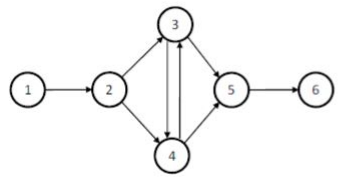
0.0.1 Graphs

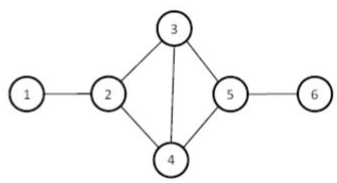
Directed:



a) Directed graph

$$
\begin{aligned}
G = &\ (V, E) \\
V = &\ \{v_1, v_2, v_3, v_4, v_5, v_6\}, && |V| = 6 \\
E = &\ \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_4), \\
&\ (v_4, v_3), (v_3, v_5), (v_4, v_5), (v_5, v_6)\}, && |E| = 8
\end{aligned}
$$

Undirected:

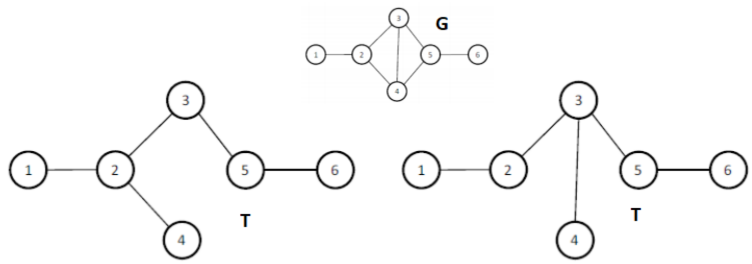


b) Undirected graph

$$
\begin{aligned}
G = &\ (V, E) \\
V = &\ \{v_1, v_2, v_3, v_4, v_5, v_6\}, && |V| = 6 \\
E = &\ \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_4), \\
&\ (v_3, v_5), (v_4, v_5), (v_5, v_6)\}, && |E| = 7
\end{aligned}
$$

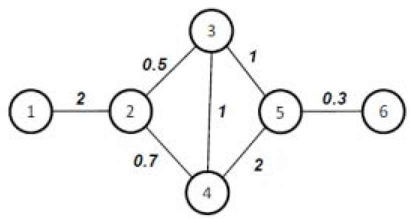
0.0.2 Tree

- Tree T = (V,E)
  - Graph with no cycles
  - |E| = |V| - 1
  - Any two V connected by only one E
- A tree T spans a graph G = (V,E) (spanning tree) if
  - T = (V,E') & E $\subseteq$ E' (T must have the same vertices and a subset of the graph edges)



0.0.3 Shortest Path Trees

- Graphs and Trees can be weighted
  - G=(V,E,W)
  - T=(V,E',W)



- Total cost of a tree T

$$C_{total}(T) = \sum_{i=1}^{|E|}$$

  (sum of all tree edges weight)

- Minimum spanning tree T*

$$C_{total}(T^*) = min(C_{total}(T))$$

  - algorithms used to compute MST: Prism, Kruskal

- **Shortest Path Tree (SPT) rooted at vertex** s
  - tree composed by the **union of the shortest paths between** s **and each vertex of** G
  - algorithms used to compute SPT: **Dijkstra**, **Bellman-Ford**

- Computer networks use **Shortest Path Trees**

0.1 Routing in Layer 3 Networks

0.1.1 Forwarding, Routing

- **Forwarding** → data plane
  - directing packet from input to output link
  - using a forwarding table
- **Routing** → control plane
  - computing paths the packets will follow
  - routers exchange messages
  - each router creates its forwarding table

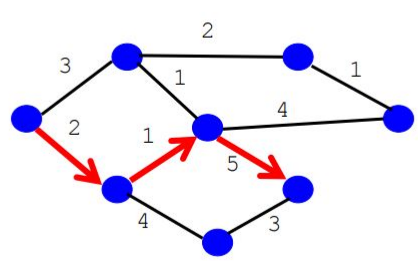
0.1.2 Importance of Routing

- End-to-end performance
  - path affects quality of service
  - delay, throughput, packet loss
- Use of network resources
  - balance traffic over routers and links
  - avoiding congestion by directing traffic to less-loaded links
- Transient disruptions
  - failures, maintenance
  - limiting packet loss and delay during changes

0.1.3 Shortest-Path Routing

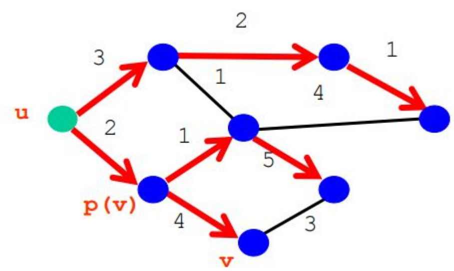
Path-selection model

- Destination-based
- Load-insensitive (ex: static link weights)
- Minimum hop count or minimum sum of link weights



0.1.4 Shortest-Path Problem

- Given a network topology with link costs
  - **c(x,y)** - link cost from node x to node y
  - ∞ if x and y are not direct neighbors
- Compute the least-cost paths from source **u** to all nodes
  - **p(v)** - node predecessor of node v in the path from u



0.1.5 Dijkstra's Shortest-Path Algorithm

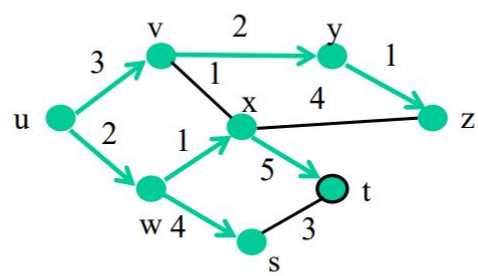
- Iterative algorithm
  - After k iterations $\rightarrow$ known least-cost paths to k nodes
- **S** $\rightarrow$ set of nodes for which least-cost path is known
  - Initially, **S={u}**, where **u** is the source node
  - Add one node to **S** in each iteration
- **D(v)** $\rightarrow$ current cost of path from source to node **v**
  - Initially
    - **D(v)=c(u,v)** for all nodes adjacent to **u**
    - **D(v)=**∞ for all other nodes **v**
  - Continually update **D(v)** when shorter paths are learned

```
Initialization:
  S = {u}
  for all nodes v
    if v adjacent to u {
       D(v) = c(u,v) }
    else D(v) = ∞

 Loop
   find node w not in S with the smallest D(w)
   add w to S
   update D(v) for all v adjacent to w and not in S:
      D(v) = min{D(v), D(w) + c(w,v)}
 until all nodes in S
```

0.1.6 Shortest-Path Tree

- Shortest-path tree from u



- Forwarding table at u

	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

0.1.7 Link-State Routing

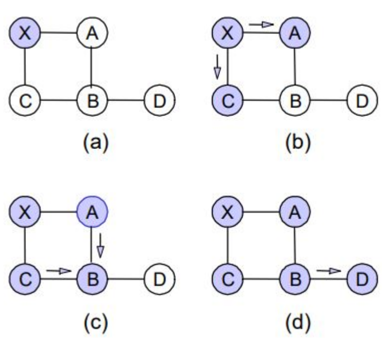
- Each router keeps track of its incident links
  - link up, link down
  - cost on the link
- Each router broadcasts link state
  - every router gets a complete view of the graph
- **Each router runs Dijkstra's algorithm**, to
  - compute the shortest paths
  - construct the forwarding table

0.1.8 Detection of Topology Changes

- Beacons generated by routers on links
  - periodic "hello" messages in both directions
  - few missed "hellos" → link failure

0.1.9 Broadcasting the Link State

- How to Flood the link state?
  - every node sends link-state information through adjacent links
  - next nodes forward that info to all links except the one where the information arrived

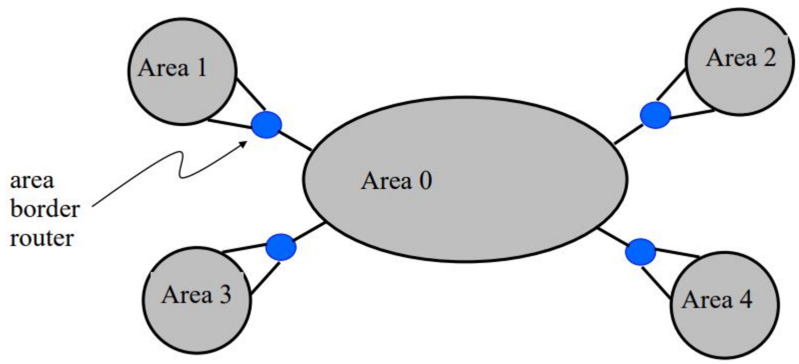


- When to initiate flooding?
  - Topology change
    - link or node failure/recovery
    - link cost change

  - Periodically
    * refresh link-state information
    * typically 30 minutes

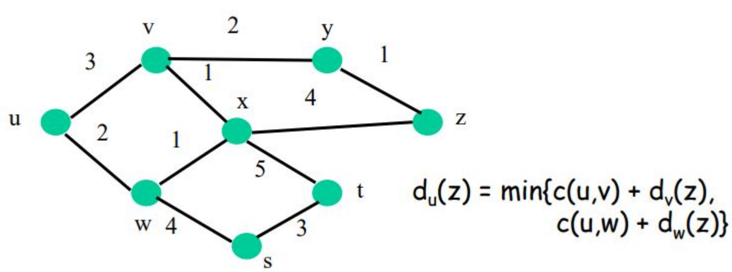
0.1.10 Scaling Link-State Routing

- Overhead of link-state routing
  - flooding link-state packets throughout the network
  - running Dijkstra's shortest-path algorithm
- Introducing hierarchy through "areas"



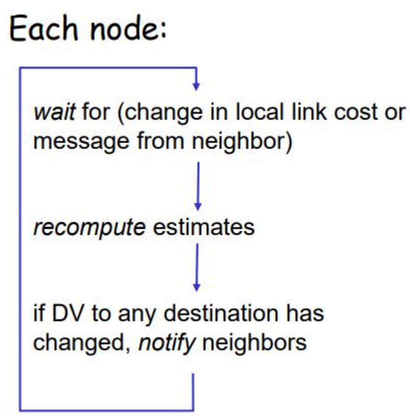
0.1.11 Bellman-Ford Algorithm

- Define distances at each node x
  - d_x(y) = cost of least-cost path from x to y
- Update distances based on neighbors
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors v



0.1.12 Distance Vector Algorithm

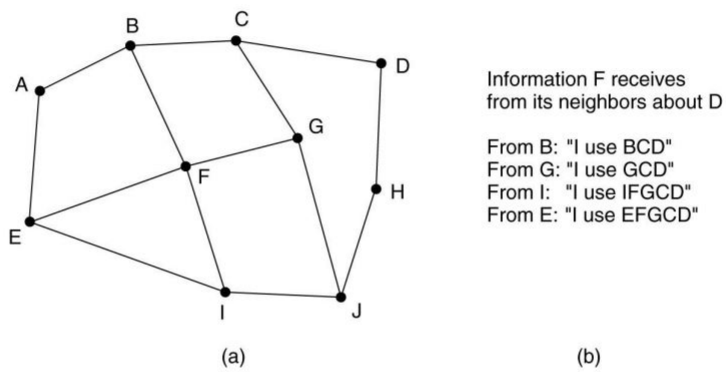
- c(x,v) = cost for direct link from x to v
  - node x maintains costs of direct links **c(x,v)**
- D_x(y) = estimate of least cost from x to y
  - node x maintains distance vector $\mathbf{D}_x = [\mathbf{D}_x(\mathbf{y}) \mathbf{:}\ \mathbf{y} \in \mathbf{N}\ ]$
- Node x maintains also its neighbors' distance vectors
  - for each neighbor v, x maintains $\mathbf{D}_v = [\mathbf{D}_v(\mathbf{y}) \mathbf{:}\ \mathbf{y} \in \mathbf{N}\ ]$
- Each node v periodically sends D_v to its neighbors
  - and neighbors update their own distance vectors
  - $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$ for each node $y \in N$
- Over time, the distance vector D_x converges
- Iterative, asynchronous, each local iteration caused by:
  - local link cost change
  - distance vector update message from neighbor
- Distributed
  - node notifies neighbors only when its DV changes
- Neighbors then notify their neighbors, if necessary



0.1.13 Routing Information Protocol (RIP)

- Distance vector protocol
  - nodes send distance vectors every 30 seconds
  - or when an update causes a change in routing
- RIP is limited to small networks

0.1.14 BGP – The Exterior Gateway Routing Protocol

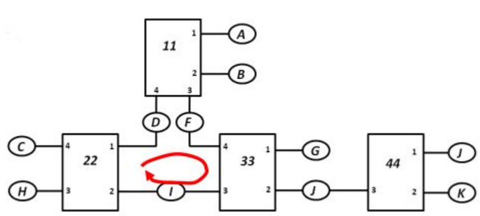


(a) A set of BGP routers. (b) Information sent to F

0.2 Unique Spanning Tree in Ethernet Networks

0.2.1 L2 Networking - Single Tree Required

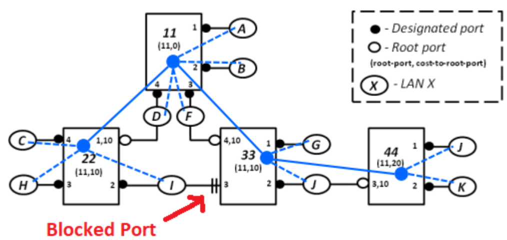
- Ethernet frame
  - No hop-count
  - Could loop forever
  - broadcast frame, mis-configuration



- Layer 2 network

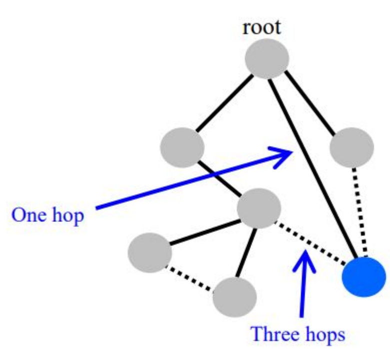
- **Required to have tree topology**
- Single path between every pair of stations

- Spanning Tree Protocol (STP)
  - Running in bridges
  - Helps building the spanning tree
  - Blocks ports



0.2.2 Constructing a Spanning Tree

- Distributed algorithm
  - switches need to elect a "root"
    * the switch with the smallest identifier
  - each switch identifies if its interface is on **the shortest path from the root**
  - messages(Y,d,X)
    * from node X
    * claiming Y is the root
    * and the distance is d



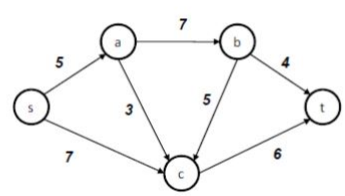
0.2.3 Steps in Spanning Tree Algorithm

- Initially, each switch thinks it is the root
  - switch sends a message out every interface
  - identifying itself as the root with distance 0
- Other switches update their view of the root
  - upon receiving a message, check the root id
  - if the new id is smaller, start viewing that switch as root
- Switches compute their distance from the root
  - add 1 to the distance received from a neighbor
  - identify interfaces not on a shortest path to the root and exclude them from the spanning tree

0.3 Maximum Flow of a Network

0.3.1 Flow Network Model

- **Flow network**
  - source s
  - sink t
  - nodes a, b and c
- Edges are labeled with **capacities (ex: bit/s)**



- Communication networks are not flow networks
  - they are queue networks
  - flow networks enable to determine limit values

0.3.2 Maximum Capacity of a Flow Network

- Max-flow min-cut theorem
  - maximum amount of flow transferable through a network
  - equals minimum value among all simple cuts of the network
- Cut → split of the nodes V into two disjoint sets S and T
  - $S \cup T = V$
  - there are $2^{|V|-2}$ possible cuts
- Capacity of cut (S,T):

$$c(S,T) = \sum_{(u,v)|u \in S, v \in T, (u,v) \in E} c(u,v)$$

  - (sum of the cost of all edges from S to T)