CS 5224

Routing

Dr. Chan Mun Choon
School of Computing, National University of Singapore

References

- Reading
  - Bertsekas and Gallager, "Data Networks," Chapter 5: Routing in Data Networks.
    - Read Chapter 5.1 and **5.2**
- Some slides are taken from the following source:
  - S. Keshav, "An Engineering Approach to Computer Networking", Chapter 11: Routing
  - Kurose and Ross, "Computer Networking: A Top Down Approach Featuring the Internet," Chapter 4: Network Layer and Routing

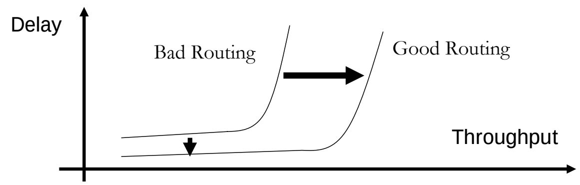
Outline

- Introduction
- Spanning tree routing in LAN
- Network Algorithms and Shortest Path Routing
- Distance Vector and Link State Routing

What is it?

- Process of finding a path from a source to every destination in the network
- Suppose you want to connect to Antarctica from your desktop
  - what route should you take?
  - does a "better" route exist?
    - "better" can be less hops, lower delay, higher bandwidth...
  - what if a link along the route goes down?
  - what if you're on a mobile wireless link?
- Routing deals with these types of issues

Objectives



- The effect of good routing is to
  - **increase throughput** for the same average delay per packet under high offered load conditions
  - **decrease average delay** per packet under low and moderate offered load conditions

Choices

- Centralized vs. distributed routing
  - centralized is simpler, but prone to failure and congestion
- Source-based vs. hop-by-hop
  - how much is in packet header?
  - Intermediate: *loose source route*
- Single vs. multiple path
  - primary and alternative paths
- State-dependent vs. state-independent
  - do routes depend on current network state (e.g. delay)

Flooding

- Flooding and broadcasting
  - Simple and does not need any pre-processing
  - Used when information is of interest to many nodes, of importance/urgency or no route exists
  - **Cost is high**
- The origin node sends a packet to all its neighbors. Neighbors relay the packet to their neighbors until all node receive the packet
  - Rule 1: A node will not relay the packet back to node it received from
  - Rule 2: A node will transmit the packet to its neighbor only once
  - Packet ID and sequence # of packets are kept to help enforce these rules
  - Total number of packets send is
    - between L and 2L, where L is the total number of bi-directional links

Flooding on Spanning Tree

- A spanning tree is a connected subgraph of the network that includes all N nodes and has no cycles (and therefore has N-1 links).
  - Note that $N < L$ if graph is connected
- A more efficient flooding can be performed using the spanning tree using only N-1 packets
  - Flooding over the entire network can be used to construct a spanning tree
  - Spanning tree can be used for routing

Routing on a Spanning Tree

- How many hops on the average from source to any node?
- How do you go from any source node to any destination node?
- When would you use a spanning tree for routing?

Outline

- Introduction
- Network Algorithms and Shortest Path Routing
- **Spanning tree routing in LAN**
- Distance Vector Routing & Link State Routing

Spanning Tree Routing in LAN

- Used for routing packets in the local area networks
  - Transparent bridging: routing without network layer
- Assume that each station has a unique ID known through a directory that can be access by all stations
  - For LAN, this may be performed using Address Resolution Protocol (ARP) which translates IP address to MAC address
- The location of a station is unknown and that stations can be turned on and off, and moved to another location
- **Stations** are connected through **bridges**

Spanning Tree Routing

- Bridges form a spanning tree
- First, one of the bridges is chosen to be the root of the spanning tree
  - Choice made by having each bridge broadcast its "unique id". The bridge with the lowest id becomes the root
  - In many cases, the "unique id" is a hardware identifier installed by the manufacturer and is guaranteed to be unique
  - Next a spanning tree is constructed
  - If one of the bridges fails, a new tree is computed

Bridge Routing

- Once the spanning tree is constructed, the ports of a bridge that is on the spanning tree are the **active ports**
- For each active port, a forwarding database (FDB) is maintained
  - Packets are routed based on the destination MAC address to the correct active port
- FDB is populated through **bridge learning**
- **Bridge Learning**:
  - When a port first becomes active, its FDB is empty
  - Learns the addresses of all stations on LANs directly connected to the active ports whenever these stations transmit packets (route on destination address, learning on source address)
  - (**Aging**) If the MAC address has not been transmitted on the LAN for some time, it is removed from the FDB

Bridge Learning

- What happens if destination is not in the FDB?
  - When the MAC address is not found in the FDB, the packet is broadcast along the spanning tree
  - Packet will eventually reach its destination
- Destination station replies and its location on the spanning tree is now known along the path the reply message

Example

Dealing with Failures

- The root bridge has to periodically transmit configuration messages
  - If no message is received from the root in a certain time period (15s), the bridge will discard all information about the root and the spanning tree algorithm will compute a new tree
- When a station moves, the FDB may be inconsistent, creating routing loops
  - Bridges are conservative about bringing a new station into the spanning tree
  - A timer (on the order of 30s or so) has to expired before the bridge forwards data to and from the new interface

Outline

- Introduction
- Spanning tree routing in LAN
- **Network Algorithms and Shortest Path Routing**
- Distance Vector and Link State Routing

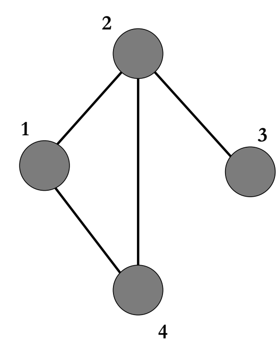
Shortest Path Routing

- Many practical routing algorithms are based on the notion of shortest path between two nodes.
  - Each link is assigned a positive number called its length
  - A shortest path routing algorithm routes each packet along the minimum length (or shortest) path between the source and destination
  - If length is always 1, then shortest path becomes minimum hop routing
  - By defining the length using other metrics like cost, delay etc, other paths can be obtained

Undirected Graph

- A graph G = (N,A) is a finite non-empty set N of nodes and a collection A of pairs of distinct nodes from N
- Each pair of nodes in A is called an arc
  - Maximum of one arc between any two nodes

Undirected Graph

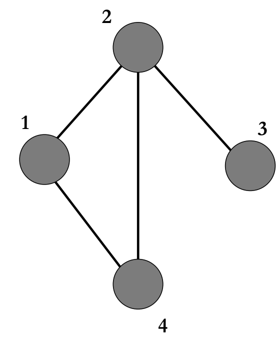


- G=(N,A)
- N = {1,2,3,4}
- A = {(1,2),(1,4),(2,3),(2,4)}
- $0.5|N|^2 > |A|$
- $|A| >= |N|-1$
  - if graph is connected

Undirected Graph

- A walk in a graph G is a sequence of nodes (n1, n2, … nl) of nodes such that each of the pairs (n1,n2) … (nl-1,nl) are arcs of G.
- A walk with no repeated nodes is a path
- A walk with n1 = nl, l > 3, and no repeated nodes other than n1 = nl is called a cycle

Undirected Graph (walk, path, cycle)



- (1,4,2,3)
  - walk, path
- (1,4,2,1)
  - walk, cycle
- (1,4,2,4,1)
  - walk
- (2)
  - walk, path
- (2,3,2)
  - Walk

Undirected Graph

- A graph is **connected** if for each node i there is a path to all other nodes.
- Lemma 1:
  - Let G=(N,A) be a connected graph and let S be any non-empty strict subset of N. Then at least one arc (i,j) exists such that i is an element of S, and j is not an element of S.
- G'=(N',A') is a **subgraph** of G if G' is a graph, N' is a subset of N, and A' is a subset of A

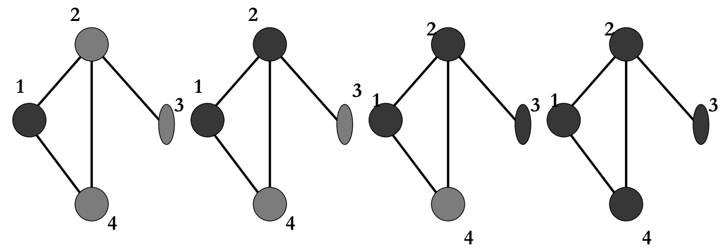
Undirected Graph

- A tree is a connected graph that contains no cycle.
- A spanning tree of a graph G is a subgraph of G that is a tree and includes all the nodes of G
- Construction of a spanning tree G'=(N',A')
  - Pick an arbitrary node, n, in N and let A' be empty
  - while (N != N') {
    - pick (i,j) an element of A such that I is an element of N', j is an element of N – N'
    - N' = N' U {j}
    - A' = A' U {(i,j)}
  - }

Spanning Tree

- The algorithm works because
  - Lemma 1 guarantees the existence of the arc (i,j)
  - Starting with a tree (empty set), the new graph remains a tree after the addition of a new link
- Proposition
  - A connected graph G always contains a spanning tree
  - |A| >= |N| - 1
  - G is a tree if and only if |A| = |N| - 1

Spanning Tree



Minimum Spanning Tree

- If the weights of arcs are different, we may consider building a minimum weight spanning tree (MST)
- A MST is a spanning tree with minimum sum of arc weights. The total spanning tree weight represents the cost of broadcasting a message to all nodes along the spanning tree
- Any subtree/subgraph of a **MST** is called a **fragment**
  - A node by itself is considered a fragment

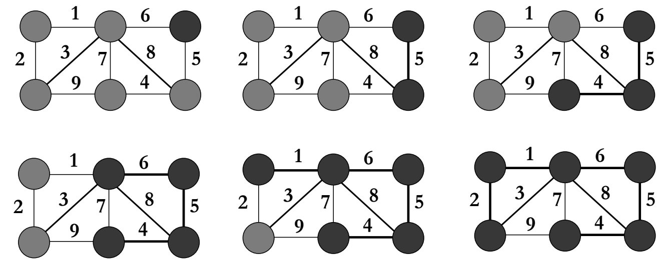
MST

- **Proposition 1**: Given a fragment F, let α = (i,j) be a minimum weight outgoing arc from F, while the node j is not in F. Then F, extended by arc α and node j is a fragment.
- **Proof (by contradiction):**
  - Denote by M the MST of which F is a subtree.
  - Assume α does not belong to M.
  - Since node j does not belong to F, there must be some other arc β that belongs to M and form a cycle together with α.
  - Deleting β from M and adding α results in another spanning tree which has less total weight, a contradiction

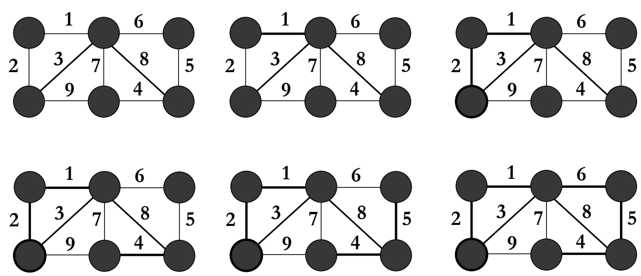
MST Algorithms

- Proposition 1 can be used as the basis for MST construction algorithms
- The Prim-Dijkstra Algorithm
  - Starts with an arbitrarily selected single node as a fragment and enlarges the fragment by successively adding minimum weight outgoing arcs
- Kruskal's Algorithm
  - Starts with all nodes being a single fragment
  - Successively combines two of the fragments by using the arcs with the overall minimum weight.
- All algorithms terminate in N -1 iterations

Example (Prim-Dijkstra)



Example (Krushal)

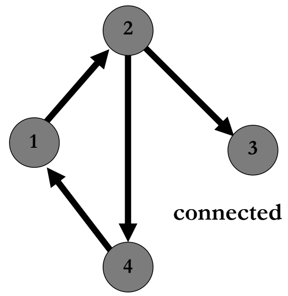


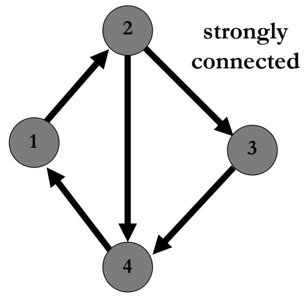
Directed Graph

- A directed graph (digraph) G = (N,A) is a finite non-empty set N of nodes and a collection A of ordered pairs of distinct nodes from N; each ordered pair of nodes in A is called a directed arc.
- A directed walk in G is a sequence of nodes (n1, n2, … nl) of nodes such that each of the pairs (n1,n2) … (nl-1,nl) are directed arcs of G.
- A directed walk with no repeated nodes is a directed path
- A directed walk with n1 = nl, **l > 2**, and no repeated nodes other than n1 = nl is called a directed cycle

Directed Graph

- A digraph is strongly connected if for each pair of nodes i and j there is a directed path
- A digraph is connected if the associated graph is connected





Shortest Path

- Given a digraph G in which each arc (i,j) is assigned some real number d_{ij} as the length or distance of the arc.
- Given any directed path (i,j,k,…,l,m), the length of p is defined as $d_{ij} + d_{jk} + \ldots + d_{lm}$.
- The shortest path problem is to find a minimum length directed path from i to m.

Bellman-Ford Algorithm (BFA)

- Suppose node 1 is the destination node and consider the problem of **finding a shortest path from every node to node 1**
  - Assume graph is strongly connected
- Let d_{ij} = infinity if (i,j) is not an arc of the graph
- A shortest walk from a given node i to node 1, subject to the constraint that the walk contains at most h arcs and goes through node 1 only once is called a shortest (<=h) walk and its length is denoted by D^h_i
- $D^h_1 = 0$ for all h (distance to yourself is always 0)

Bellman-Ford Algorithm (BFA)

Bellman-Ford Algorithm

- Initially, D^0_i= infinity for all i not equal 1
- For each iteration, h= 1,2,3 ….
  - $\mathbf{D^{h+1}_i = \min_j [d_{ij} + D^h_i]}$ for all i not equal to 1

1. The scalars $\mathbf{D^h_i}$ generated by the algorithm are equal to the shortest (<=h) walk lengths from node i to 1
2. The algorithm terminates after a finite number of iterations if and only if all cycles not containing node1 have non-negative length. Furthermore, if the algorithm terminates, it does so after at most h <= N iterations. At termination, $\mathbf{D^h_i}$ is the shortest path length from i to 1

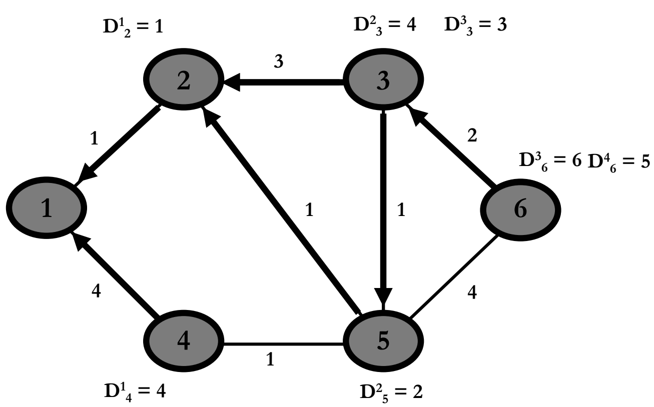
Bellman-Ford Algorithm (BFA)

- Sketch of Proof (by induction)
  - $D^{1}_{i} = d_{i1}$ for all i not equal to 1
  - If D^{k}_{i} is the shortest (<=k) walk length from i to 1, then D^{k+1}_{i} is the shortest (<= k+1) walk length from i to 1
  - $D^{k+1}_{i} <= D^{k}_{i}$
  - After h iterations, $D^{k}_{i} = D^{h}_{i}$ for k >= h
    - Length cannot be reduced by allowing longer walks
    - Not true if there is no negative-length cycle not including node 1
    - Since there is no cycle in a path, the longer path is at most |N| - 1. Therefore, after h = |N| - 1, D^{h}_{i} is the minimum
    - If D^{h}_{i} = infinity after |N| iterations, node i is not connected to node 1

BFA

- Computation Complexity:
  - Worst case $O(N^3)$ – why?
  - A tighter bound is O(mA)
    - A is the number of arcs, worst case is N^2
    - m is the maximum number of hops over all shortest paths, worst case is N

Bellman-Ford Algorithm (BFA)



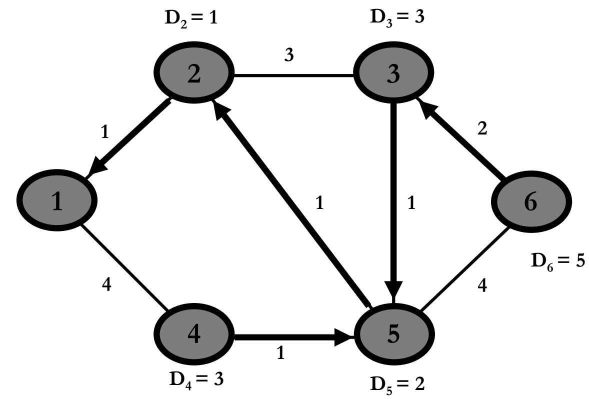
Dijkstra's Algorithm

- Dijkstra's algorithm requires that all arcs length are non-negative (which is true for most applications)
- Worst-case computation are less than Bellman-Ford algorithm
- General idea is to find the shortest paths in order of increasing path length

Dijkstra's Algorithm

- Let D_i be shortest path length from node i to node 1
- Let d_{ij} = infinity if (i,j) is not an arc of the graph
- Initial P = {1}, $D_1 = 0$, $D_j = d_{j1}$ for j not equal to 1
- Step 1 (Find closest node): Find i not in P such that
  - **$D_i = \min_{j \text{ not in } P} D_j$**, set P = P U {i}.
  - If P contains all nodes, END
- Step 2 (Update D): For all j not in P
  - **$D_j = \min_i[D_j, d_{ji}+D_i]$**, goto step 1

Dijkstra's Algorithm



Dijkstra's Algorithm

- Computation Complexity
  - Worst case $O(N^2)$, compare to $O(N^3)$ for Bellman-Ford
  - In cases where $A << N^2$ (spared graph), and m small, Bellman-Ford can terminate in a few iterations and O(mA) can be less than $O(N^2)$
  - Generally, for non-distributed applications, the two algorithms appear to be competitive

Distributed Asynchronous BFA

- Similar to the routing algorithm originally implemented in the ARPANET in 1969
- Requires very little information to be stored
- Sufficient to know the length of the outgoing link and the identity of every destination
- Uses the Bellman's Equation
  - $D_1 = 0$
  - $D_i = \min_{j \text{ in } N(i)} [d_{ij} + D_j]$ for all i not equal to 1, and N(i) is the set of neighbor nodes of node i

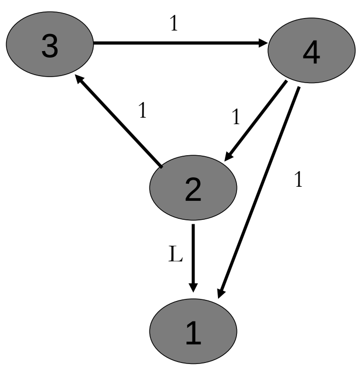
Distributed Asynchronous (BFA)

- Nodes asynchronously send to their neighbors estimates of $D_i(t)$
- Nodes receives messages from all their neighbors and updates their own estimates
- Estimates $D_i(t)$ converge to the correct shortest distance within finite time
- If there is a change in topology or length of a link, the old distance information is eventually purged from the system

Weakness

- Two known weakness
  1. In the worst case, the algorithm may require an excessive number of iterations to terminate
     - Not due to the asynchronous nature but due to arbitrary choice of initial conditions
  2. In the worst case, the algorithm requires an excessive number of message transmissions

Weakness 1: Counting to Infinity



Initially,

D2=3(3), D3=2(4), D4=1(1)

Link {4,1} breaks

D2=3(3), D3=2(4), D4=4(2)

D2=3(3), D3=5(4), D4=4(2)

D2=6(3), D3=5(4), D4=4(2)

D2=6(3), D3=5(4), D4=7(2)

…….

D2=L(1), D3=L+2(4), D4=L+1(2)

What happen if L = infinity?

Outline

- Introduction
- Network Algorithms and Shortest Path Routing
- Spanning tree routing in LAN
- **Distance Vector Routing & Link State Routing**

WAN Routing

- Environment
  - links and routers unreliable
  - alternative paths scarce
  - traffic patterns can change rapidly
- Two key algorithms
  - **distance vector**
  - **link-state**
- Both assume router knows
  - address of each neighbor
  - cost of reaching each neighbor
- Both allow a router to determine global routing information by talking to its neighbors

Routing in the Internet

- An autonomous system (AS) is a collection of routers under the same administrative and technical control, and that all run the same routing protocol among themselves
- Intra-AS routing protocols
  - Routing Information Protocol (**RIP**) RFC 1058,2453
  - Open Shortest Path First (**OSPF**) RFC 2328
- Inter-AS routing protocols
  - Border Gateway Protocol (**BGP** version 4) RFC 1771,1772,1773

Distance Vector

- Node tells its neighbors its best idea of distance to *every* other node in the network
- Node receives these *distance vectors* from its neighbors
- Updates its notion of best path to each destination, and the next hop for this destination
- Features
  - distributed
  - adapts to traffic changes and link failures
  - suitable for networks with multiple administrative entities
- Used in BGP and RIP

Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

asynchronous:

- nodes need *not* exchange info/iterate in lock step!

distributed:

- each node communicates *only* with directly-attached neighbors

Distance Table data structure

- each node has its own
- row for each possible destination
- column for each directly-attached neighbor to node

- **The heart of the Distance Vector algorithm is the Distributed Asynchronous Bellman-Ford Algorithm**

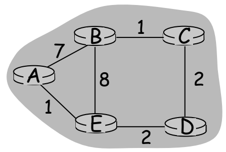
Distance and routing table of Node E

Distance table ($D^E()$, cost to destination via):

destination	A	B	D
A	(1)	14	5
B	7	8	(5)
C	6	9	(4)
D	4	11	(2)

Routing table:

destination	Outgoing link to use, cost
A	A,1
B	D,5
C	D,4
D	D,4



Distance table → Routing table

Dealing with problems

- Path vector
  - DV carries path to reach each destination; solves the count-to-infinity problem
  - used in the BGP protocol in the Internet core
  - trade-off a larger routing table and extra control overhead for robustness
- Split horizon
  - use of path vector increases routing table significantly
  - **Alternative**: never tell neighbor cost to X if neighbor is next hop to X
  - doesn't work for 3-way count to infinity

Dealing with problems

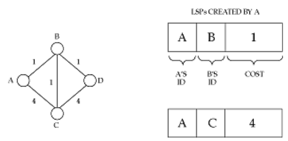
- Split Horizon with Poison Reverse
  - slight improvement over split horizon
  - tell neighbor cost to X is infinity if neighbor is next hop to X
  - Still doesn't work for 3-way count to infinity but can accelerates converge sometimes
  - Used in RIP
- Triggered updates
  - exchange routes on change, instead of on timer
  - faster count up to infinity
  - Used in RIP

Link state routing

- In distance vector, router knows only *cost* to each destination
  - hides information, causing problems
- In link state, router knows entire network topology, and computes shortest path by itself
  - independent computation of routes
- Key elements
  - **topology dissemination**
  - **computing shortest routes**
- **OSPF uses link state routing**

Link state: topology dissemination

- A router describes its neighbors with a *link state packet (LSP)*



- Use *controlled flooding* to distribute this everywhere
  - store an LSP in an *LSP database*
  - if new, forward to every interface other than incoming one
  - a network with E edges will copy at most 2E times

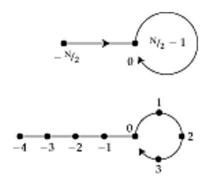
Sequence numbers

- How do we know an LSP is new?
- Use a sequence number in LSP header
- Greater sequence number is newer
- What if sequence number wraps around?
  - smaller sequence number is now newer!
  - use a large (enough) sequence space
- On boot up, what should be the initial sequence number?
  - have to somehow purge old LSPs
  - two solutions
    - aging
    - lollipop sequence space

Aging

- Creator of LSP puts timeout value in the header
- Router removes LSP when it times out
  - also floods this information to the rest of the network (why?)
- So, on booting, router just has to wait for its old LSPs to be purged
- But what age to choose?
  - if too small
    - purged before fully flooded (why?)
    - needs frequent updates
  - if too large
    - router waits idle for a long time on rebooting

A better solution



- Need a *unique* start sequence number
  - Use a lollipop sequence space
- a is older than b if:
  - a < 0 and a < b
  - a > o, a < b, and b-a < N/4
  - a > 0, b > 0, a > b, and a-b > N/4

More on lollipops

- If a router gets an older LSP, it tells the sender about the newer LSP
- So, newly booted router quickly finds out its most recent sequence number
- It jumps to one more than that
- -N/2 is a *trigger* to evoke a response from community memory
- Used in OSPFv1 (RFC 1131)

Link-State Routing Algorithm

- uses Dijkstra's algorithm
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives routing table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Link State (LS) vs. Distance Vector (DV)*

- Criteria
  - Memory
  - Bandwidth Consumed
  - Computation
  - Robustness
  - Functionality
  - Speed of Convergence

* Radia Perlman, "Interconnections: Bridges, Routers, Switches and Internetworking Protocols," Addison-Wesley, 2000.

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent each
- DV: exchange between neighbors only, convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network