info
  - :**Link state Algorithms**
- **Decentralized**: Iterative process of computation, exchange of info with neighbors
  - Routers initially only know link costs to attached neighbors
  - **Distance Vector Algorithms**
- ARPANET
  - First generation: Distance Vector Routing
  - Second Generation: Link-State Routing

Optimality Principle (Bellman 1957)

Optimality Principle: **It states that if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route**.

Routing algorithms which make decisions based on Topology

- Shortest Path
- Flooding
- Distance-Vector routing
- Link-state routing
- Hierarchical Routing
- Broadcast Routing
- Multicast Routing

Shortest Path Algorithm

Shortest Path Algorithm

- Build a graph of the network
  - Each node of the graph represents a router
  - Each edge of the graph representing a communication line
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph
- Here we describe Dijkstra algorithm to find the shortest path between source and destination (A to D)
- Each node is labeled (in parentheses) with its distance from the source node along the best known path
  - Initially, no paths are known, so all nodes are labeled with infinity.

Find the shortest path from A to D

- Start by marking node A as permanent, indicated by a filled-in circle
- Then we examine, in turn, each of the nodes adjacent to A
  - Relabeling each one with the distance to A.
  - we also label it with the node from which the probe was made
    - Used to reconstruct the final path later.
- Examine all the tentatively labeled nodes (nodes B and G)
  - Make the one with the smallest label permanent (node B)
    - Node-B becomes the new working node
- Start at B and examine all nodes adjacent to it.
- Repeat the above process
- Shortest path is A-B-E-F-H-D

Shortest Path Algorithm

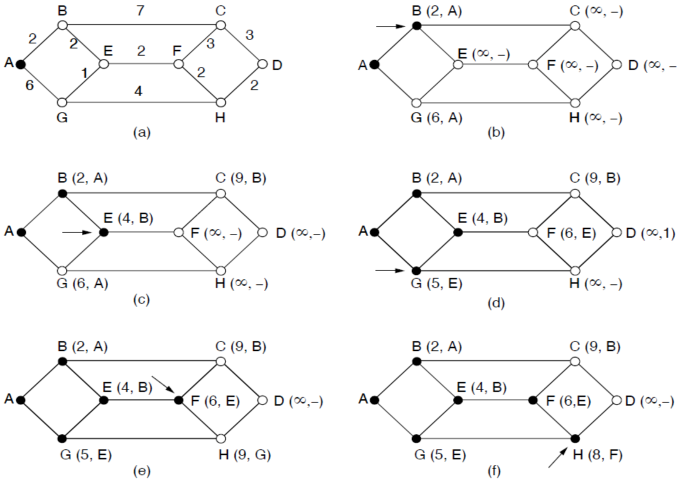


Figure: Computing the shortest path from A to D.

Flooding

Flooding: Every incoming packet to a node is sent out on every outgoing line except the one it arrived on

- Requires no network information like topology, load condition, cost of different paths
- Limitation: Flooding generates vast number of duplicate packets
  - Duplicates can become infinite unless some measures are taken to damp the process
- **Hop-Counter** is a measure to dampen the duplicate packets
  - Packet is discarded when the counter reaches zero.
- If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the network.

Flooding

- Flooding with a hop count can produce an exponential number of duplicate packets
- Routers keep track of which packets have been flooded, to avoid sending them out a second time
    - Each router then needs a list per source router telling which sequence numbers originating at that source
- To prevent the list from growing without bound
    - Each list should be augmented by a counter k
    - check if the packet has already been flooded (by comparing its sequence number to k);
        - if so, it is discarded
- Uses: It can be used as a building block for other routing algorithms that are more efficient
    - Effective for broadcasting information
    - Flooding is tremendously robust.
        - If large numbers of routers are blown (military targets), flooding will find a path if one exists

Distance-Vector Routing

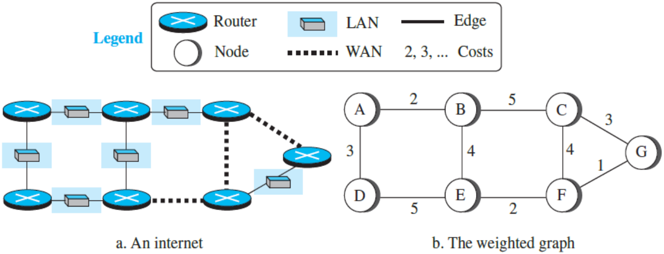
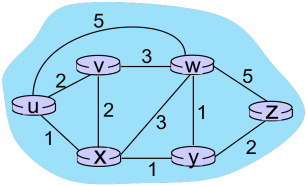


Figure: Internet and its graphical representation

Graph Abstraction



- graph: G=(N,E);
- N: set of routers = { u, v, w, x, y, z }
- E: set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }
- $C_{a,b}$ cost of **direct** link connecting *a* and *b*
  - Cost of link: $c_{w,z} = 5, c_{u,z} = \text{inf}$
- Cost: Distance, Number of hops, or estimated transit time

Least-cost trees

A least-cost tree is a tree with the source router as the root that spans the whole graph

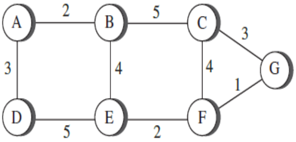
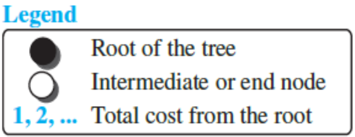


Figure: Weighted Graph



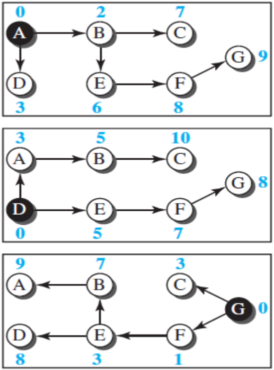


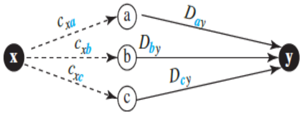
Figure: Only one shortest-path tree for each node

Distance Vector Routing

- In distance-vector routing, the first thing each node creates is its own least-cost tree
  - Cost is estimated delay
- A router continuously tells all its neighbors about what it knows
- To build a least-cost tree, we need to know about
  - **Bellman-Ford equation**
  - **Distance Vectors**
- **Bellman-Ford equation** is used to find the least cost (shortest distance) between a source node **x** and a destination node **y** through some intermediary nodes $(a, b, c, ...)$

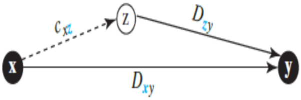
$$D_{xy} = min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

Bellman-Ford equation



$$D_{xy} = min\{(c_{xa}+D_{ay}),(c_{xb}+D_{by}),(c_{xc}+D_{cy}),\ldots\}$$

Update an existing least cost with a least cost through an intermediary node z

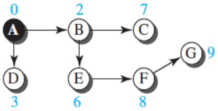


$$D_{xy} = min\{(c_{xz}+D_{zy}),D_{xy}\}$$

Distance Vectors

Why we need Distance Vectors?

We build the forwarding table from the distance vectors



a. Tree for node A

	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

b. Distance vector for node A

- Value of each cell defines the least cost from the root to the destination.
- Distance vector $\Longrightarrow$ gives only the least costs to the destinations
- Least-cost tree $\Longrightarrow$ gives the path to the destinations

First Distance Vector

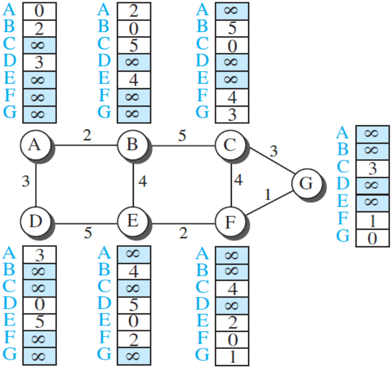


Figure: First distance vector for the Network

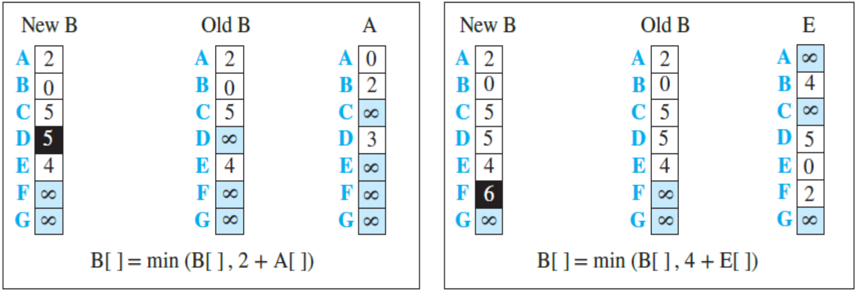
How to Build the distance vectors containing the least cost to the destination?

- Each node in an internet, when it is booted, creates a very rudimentary distance vector with the minimum information the node can obtain from its neighborhood
- *name* defines the root
- *indexes* define the destinations
- *value* of each cell defines the least cost from the root to the destination

Creating a distance Vector

- When a node (router) is booted, it sends a greeting messages out of its interfaces and discovers the identity of the immediate neighbors
- Makes a distance vector by inserting the discovered distances in the corresponding cells
  - leaves the value of other cells as infinity
- To improve these vectors, the nodes send a copy of the vector to all its immediate neighbors
- After a node receives a distance vector from a neighbor, it updates its distance vector using the Bellman-Ford equation

- First Event: Node A has sent its vector to node B. Node B updates its vector using the cost $c_{BA} = 2$.
- Second event: Node E has sent its vector to node B. Node B updates its vector using the cost $c_{EA} = 4$.



a. First event: B receives a copy of A's vector.

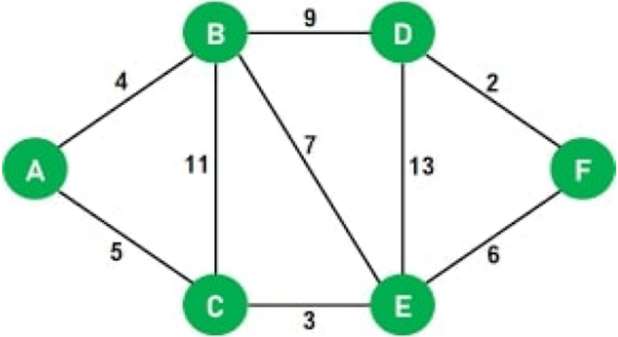
b. Second event: B receives a copy of E's vector.

Figure: Updating distance vector

Count to Infinity

- Problem with distance-vector routing:
  - any decrease in cost (good news) propagates quickly
  - any increase in cost (bad news) will propagate slowly
- In distance-vector routing, if link is broken $\implies$ cost=inf)
- Every other router should be aware of it immediately
  - But this takes time: Referred as *count to infinity*

Using Dijkstra's Algorithm, find the shortest distance from source vertex 'A' to the remaining vertices in the following graph.



Count to Infinity

A	B	C	D	E	
	·	·	·	·	Initially
	1	·	·	·	After 1 exchange
	1	2	·	·	After 2 exchanges
	1	2	3	·	After 3 exchanges
	1	2	3	4	After 4 exchanges

(a)

A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	·	·	·	·	

(b)

Figure: Count-to-infinity problem

Link-State Routing

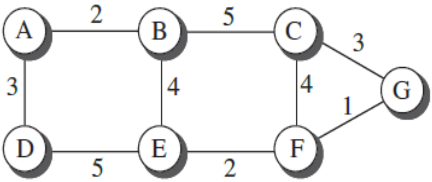
Link-State Routing

- The term link state defines the characteristic of a link (an edge) in the internet
- State of the link: cost associated with an edge
  - Lower costs are preferred to links with higher costs
  - If the cost of a link is infinity $\implies$
    - The link does not exist or has been broken.
- To create a least-cost tree, each node needs to have a complete map of the network $\implies$ It needs to know the state of each link.
- The collection of states for all links is called the **link-state database (LSDB)**[1]
- Variants of link state routing called IS-IS and OSPF are most widely used in the Internet today

[1] There is only one LSDB for the whole internet

Link-State Database (LSDB)

- LSDB can be represented as a two-dimensional array
  - Value of each cell = cost of the corresponding link



a. The weighted graph

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

Figure: Example of a link-state database

Building LSDB

Q. How can each node create this LSDB that contains information about the whole internet?

Ans: Flooding

What happens in Flooding?

- Each node can send some greeting messages (HELLO packet) to all its immediate neighbors
- Collect two pieces of information for each neighboring node:
  - The identity of the node
  - The cost of the link
- **LS Packet** (LSP) contains {Identity of Node+ Cost of the link}

Building Link State Packets

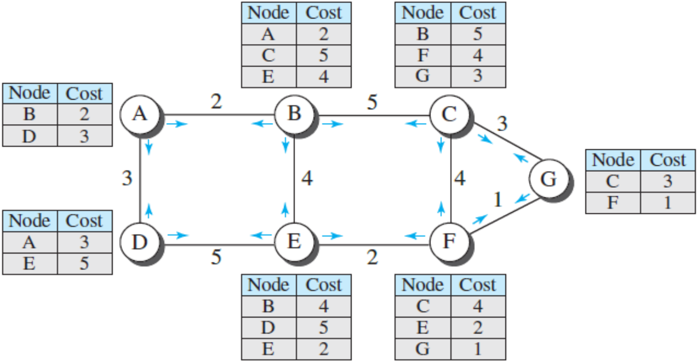


Figure: LSP is sent out of each interface

Distributing the Link State Packets

- All of the routers must get all of the link state packets quickly and reliably
- Use flooding to distribute the link state packets to all routers.
- To keep the flood in check
  - each packet contains a sequence number that is incremented for each new packet sent.
- When a new link state packet comes in, it is checked against the list of packets already seen
  - If it is new, it is forwarded on all lines except the one it arrived on
  - If it is a duplicate, it is discarded.

Link State Routing: Summary

- Discover its neighbors and learn their network addresses
- Set the distance or cost metric to each of its neighbors.
- Construct a packet telling all it has just learned
- Send this packet to and receive packets from all other routers.
- Compute the shortest path to every other router.

Hierarchical Routing

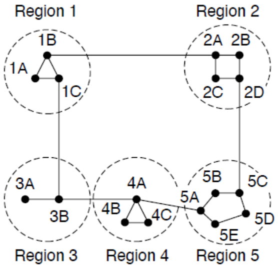
Hierarchical Routing: Motivation

- As networks grow in size
    - The router routing tables grow proportionally
    - Router memory consumed by ever-increasing tables
    - CPU time is needed to scan them
    - More bandwidth is needed to send status reports
- Even if every router could store the entire topology, recomputing shortest paths every time the network experienced changes in the topology would be prohibitive;
- At a certain point, it is no longer feasible for every router to have an entry for every other router
- So the routing will have to be done hierarchically, through the use of routing areas

Hierarchical Routing

- Unicast routing protocol used in the Internet.
- In Hierarchical routing: the routers are divided into **regions** or **areas**
- Each router knows all the details about how to route packets to destinations within its own region
  - but knows nothing about the internal structure of other regions
- The full routing table for router 1A has 17 entries
- Hierarchical routing has reduced the table from 17 to 7 entries.

Hierarchical Routing



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Acknowledge various sources for the images.
Thankyou