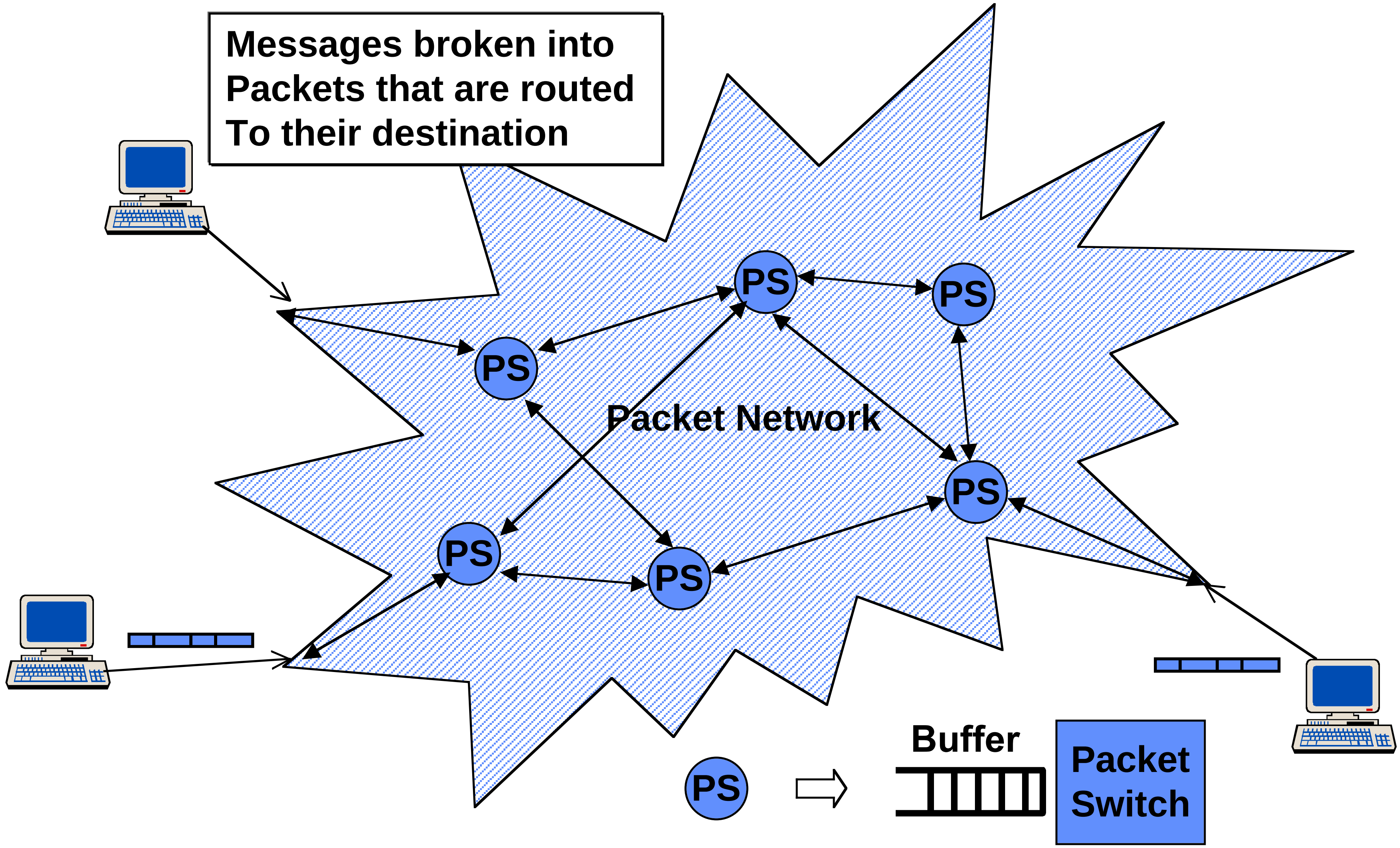
Lectures 21: Routing in Data Networks

Eytan Modiano

Packet Switched Networks

Messages broken into Packets that are routed To their destination



Routing

- **Must choose routes for various origin destination pairs (O/D pairs) or for various sessions**
  - **Datagram routing: route chosen on a packet by packet basis**
    - **Using datagram routing is an easy way to split paths**
  - **Virtual circuit routing: route chosen a session by session basis**
  - **Static routing: route chosen in a prearranged way based on O/D pairs**

Broadcast Routing

- **Route a packet from a source to all nodes in the network**

- **Possible solutions:**

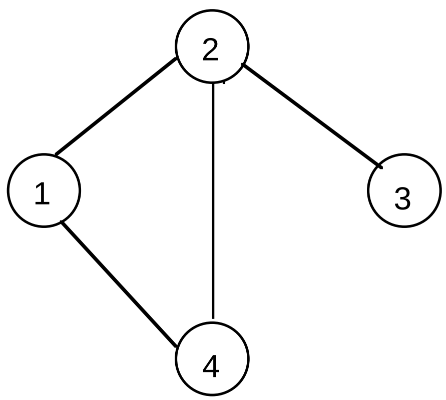
  – **Flooding: Each node sends packet on all outgoing links**

    Discard packets received a second time

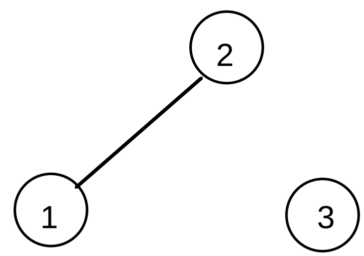
  – **Spanning Tree Routing: Send packet along a tree that includes all of the nodes in the network**

Graphs

- **A graph G = (N,A) is a finite nonempty set of nodes and a set of node pairs A called arcs (or links or edges)**



N = {1,2,3,4}
A = {(1,2),(2,3),(1,4),(2,4)}

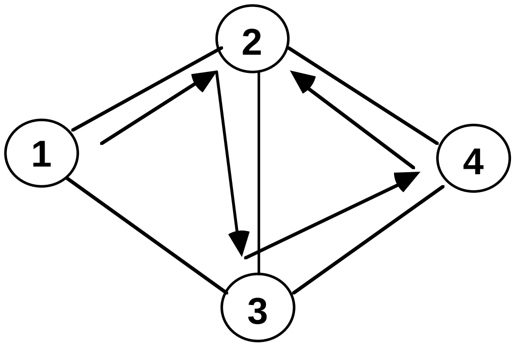


N = {1,2,3}
A = {(1,2)}

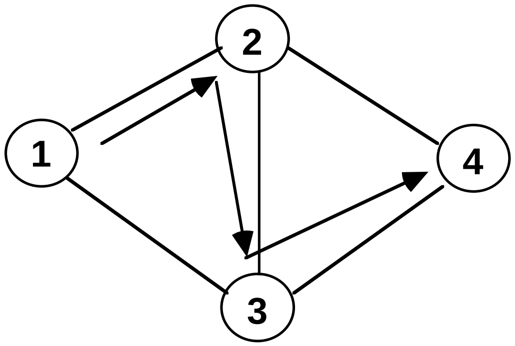
Walks and paths

- A walk is a sequence of nodes (n1, n2, ...,nk) in which each adjacent node pair is an arc.

- A path is a walk with no repeated nodes.



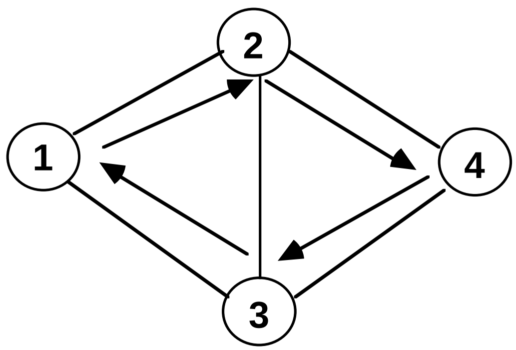
Walk (1,2,3,4,2)



Path (1,2,3,4)

Cycles

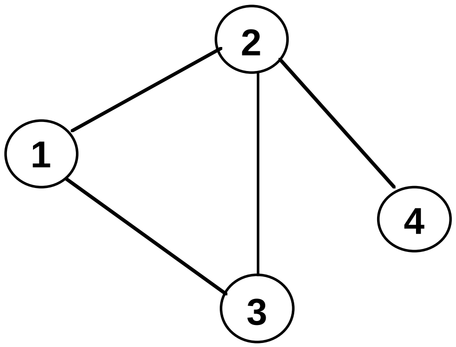
- A cycle is a walk (n1, n2,...,nk) with n1 = nk, k>3, and with no repeated nodes except n1 = nk



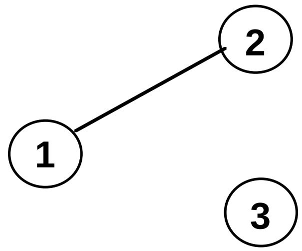
Cycle (1,2,4,3,1)

Connected graph

- A graph is connected if a path exists between each pair of nodes.



Connected

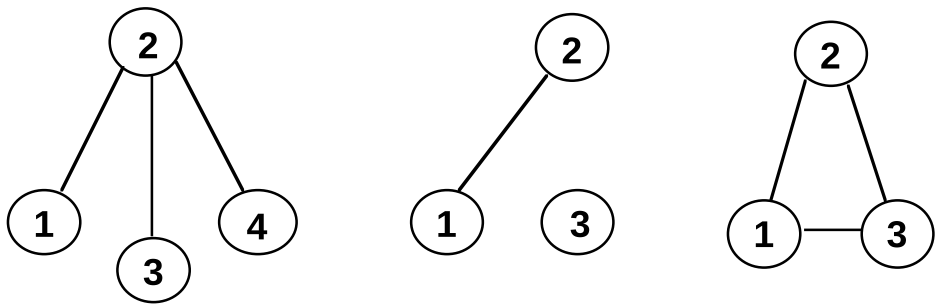


Unconnected

- An unconnected graph can be separated into two or more connected components.

Acyclic graphs and trees

- An acyclic graph is a graph with no cycles.
- A tree is an acyclic connected graph.



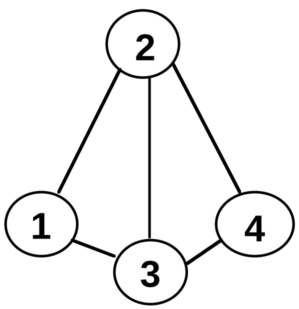
Acyclic, connected | unconnected not tree | Cyclic, not tree

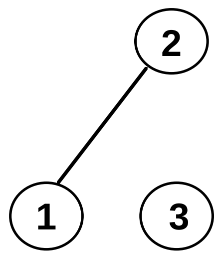
- The number of arcs in a tree is always one less than the number of nodes
  - Proof: start with arbitrary node and each time you add an arc you add a node => N nodes and N-1 links. If you add an arc without adding a node, the arc must go to a node already in the tree and hence form a cycle

Subgraphs

- G' = (N',A') is a subgraph of G = (N,A) if
  - 1) G' is a graph
  - 2) N' is a subset of N
  - 3) A' is a subset of A

- One obtains a subgraph by deleting nodes and arcs from a graph
  - Note: arcs adjacent to a deleted node must also be deleted

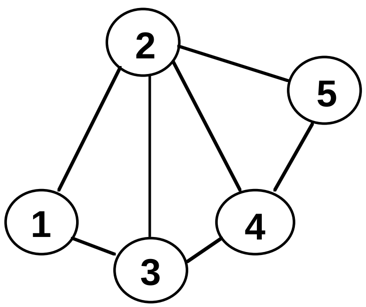




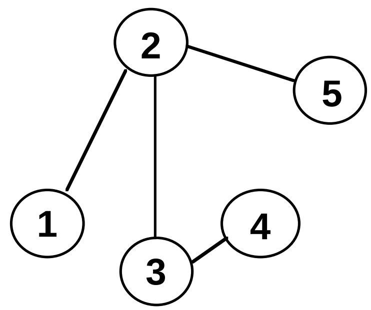
– Graph G Subgraph G' of G

Spanning trees

- T = (N',A') is a spanning tree of G = (N,A) if
  - T is a subgraph of G with N' = N and T is a tree



Graph G



Spanning tree of G

Spanning trees

- Spanning trees are useful for disseminating and collecting control information in networks; they are sometimes useful for routing

- To disseminate data from Node n:
  - Node n broadcasts data on all adjacent tree arcs
  - Other nodes relay data on other adjacent tree arcs

- To collect data at node n:
  - All leaves of tree (other than n) send data
  - Other nodes (other than n) wait to receive data on all but one adjacent arc, and then send received plus local data on remaining arc

General construction of a spanning tree

- Algorithm to construct a spanning tree for a connected graph G = (N,A):

  1) Select any node n in N; N' = {n}; A' = { }

  2) If N' = N, then stop (T=(N',A') is a spanning tree)

  3) Choose $(i,j) \in A$, $i \in N'$, $j \notin N'$

  $N' := N' \cup \{j\}$; $A' := A' \cup \{(i,j)\}$; go to step 2

- Connectedness of G assures that an arc can be chosen in step 3 as long as $N' \neq N$

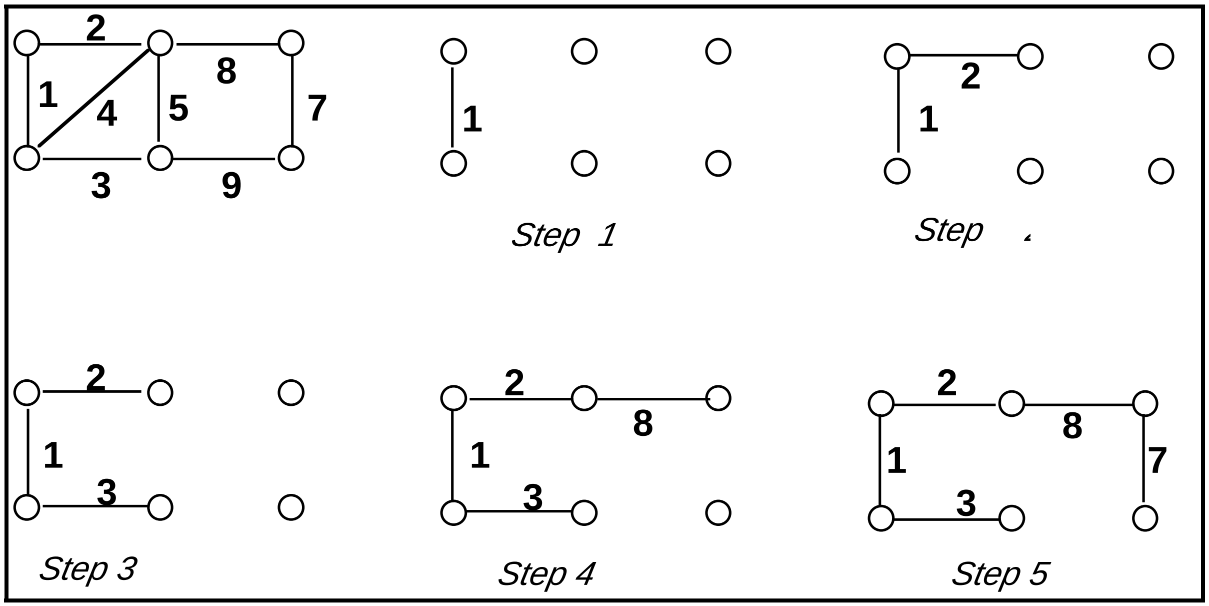
- Is spanning tree unique?

- What makes for a good spanning tree?

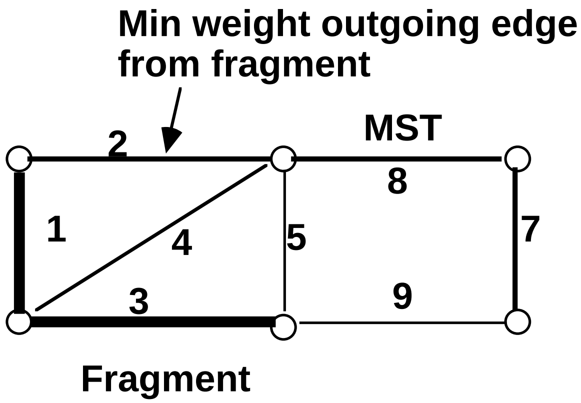
Minimum Weight Spanning Tree (MST)

- **Generic MST algorithm steps:**
  - Given a collection of subtrees of an MST (called fragments) add a minimum weight outgoing edge to some fragment
- **Prim-Dijkstra: Start with an arbitrary single node as a fragment**
  - Add minimum weight outgoing edge
- **Kruskal: Start with each node as a fragment;**
  - Add the minimum weight outgoing edge, minimized over all fragments

Prim-Dijkstra Algorithm



Kruskal's Algorithm Example



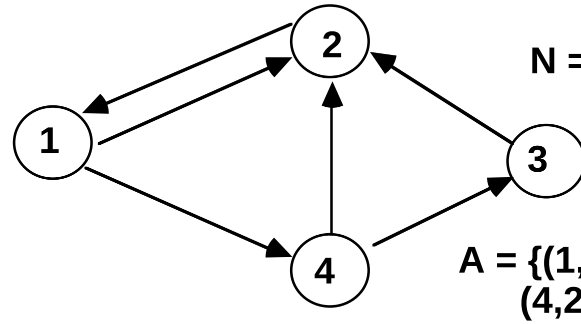
- **Suppose the arcs of weight 1 and 3 are a fragment**
  - Consider any spanning tree using those arcs and the arc of weight 4, say, which is an outgoing arc from the fragment.
  - Suppose that spanning tree does not use the arc of weight 2.
  - Removing the arc of weight 4 and adding the arc of weight 2 yields another tree of smaller weight.
  - Thus an outgoing arc of min weight from fragment must be in MST.

Shortest Path routing

- Each link has a cost that reflects
  - The length of the link
  - Delay on the link
  - Congestion
  - $$ cost
- Cost may change with time
- The length of the route is the sum of the costs along the route
- The shortest path is the path with minimum length
- Shortest Path algorithms
  - Bellman-Ford: centralized and distributed versions
  - Dijkstra's algorithm
  - Many others

Directed graphs (digraphs)

- A directed graph (digraph) G = (N,A) is a finite nonempty set of nodes N and a set of ordered node pairs A called directed arcs.



N = {1,2,3,4}

A = {(1,2), (2,1),(1,4),
(4,2), (4,3),(3,2)}

- Directed walk: (4,2,1,4,3,2)

- Directed path: (4,2,1)

- Directed cycle: (4,2,1,4)

- Data networks are best represented with digraphs, although typically links tend to be bi-directional (cost may differ in each direction)
  - For simplicity we will use bi-directional links of equal costs in our examples

Bellman Ford algorithm

- Finds the shortest paths, from a given source node, say node 1, to all other nodes.

- General idea:

  - First find the shortest single arc path,
  - Then the shortest path of at most two arcs, etc.
  - Let $d_{ij}=\infty$ if (i,j) is not an arc.

- Let $D_i(h)$ be the shortest distance from 1 to i using at most h arcs.
  - $D_i(1) = d_{1i}$; $i \neq 1$ $D_1(1) = 0$
  - $D_i(h+1) = \min_{\{j\}} [D_j(h) + d_{ji}]$; $i \neq 1$ $D_1(h+1) = 0$

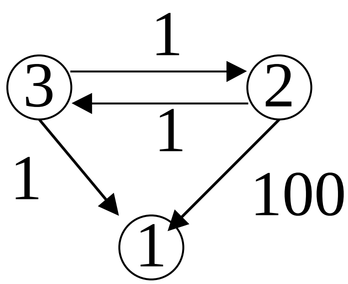
- If all weights are positive, algorithm terminates in N-1 steps.

Bellman Ford - example

Distributed Bellman Ford

- Link costs may change over time
  - Changes in traffic conditions
  - Link failures
  - Mobility
- Each node maintains its own routing table
  - Need to update table regularly to reflect changes in network
- Let Di be the shortest distance from node i to the destination
  - Di = min {j} [Dj + dij] : update equation
- Each node (i) regularly updates the values of Di using the update equation
  - Each node maintains the values of dij to its neighbors, as well as values of Dj received from its neighbors
  - Uses those to compute Di and send new value of Di to its neighbors
  - If no changes occur in the network, algorithm will converge to shortest paths in no more than N steps

Slow reaction to link failures



- Start with D3=1 and D2=100
  - After one iteration node 2 receives D3=1 and D2 = min [1+1, 100] = 2
- In practice, link lengths occasionally change
  - Suppose link between 3 and 1fails (I.e., d31=infinity)
  - Node 3 will update D3 = d32 + D2 = 3
  - In the next step node 2 will update: D2 = d23+D3 = 4
  - It will take nearly 100 iterations before node 2 converges on the correct route to node 1
- Possible solutions:
  - Propagate route information as well
  - Wait before rerouting along a path with increasing cost

    Node next to failed link should announce D=infinity for some time to prevent loops

Dijkstra's algorithm

- Find the shortest path from a given source node to all other nodes
  - Requires non-negative arc weights
- Algorithm works in stages:
  - Stage k: the k closest nodes to the source have been found
  - Stage k+1: Given k closest nodes to the source node, find k+1st.
- Key observation: the path to the k+1st closest nodes includes only nodes from among the k closest nodes
- Let M be the set of nodes already incorporated by the algorithm
  - Start with Dn = dsn for all n (Dn = shortest path distance from node n to the source node
  - Repeat until M=N

    Find node w$\notin$M which has the next least cost distance to the source node

    Add w to M

    Update distances: Dn = min [Dn, Dw + dwn] (for all nodes n $\notin$M)
  - Notice that the update of Dn need only be done for nodes not already in M and that the update only requires the computation of a new distance by going through the newly added node w.

Dijkstra example

Dijkstra's algorithm implementation

- **Centralized version: Single node gets topology information and computes the routes**
  - **Routes can then be broadcast to the rest of the network**

- **Distributed version: each node i broadcasts {dij all j} to all nodes of the network; all nodes can then calculate shortest paths to each other node**
  - **Open Shortest Path First (OSPF) protocol used in the internet**

Routing in the Internet

- Autonomous systems (AS)
  - Internet is divided into AS's each under the control of a single authority
- Routing protocol can be classified in two categories
  - Interior protocols - operate within an AS
  - Exterior protocols - operate between AS's

- Interior protocols
  - Typically use shortest path algorithms
    - Distance vector - based on distributed Bellman-ford
    - link state protocols - Based on "distributed" Dijkstra's

Distance vector protocols

- Based on distributed Bellman-Ford
  - Nodes exchange routing table information with their neighbors

- Examples:
  - Routing information protocols (RIP)
    - Metric used is hop-count (dij=1)
    - Routing information exchanged every 30 seconds

  - Interior Gateway Routing Protocol (IGRP)
    - CISCO proprietary
    - Metric takes load into account
    - Dij ~ $1/(\mu-\lambda)$ (estimate delay through link)

    - Update every 90 seconds
    - Multi-path routing capability

Link State Protocols

- **Based on Dijkstra's Shortest path algorithm**
  - **Avoids loops**
  - **Routers monitor the state of their outgoing links**
  - **Routers broadcast the state of their links within the AS**
  - **Every node knows the status of all links and can calculate all routes using dijkstra's algorithm**
    - **Nonetheless, nodes only send packet to the next node along the route with the packets destination address. The next node will look-up the address in the routing table**

- **Example: Open Shortest Path First (OSPF) commonly used in the internet**

- **Link State protocols typically generate less "control" traffic than Distance-vector**

Inter-Domain routing

- **Used to route packets across different AS's**

- **Options:**

  – **Static routing - manually configured routes**

  – **Distance-vector routing**

    Exterior Gateway Protocol (EGP)

    Border Gateway Protocol (BGP)

- **Issues**
  – **What cost "metric" to use for Distance-Vector routing**

    Policy issues: Network provider A may not want B's packets routed through its network or two network providers may have an agreement

    Cost issues: Network providers may charge each other for delivery of packets