MIEEC

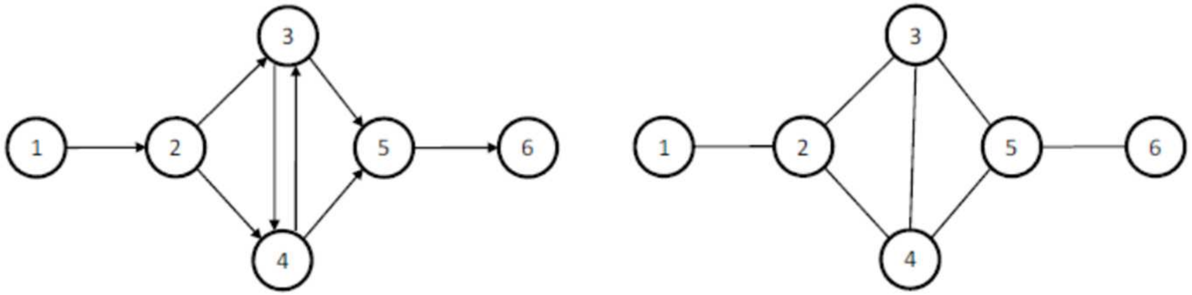
Computer Networks

Lecture note 9

Routing and shortest paths

Graphs and shortest paths

Graph notation

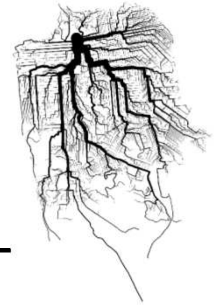


a) Directed graph b) Undirected graph

- G={V,E}
- V={v1,v2,v3,v4,v5,v6}
  - |V| = 6
- E={(v1,v2), (v2,v3), (v2,v4), (v3,v4), (v4,v3), (v3,v5), (v4,v5), (v5,v6)}
  - |E|=8

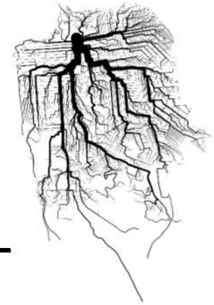
- G={V,E}
- V={v1,v2,v3,v4,v5,v6}
  - |V|=6
- E={(v1,v2), (v2,v3), (v2,v4), (v4,v3), (v3,v5), (v4,v5), (v5,v6)}
  - |E|=7

Tree

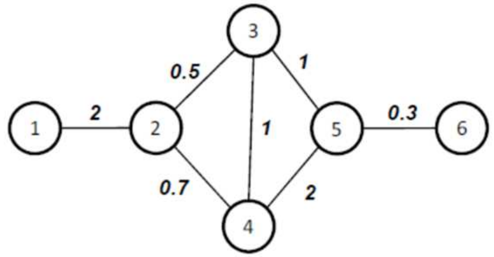


- Tree T = {V,E} is a graph
  - No cycles
  - |E|=|V|-1
  - Connected
    - a path exists between any pair of vertices
- Spanning tree
  - G={V,E}
  - T={V,E'} is a tree and
  - E' ⊆ E (is a subset of or equal to)

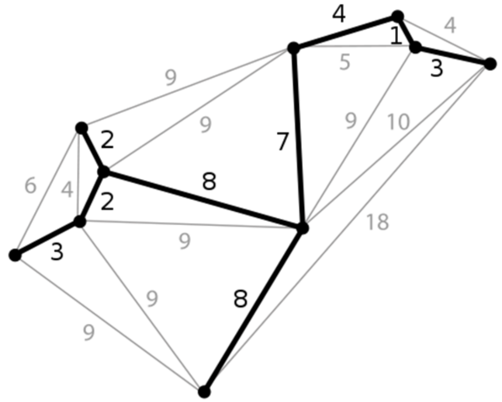
Shortest path tree



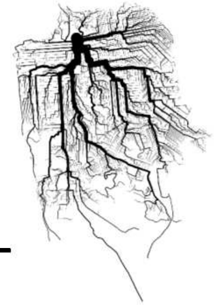
- Weight on graph edges
  - G={V,E,w}
  - T={V,E',w}
  - w: E → ℝ



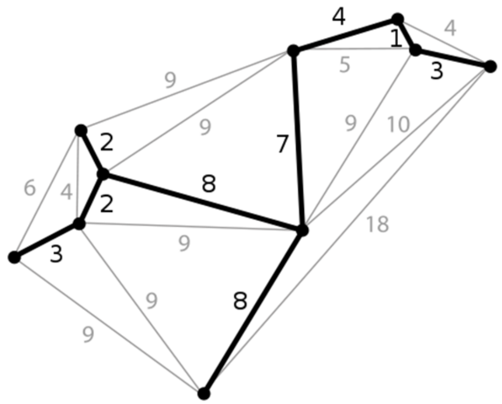
- Total cost of Tree T
  - $C_{total}(T) = \sum_{i=1}^{|E'|} w(e'_i)$
- Minimum spanning tree T*
  - $C_{total}(T^*) = \min_{T}(C_{total}(T))$
  - Single MST for each graph G
  - Prism, Kruskal



Shortest path tree

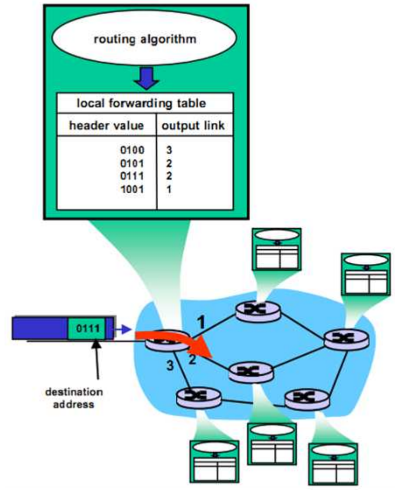


- Shortest Path Tree SPT at vertex s
  - Tree composed of
  - Union of shortest paths between s and the other vertices
  - Yes, it's a tree
  - Dijkstra, Bellman-Ford
- One SPT per vertex
  - Minimum spanning tree is not necessarily a SPT



Routing in Layer 3 networks

Forwarding vs. Routing



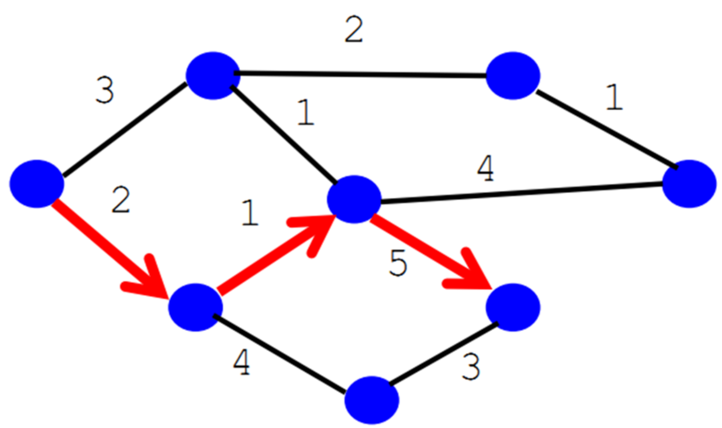
- Forwarding
  - Take packets from input ports
  - Send them to output ports
  - Which output port? Forwarding table
  - DATA PLANE
- Routing
  - Compute forwarding table entries
  - By computing the (best) path packets should take
  - Distributed
    - Routers compute best paths locally
    - Send messages to each other to update routing/link information
  - CONTROL PLANE

Importance of routing

- I.e. "Why do you want the best path?"
- End-to-end performance
  - Path affects QoS – delay, throughput, fairness
- Balanced used of network resources
  - Avoid congestion by choosing paths with less-loaded links
- Responsive routing
  - Compute paths on the fly
  - Responsive to link/router failures/maintenance
  - Smaller packet loss and delay

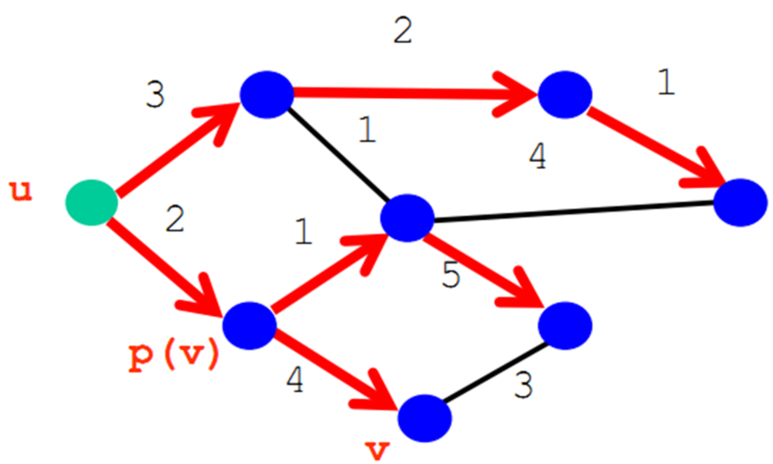
Shortest Path Routing

- Path selection model:
  - Destination-based
  - Load-insensitive (link weights are static)
  - Minimum hop count / sum of link weights



Shortest Path Problem

- Given network topology with link costs
  - c(x,y)
  - No link -> infinite cost
- Compute least-cost paths from source u to all other nodes
  - p(v) : predecessor to v in the path from u



Dijkstra’s Shortest Path Algorithm

- Iterative Algorithm
  - After k iterations know least-cost paths to k nodes
- S : set of nodes whose least-cost path is known
  - Initially S={u}, u is the source node
  - Add one node to S per iteration
- D(v) : current cost of path from source to node v
  - Initially
    - D(v) = c(u, v) for each node v adjacent to u
    - D(v) = ∞ , other nodes v
  - Update D(v) when shortest paths are learned

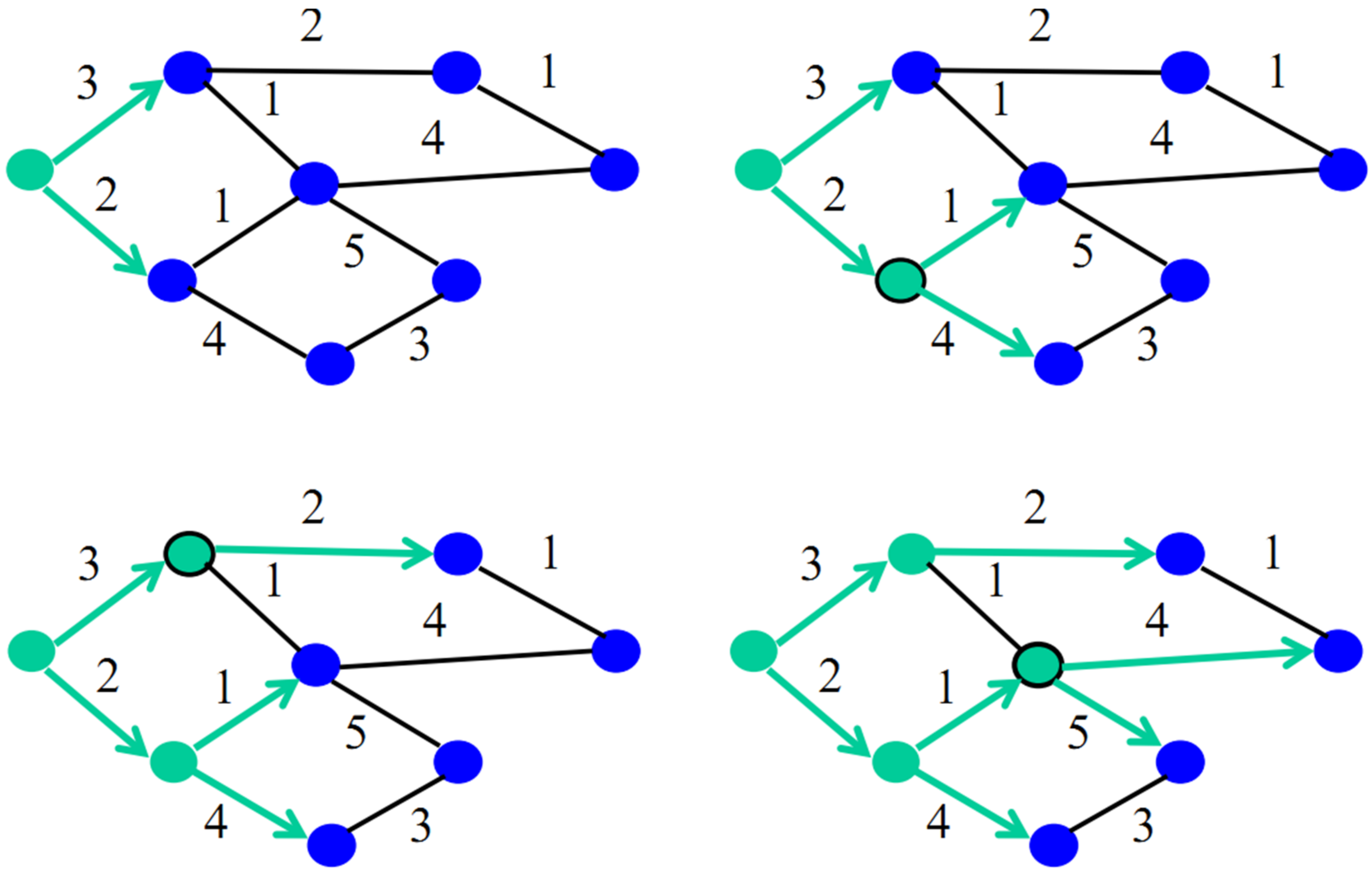
Dijkstra’s Shortest Path Algorithm

```
function Dijkstra(Graph, source):
    for each vertex v in Graph:             // Initializations
        dist[v] := infinity ;               // Unknown distance function from source to v
        previous[v] := undefined ;          // Previous node in optimal path from source
    end for ;
    dist[source] := 0 ;                     // Distance from source to source
    Q := the set of all nodes in Graph ;    // All nodes in the graph are unoptimized - thus are in Q
    while Q is not empty:                   // The main loop
        u := vertex in Q with smallest distance in dist[] ;
        if dist[u] = infinity:
            break ;                         // all remaining vertices are inaccessible from source
        end if ;
        remove u from Q ;
        for each neighbor v of u:           // where v has not yet been removed from Q.
            alt := dist[u] + dist_between(u, v) ;
            if alt < dist[v]:               // Relax (u,v,a)
                dist[v] := alt ;
                previous[v] := u ;
                decrease-key v in Q;        // Reorder v in the Queue
            end if ;
        end for ;
    end while ;
    return dist[] ;
end Dijkstra.
```

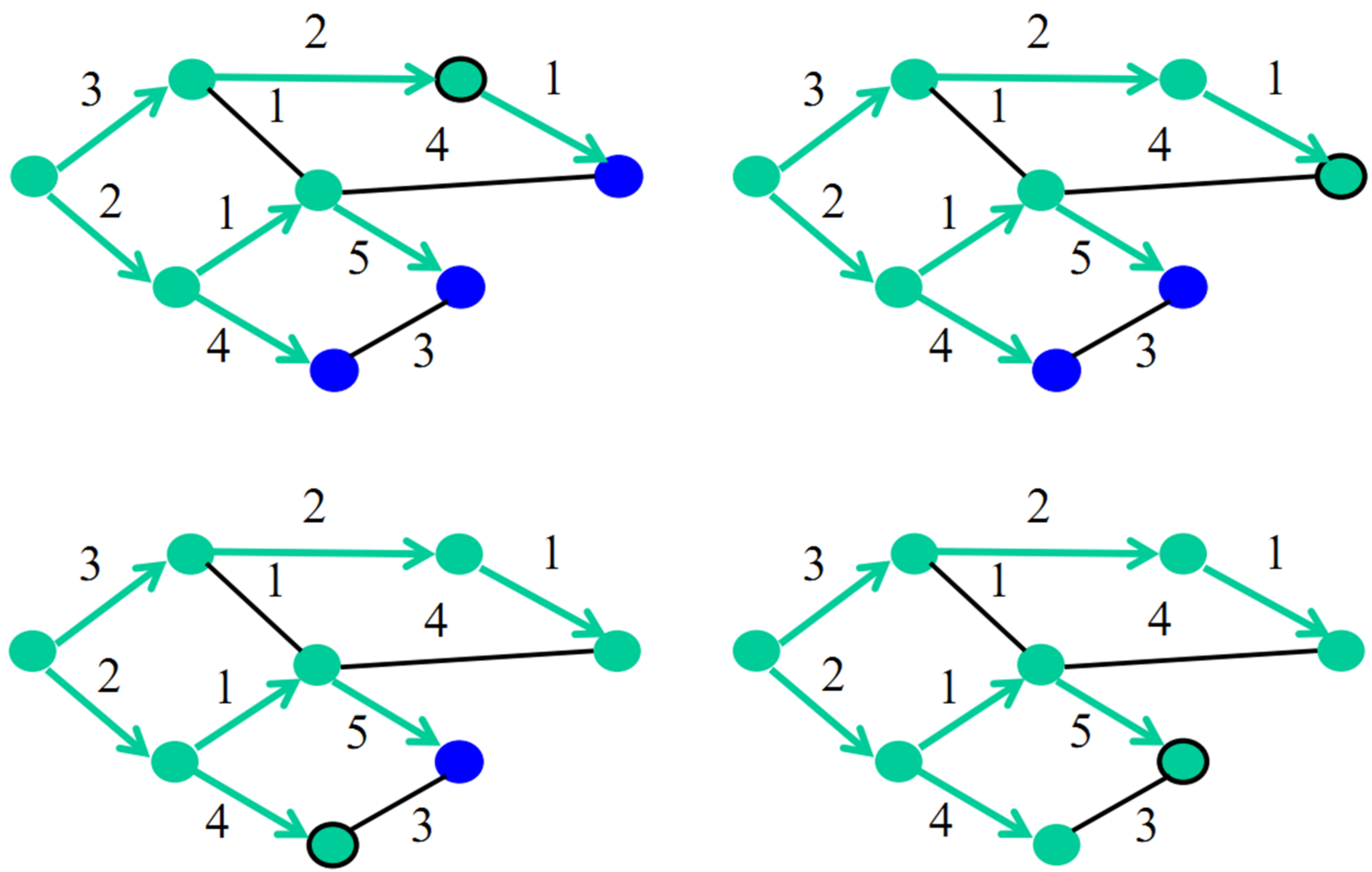
Retrieve path from source to target

```
S := empty sequence
u := target
while previous[u] is defined:
    insert u at the beginning of S
    u := previous[u]
end while ;
```

Dijkstra's example (1)

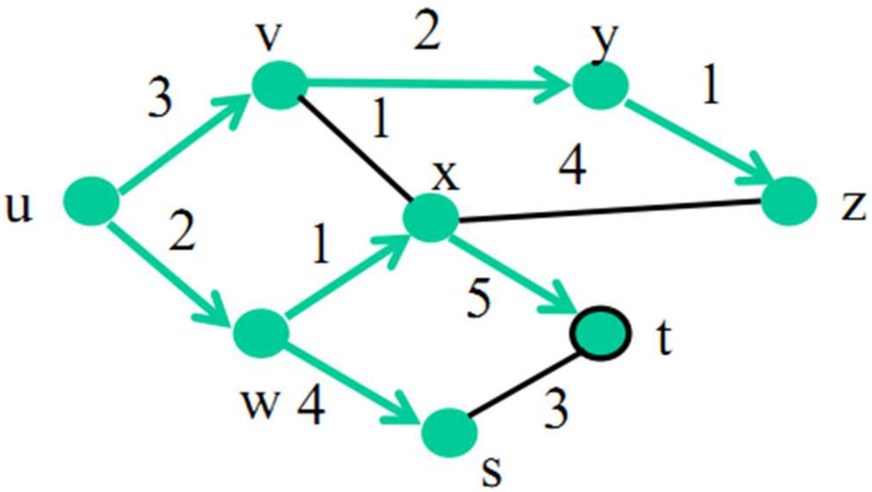


Dijkstra’s example (2)



Forwarding table at u

- Shortest Path Tree



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

TO THINK

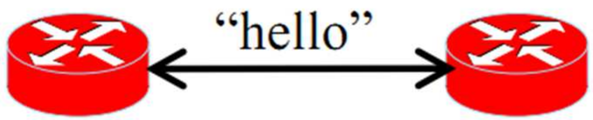
- How does router u know the link costs of non-adjacent routers?

Link state routing

- Each router keeps track of its incident links
  - Link up/down
  - Link cost
- Each router broadcasts the link state
  - Every router gets a full view of the network graph
- Each router runs Dijkstra's to
  - Compute shortest paths
  - Construct forwarding table
- Example protocols
  - OSPF, Open Shortest Path First
  - IS-IS, Intermediate System – Intermediate System

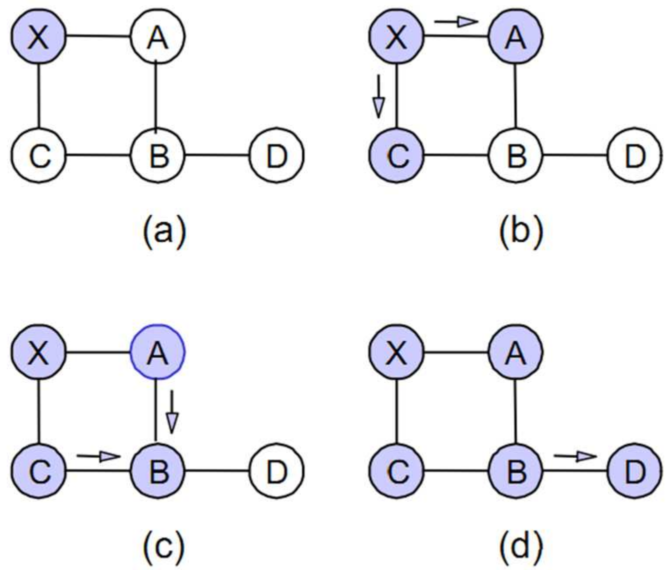
Detection of topology changes

- Beacons generated by routers on their links
  - Periodic “hello” messages on both directions
  - A few missed “hellos” => link down



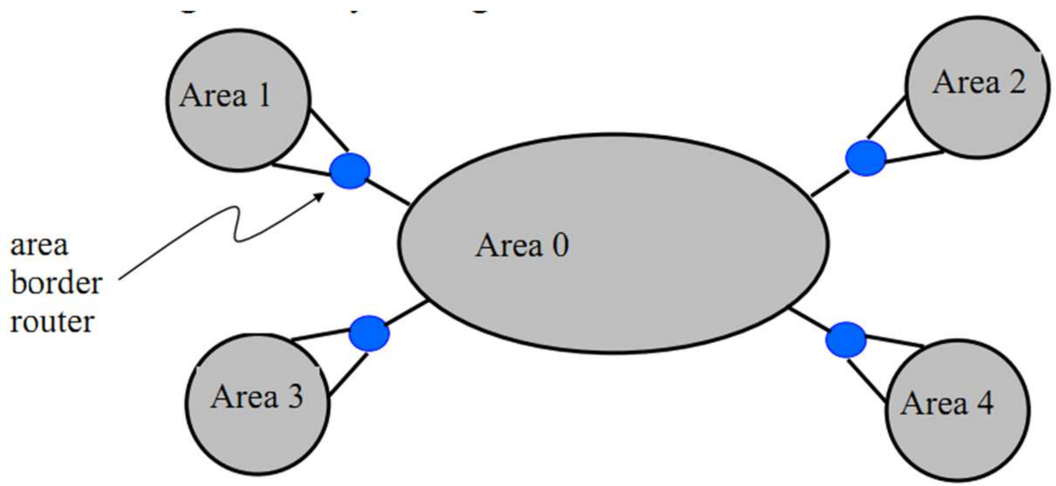
Broadcasting link state

- How to flood the link state?
  - Every node sends link-state information through their links
  - Next nodes forward link-state info on all links
    - except the one on which the info arrived
- When to start flooding?
  - Upon topology change
    - Link failure/recovery
    - Link cost change
  - Periodically
    - Refresh link-state information
    - typically 30 min



Scalability of link-state routing

- Overhead of link-state routing
  - Flooding link-state info on all the network
  - Running Dijkstra’s with all nodes
- OSPF “areas” introduce hierarchy
  - Split the network, the flooding, and routing computation

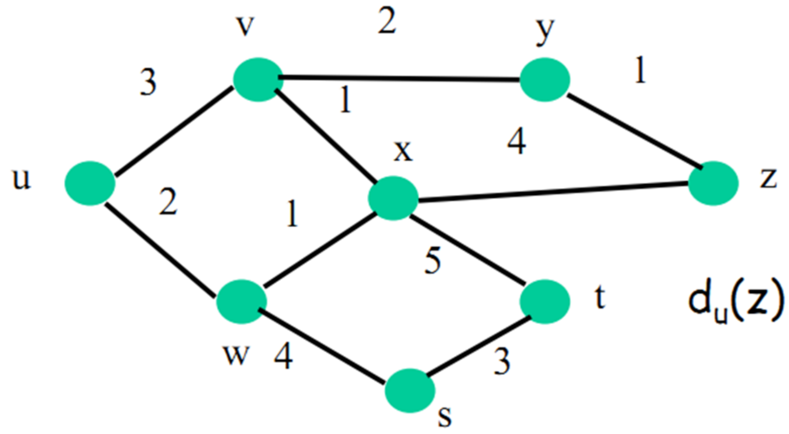


TO THINK

- Link-state/Dijkstra's requires knowledge of network topology at every router
  - c(x,y)
- Is it possible to compute shortest paths without knowing the whole topology of the network?
  - Is the distance to other nodes enough?

Bellman-Ford Algorithm

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



$d_u(z) = \min\{c(u,v) + d_v(z),$
$c(u,w) + d_w(z)\}$

Distance Vector Algorithm

- c(x,v) = cost for direct link (x,v)
  - Node x keeps costs of direct links on x
- $D_x(y)$ = estimate of least cost from x to y
  - Node x keeps vector $\mathbf{D}_\mathrm{x} = [D_x(y): y \in N]$
- Node x also keeps estimate of neighbors distance vectors
  - For each neighbor v, x keeps $\mathbf{D}_\mathrm{v} = [D_v(y): y \in N]$
- Each node v periodically sends $\mathbf{D}_\mathrm{v}$ to neighbors
  - And neighbors update their distance vectors
  - $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$ for each node $y \in N$
- Over iterations (time) vector $\mathbf{D}_\mathrm{x}$ converges

Distance Vector Algorithm

Each node:

wait for (change in local link cost or message from neighbor)

recompute estimates

if DV to any destination has changed, *notify* neighbors

- Iterative
  - Each local iteration is caused by
    - Local link cost change
    - DV update message from neighbor
- Distributed
  - Node notifies neighbors when DV changes
  - Neighbors notify neighbors, if necessary

DV Example (Initialization)

Table for A		
Dst	Cst	Hop
A	0	A
B	4	B
C	∞	–
D	∞	–
E	2	E
F	6	F

Table for B		
Dst	Cst	Hop
A	4	A
B	0	B
C	∞	–
D	3	D
E	∞	–
F	1	F

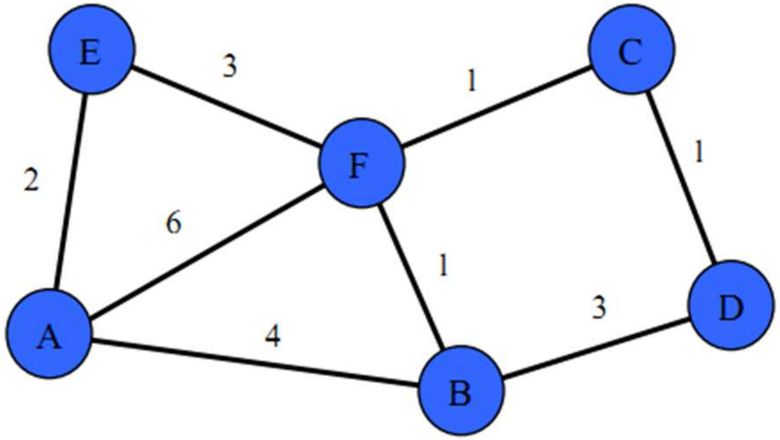


Table for C		
Dst	Cst	Hop
A	∞	–
B	∞	–
C	0	C
D	1	D
E	∞	–
F	1	F

Table for D		
Dst	Cst	Hop
A	∞	–
B	3	B
C	1	C
D	0	D
E	∞	–
F	∞	–

Table for E		
Dst	Cst	Hop
A	2	A
B	∞	–
C	∞	–
D	∞	–
E	0	E
F	3	F

Table for F		
Dst	Cst	Hop
A	6	A
B	1	B
C	1	C
D	∞	–
E	3	E
F	0	F

DV Example (Step 1)

Table for A

Dst	Cst	Hop
A	0	A
B	4	B
C	7	F
D	7	B
E	2	E
F	5	E

Table for B

Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

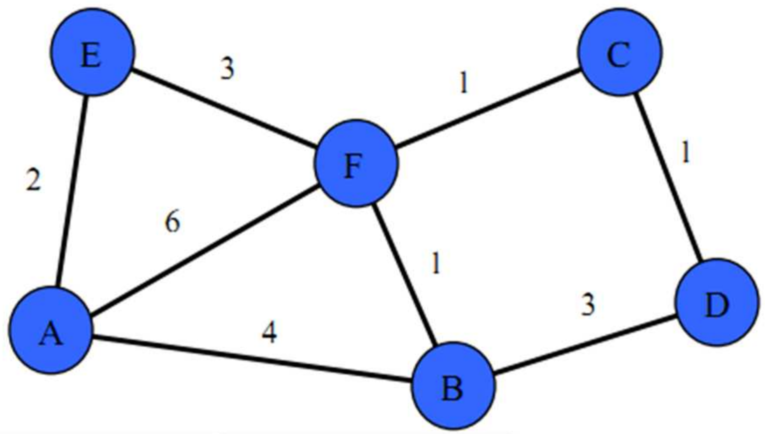


Table for C

Dst	Cst	Hop
A	7	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D

Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	∞	–
F	2	C

Table for E

Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	∞	–
E	0	E
F	3	F

Table for F

Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

DV Example (Step 2)

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	6	E	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

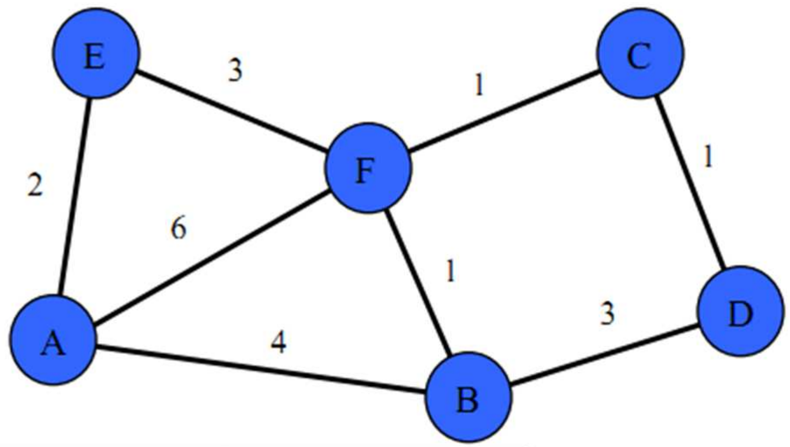


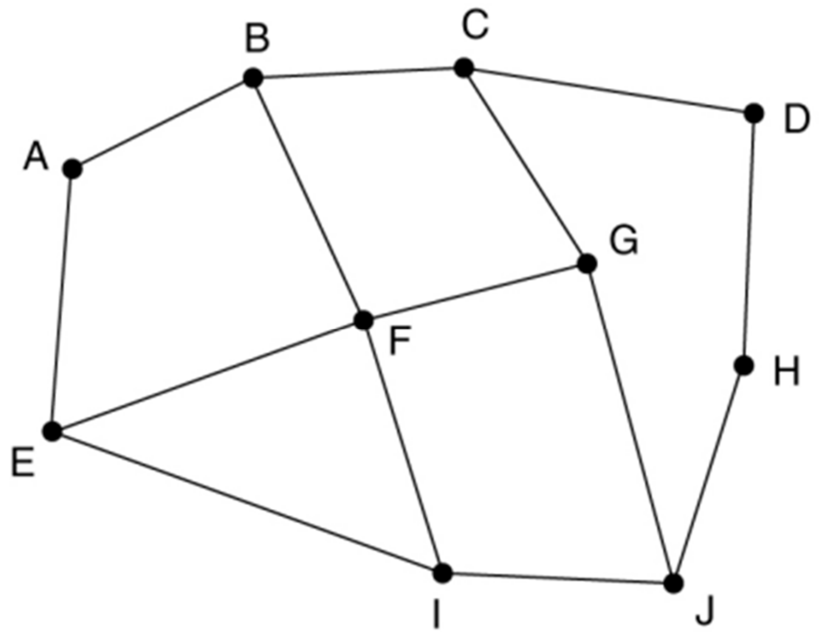
Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

RIP
Routing Information Protocol

- Distance Vector protocol
  - Nodes send distance vectors to neighbors every 30 seconds
  - Or when an update message causes change in routing
- RIP is limited to small networks

BGP
Exterior gateway routing protocol

- Path vector protocol
- Internet routing



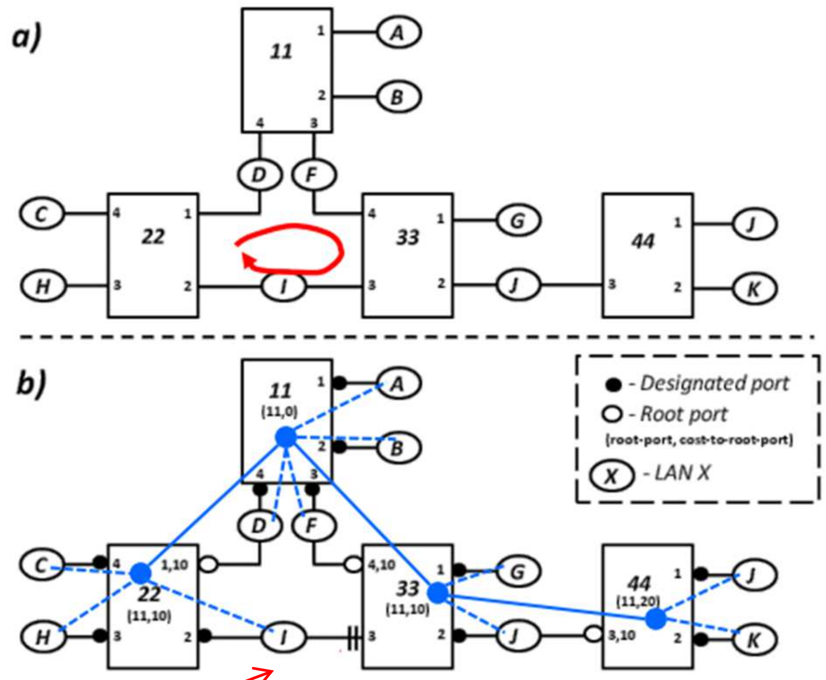
Information F receives
from its neighbors about D

From B: "I use BCD"
From G: "I use GCD"
From I: "I use IFGCD"
From E: "I use EFGCD"

Unique spanning trees in Ethernet networks

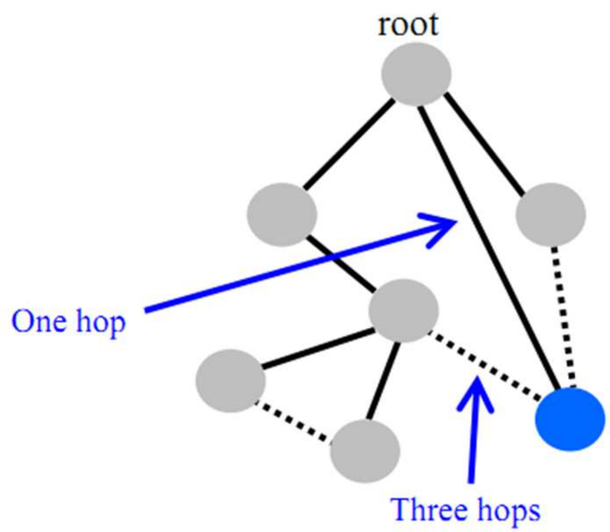
Networks of Layer 2 switches

- Ethernet frame
  - No hop-count
  - Could loop forever
  - Broadcast frame, misconfiguration
- Layer 2 network
  - Requires tree topology
  - Single path between any station pair
- Spanning Tree Protocol
  - Runs in bridges
  - Helps building the spanning tree
  - Blocks ports



Constructing a Spanning Tree

- Distributed algorithm
  - Switches need to elect a “root” for the tree
    - The switch with the smallest identifier
  - Each switch finds out if its interface is on the shortest path from the root
  - Messages (Y,d,X)
    - From node X
    - Claiming node Y is the root
    - And the distance is d

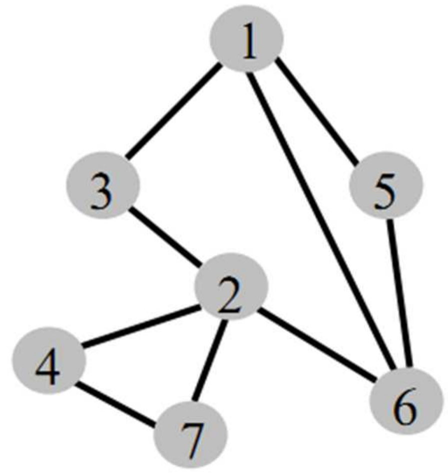


Steps in ST algorithm

- Initially each switch thinks it's the root
  - Sends message out on every interface
  - Identifies itself as root
  - Switch X announces (X,0,X)
- Other switches update their view of the root
  - Upon receiving a message, check 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 neighbor
  - Identify interfaces not in the shortest path to root
  - Exclude these interfaces from the spanning tree

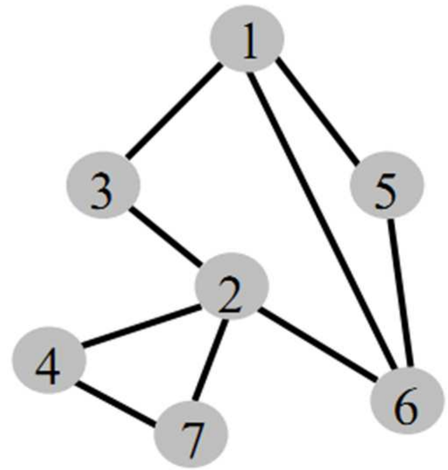
Example - Switch #4 viewpoint

- Switch #4 thinks it's the root
  - Sends (4,0,4) to 2 and 7
- Then switch #4 hears from #2
  - Receives (2,0,2)
  - Thinks #2 is the root
  - Realizes it's 1 hop from the root
- Switch #4 hears from #7
  - Receives (2,1,7) from #7
  - Realizes this is a longer path
  - Prefers its 1 hop path to #2
  - Removes 4-7 link from tree



Example - Switch #4 viewpoint

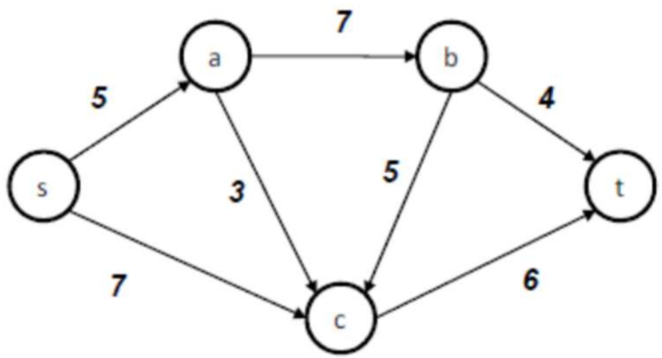
- Switch #2 hears about #1
  - Hears (1,1,3) from #3
  - Starts treating #1 as root
  - Sends (1,2,2) to neighbors
- Then switch #4 hears from #2
  - Starts treating #1 as root
  - Sends (1,3,4) to neighbors
- Switch #4 hears from #7
  - Receives (1,3,7) from #7
  - Realizes this is a longer path
  - Prefers its 3 hop path to #1
  - Removes 4-7 link from tree



Maximum flow in a network

Flow network model

- Flow network
  - Source s
  - Sink t
  - Nodes a,b,c



- Edges labeled with capacities
  - (e.g. bit/s)
- Communication networks are not flow networks
  - They are queue networks
  - Flow network approach used to find limit values

Maximum capacity of a flow network

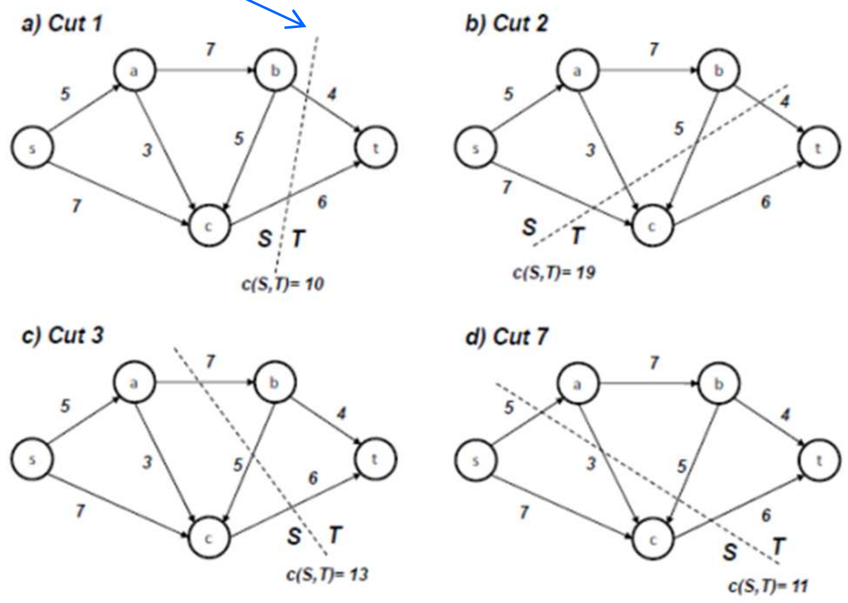
- Max-flow min-cut theorem
  - Maximum amount of flow transferrable through a network
  - Equals minimum value of all simple cuts in the network
- Cut
  - Split of nodes V in G={V,E} into disjoints sets S and T
  - $S \cup T = V$
  - There are $2^{|V|-2}$ possible cuts
- Capacity of cut (S,T)
  - $c(S,T) = \sum_{(u,v)|u \in S, v \in T, (u,v) \in E} c(u,v)$

Max-flow Min-cut example

Max-flow = 10

$$2^{|5|-2} = 8 \text{ possible cuts}$$

Cut	Vertices					$c(S,T)$	Feasability
	s	a	b	c	t		
1	S	S	S	S	T	10	✓
2	S	S	S	T	T	19	✓
3	S	S	T	S	T	13	✓
4	S	S	T	T	T	17	✓
5	S	T	S	S	T	-	×
6	S	T	S	T	T	-	×
7	S	T	T	S	T	11	✓
8	S	T	T	T	T	12	✓



- What is a graph?
- What is a spanning tree?
- What is a shortest path tree?
- How are paths defined in a network?
- How does Dijkstra's algorithm work?
- How does a link-state algorithm work?
- How do nodes find out about their neighbors?
- How does the Bellman-Ford algorithm work?
- How does a Distance-Vector algorithm work?
- What are the limitations of a layer 2 network of switches?
- How does the IEEE Spanning Tree protocol work?
- What is the maximum capacity of a flow network?

HOMEWORK

- Review slides
- Read from Tanenbaum
  - Section 5.2 – Routing Algorithms
  - Section 4.7.3 – Spanning Tree Bridges
- Do your Moodle homework