CEN445 – Network Protocols and Algorithms

Chapter 2

Routing Algorithms

Dr. Ridha Ouni
Department of Computer Engineering
College of Computer and Information Sciences
King Saud University

References

Some slides are inspired from the following references

- M. Dahshan, notes CEN445 – Network protocols and algorithms, Dep. of Comp. Eng. CCIS, KSU.
- S. Nasri, notes CSC345 – Intro. to comp. networks, Dep. of Comp. Eng. College of Computer, QU.
- R. Ouni, notes CEN531 – Computer Networks, Dep. of Comp. Eng. CCIS, KSU.

Outline

- Routing algorithms – Introduction
- Optimality principle
- Non-Adaptive Routing Algorithms
  - Dijkstra's Algorithm
  - Flooding
- Adaptive Routing Algorithms
  - Distance Vector Routing
  - Link State Routing
- Hierarchical Routing

Routing Algorithms

- Routing – main function of network layer
- Routing algorithm
  - decides which output line incoming packet should be transmitted on
  - fills up and updates routing tables
- Forwarding
  - look up the routing tables and put the packet in the appropriate output line

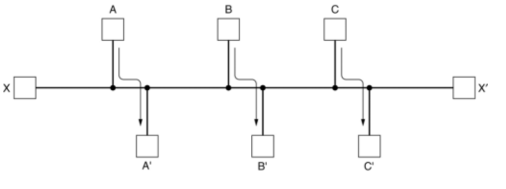
Routing Algorithms *(cont'd)*

- In connection-oriented service, the routing algorithm is performed only during connection setup
- In connectionless service, the routing algorithm is performed as each packet arrives

Desired Properties

- Correctness
- Simplicity
- Robustness: ability to handle failures
- Stability: converge to equilibrium
- Fairness
- Optimality

} Possible Conflict

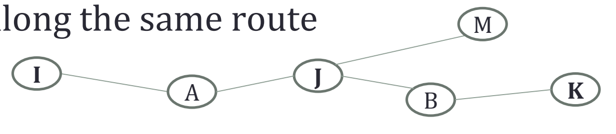


Types of routing algorithms

- Non-adaptive/static routing
  - routing decisions not based on traffic, topology, current state of the network.
  - routes are computed in advance
- Adaptive routing
  - Change their decisions to reflect changes in the topology and traffic
  - Differ in: information source, update frequency and optimization metrics
- Hierarchical Routing is used to make these algorithms scale to large networks

Optimality Principle

- 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

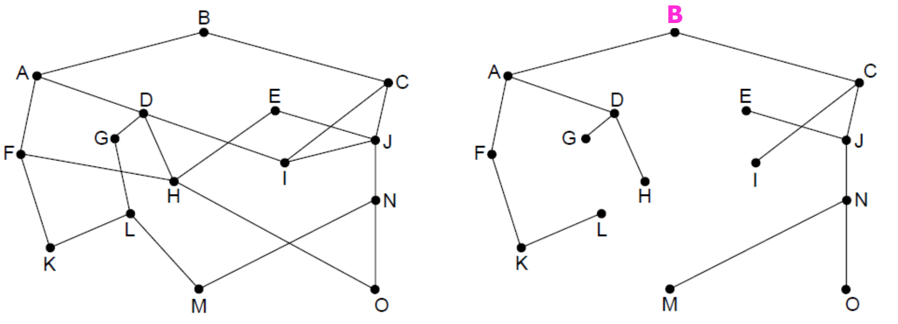


- Set of optimal routes from all sources to a given destination form a tree rooted at the destination "sink tree"
- Goal of all routing algorithms: discover and use sink tree for all routers

The Optimality Principle

Each portion of a best path is also a best path; the union of them to a router is a tree called the sink tree

- Best – means fewest hops in the example



Network

Sink tree of best paths to router B

Non-Adaptive Routing Algorithms

- Examples:
  - Shortest Path Routing
  - Flooding

Shortest Path Routing

- Build a graph of network
- Each node represent a router
- Each arc represent a link
- Find shortest path between the two nodes

Shortest Path Routing

- For a pair of communicating hosts, there is a shortest path between them
- Shortness may be defined by:
  - number of hops
  - geographic distance
  - mean queuing/transmission delay
  - bandwidth
  - cost

Dijkstra's Algorithm

- Finds shortest paths from given source node **S** to all other nodes
- Starts from the source node and finds the nearest adjacent node
- Runs in stages, each time adding node with next shortest path
- algorithm terminates when all nodes are processed by algorithm (in set *T*)

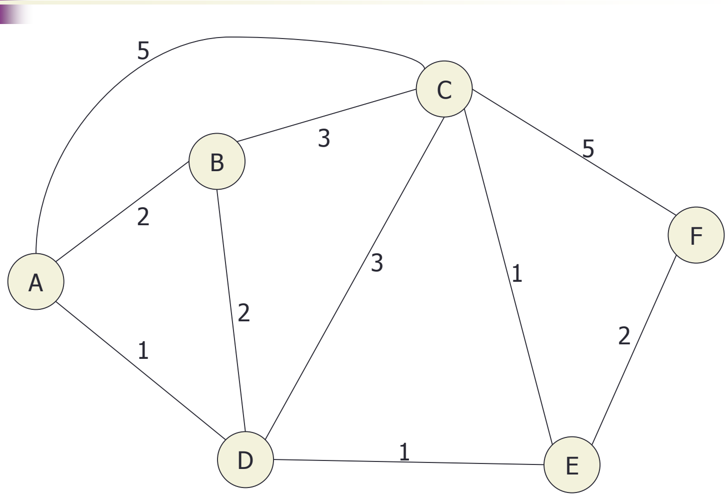
Dijkstra's Algorithm

- Step 1 [Initialization]
  - $T = \{s\}$ Set of nodes so far incorporated
  - $L(n) = w(s, n)$ for $n \neq s$
  - initial path costs to neighboring nodes are simply link costs

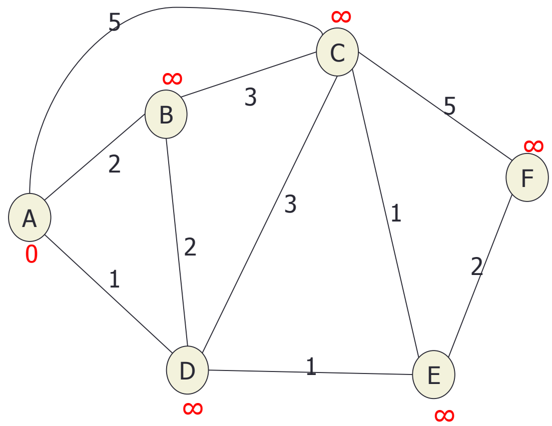
Dijkstra's Algorithm

- Step 2 [Get Next Node]
  - find neighboring node not in T with least-cost path from s
  - incorporate node x into T *(node marked as permanent)*
  - also incorporate the edge that is incident on that node and a node in T that contributes to the path
- Step 3 [Update Least-Cost Paths]
  - $L(n) = \min[L(n), L(x) + w(x, n)]$ for all $n \notin T$
  - if latter term is minimum, path from s to n is path from s to x concatenated with edge from x to n

Dijkstra's Algorithm



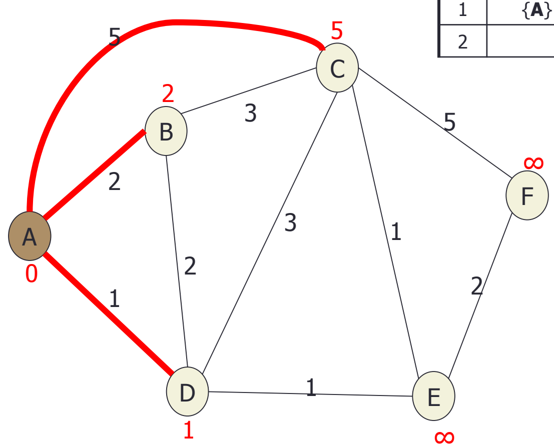
Dijkstra's Algorithm



T={}

Dijkstra's Algorithm

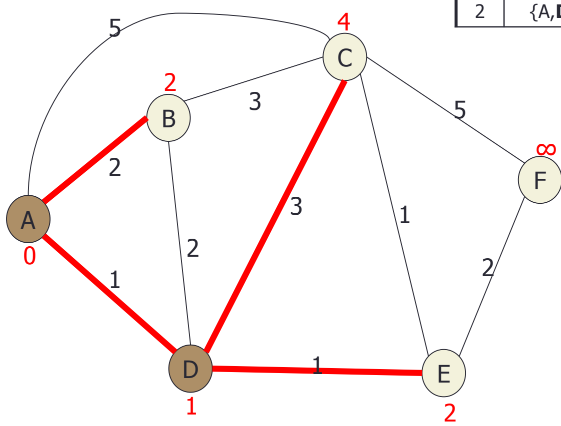
#	T	B	C	D	E	F
1	{**A**}	2, A	5, A	1, A	∞, -	∞, -
2						



T={A}

Dijkstra's Algorithm

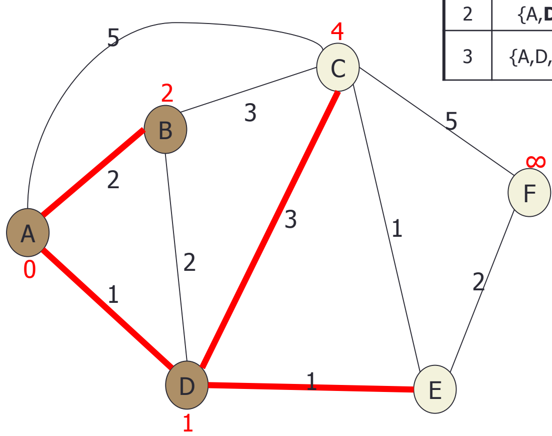
#	T	B	C	D	E	F
1	{**A**}	2, A	5, A	1, A	∞, -	∞, -
2	{A,**D**}	2, A	4, D	-	2, D	∞, -



T={A,D}

Dijkstra's Algorithm

#	T	B	C	D	E	F
1	{**A**}	2, A	5, A	1, A	∞, -	∞, -
2	{A,**D**}	2, A	4, D	-	2, D	∞, -
3	{A,D,**B**}	-	4, D	-	2, D	∞, -

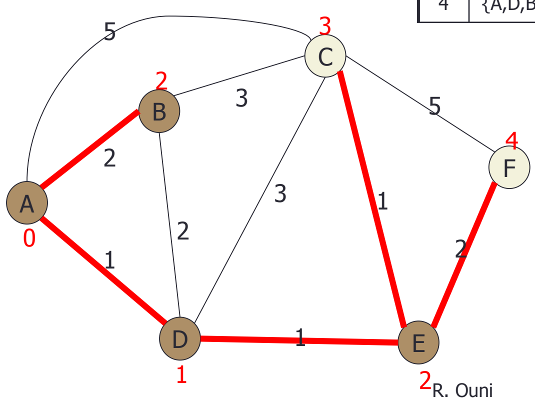


T={A,D,B}

2

Dijkstra's Algorithm

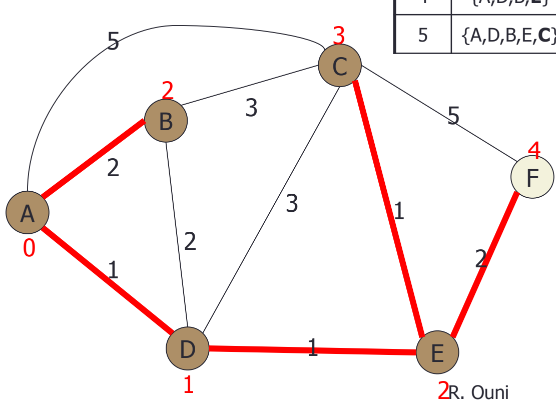
#	T	B	C	D	E	F
1	{**A**}	2, A	5, A	1, A	∞, -	∞, -
2	{A,**D**}	2, A	4, D	-	2, D	∞, -
3	{A,D,**B**}	-	4, D	-	2, D	∞, -
4	{A,D,B,**E**}	-	3, E	-	-	4, E



T={A,D,B,E}

Dijkstra's Algorithm

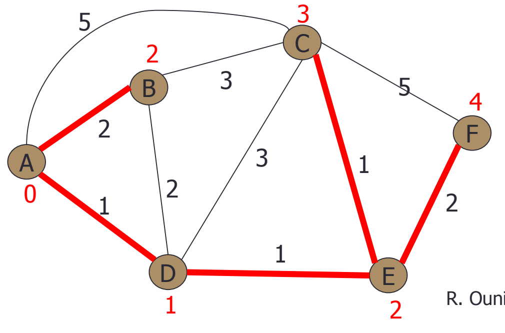
#	T	B	C	D	E	F
1	{**A**}	2, A	5, A	1, A	∞, -	∞, -
2	{A,**D**}	2, A	4, D	-	2, D	∞, -
3	{A,D,**B**}	-	4, D	-	2, D	∞, -
4	{A,D,B,**E**}	-	3, E	-	-	4, E
5	{A,D,B,E,**C**}	-	-	-	-	4, E



T={A,D,B,E,C}

Dijkstra's Algorithm

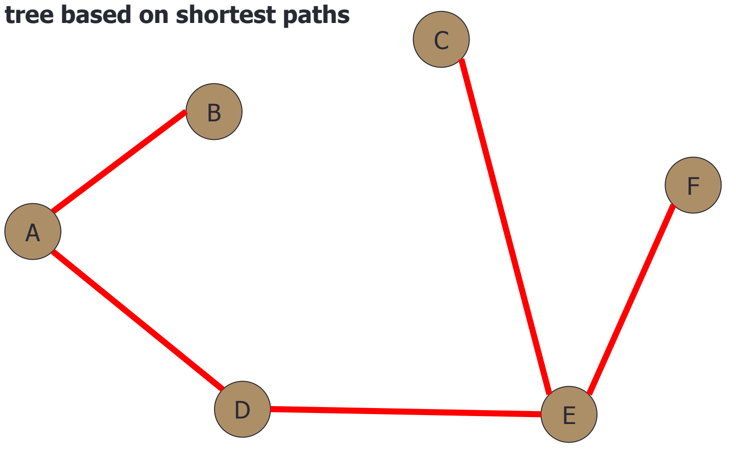
#	T	B	C	D	E	F
1	{**A**}	2, A	5, A	1, A	∞, -	∞, -
2	{A,**D**}	2, A	4, D	-	2, D	∞, -
3	{A,D,**B**}	-	4, D	-	2, D	∞, -
4	{A,D,B,**E**}	-	3, E	-	-	4, E
5	{A,D,B,E,**C**}	-	-	-	-	4, E
6	{A,D,B,E,C,**F**}	-	-	-	-	-



T={A,D,B,E,C,F}

Dijkstra's Algorithm

Sink tree based on shortest paths

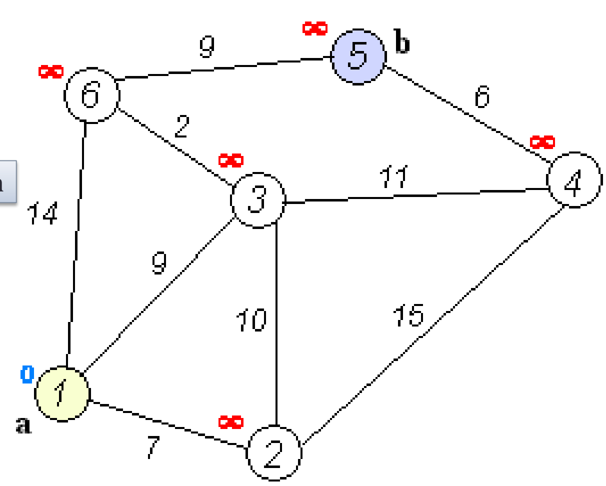


Dijkstra's Algorithm

#	T	B	C	D	E	F
1	{**A**}	2, A	5, A	1, A	∞, -	∞, -
2	{A,**D**}	2, A	4, D	-	2, D	∞, -
3	{A,D,**B**}	-	4, D	-	2, D	∞, -
4	{A,D,B,**E**}	-	3, E	-	-	4, E
5	{A,D,B,E,**C**}	-	-	-	-	4, E
6	{A,D,B,E,C,**F**}	-	-	-	-	-

Dijkstra's Algorithm

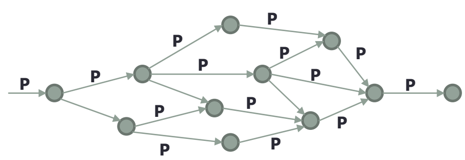
Dijktra Animation



Source: Wikipedia
http://en.wikipedia.org/wiki/File:Dijkstra_Animation.gif

Flooding

- Send every packet to all lines except the one it arrived on
- Large number of duplicate packets



Reducing Flooding Algorithm's Limited area

- Solution 1
  - Have a hop counter in the packet header
  - IMPs decrement each arriving packet's hop counter
  - IMPs discard a packet with hop count=0
  - Ideally, the hop counter should be initialized to the length of the path from the source to the destination

Reducing Flooding Algorithm's Duplicate Packets *(cont'd)*

- Solution 2
  - Require the first IMP hop to put a sequence number in each packet it receives from its hosts
  - Each IMP maintains a table listing the sequence numbers it has seen from each first-hop IMP. The IMP can then discard packets it has already seen.

Flooding: Possible Applications

- Military Applications
  - Large number of IMPs is desirable
  - If one IMP is taken out (by a bomb?) flooding will still get packets to their destinations
- Distributed Databases
  - Simultaneous updates of multiple databases can be done with a single packet transmission
- Wireless Networks
  - Inherently broadcasting/flooding

Adaptive Routing Algorithms

- Problems with non-adaptive algorithms
  - If traffic levels in different parts of the subnet change dramatically and often, nonadaptive routing algorithms are unable to cope with these changes
  - Lots of computer traffic is bursty, but nonadaptive routing algorithms are usually based on average traffic conditions
- Adaptive routing algorithms can deal with these situations

Adaptive Routing Algorithms

- Each IMP periodically exchanges routing information (e.g., estimated time delay, queue length, etc.) with its neighbors
- Examples:
  - Distance Vector Routing
    - original ARPA net routing scheme, often called RIP (route information protocol)
  - Link State Routing
    - base for the current Internet routing algorithm

Distance Vector Routing

- Each router maintains a table containing
  - destination
  - best known distance to that destination
  - line to use to get there
- Uses Bellman-Ford algorithm
- Used in ARPANET and now used in RIP
- Distance can be any metric: delay, hop count, queue length, etc.

Distance Vector Routing

- Each router exchange with its neighbors list of delays to each destination
- Router X estimates delay to router Z
  - Router Y is a neighbor to router X
  - D(X,Z) =D(X,Y)+ D(Y,Z)

Bellman-Ford Algorithm

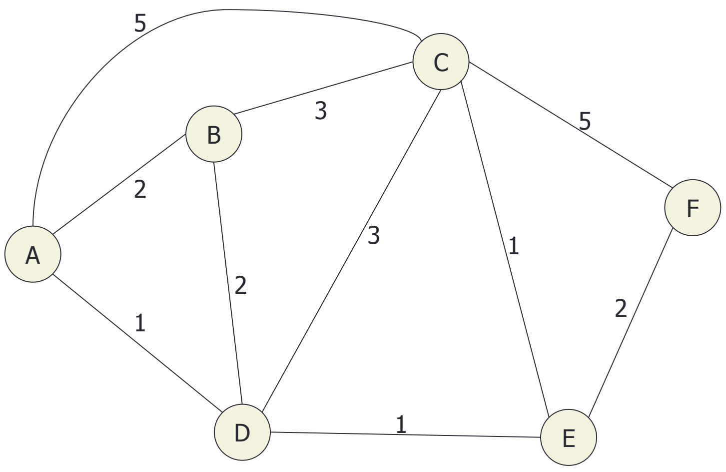
- Find shortest paths from given node subject to constraint that paths contain at most one link
- Find the shortest paths with a constraint of paths of at most two links
- and so on

Bellman-Ford Algorithm

- Step 1 [Initialization]
  - $L_0(n) = \infty$, for all $n \neq s$
  - $L_h(s) = 0$, for all h
- Step 2 [Update]
  - For each successive $h \geq 0$
    - For each $n \neq s$, compute: $L_{h+1}(n) = \min(\text{for each } j)\ [L_h(j)+w(j,n)]$
  - connect n with predecessor node j that gives min
  - eliminate any connection of n with different predecessor node formed during an earlier iteration
  - path from s to n terminates with link from j to n

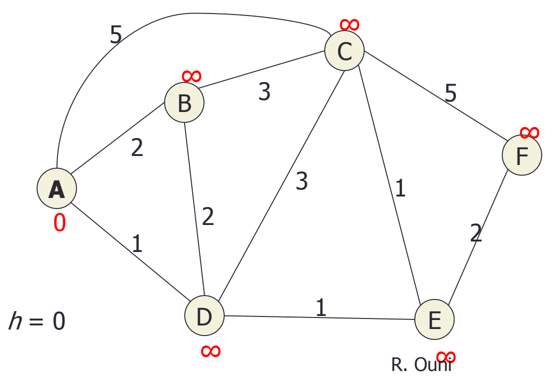
- n: node number
- s: source node
- h: hop count
- $L_h(n)$: least cost to n with no more than h hops

Bellman-Ford Algorithm



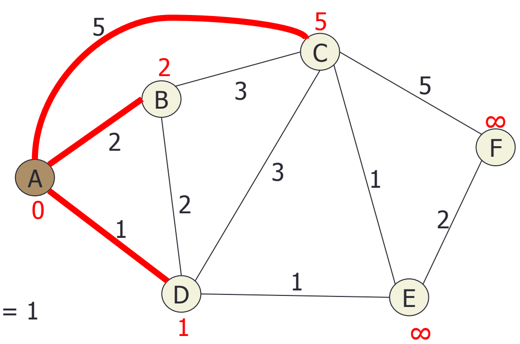
Bellman-Ford Algorithm

h	B	C	D	E	F
0	∞, -	∞, -	∞, -	∞, -	∞, -



Bellman-Ford Algorithm

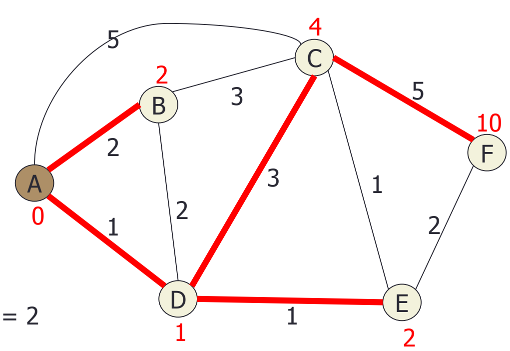
h	B	C	D	E	F
0	∞, -	∞, -	∞, -	∞, -	∞, -
1	2, A	5, A	1, A	∞, -	∞, -



h = 1

Bellman-Ford Algorithm

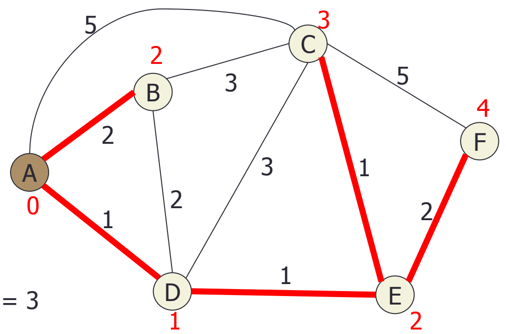
h	B	C	D	E	F
0	∞, -	∞, -	∞, -	∞, -	∞, -
1	2, A	5, A	1, A	∞, -	∞, -
2	2, A	4, D	1, A	2, D	10, C



h = 2

Bellman-Ford Algorithm

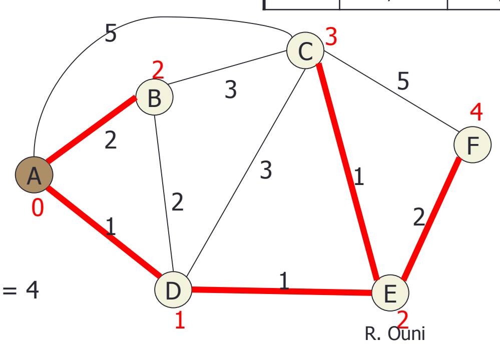
h	B	C	D	E	F
0	∞, -	∞, -	∞, -	∞, -	∞, -
1	2, A	5, A	1, A	∞, -	∞, -
2	2, A	4, D	1, A	2, D	10, C
3	2, A	3, E	1, A	2, D	4, E



h = 3

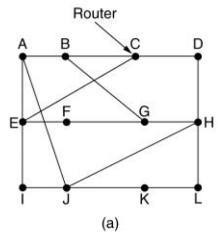
Bellman-Ford Algorithm

h	B	C	D	E	F
0	∞, -	∞, -	∞, -	∞, -	∞, -
1	2, A	5, A	1, A	∞, -	∞, -
2	2, A	4, D	1, A	2, D	10, C
3	2, A	3, E	1, A	2, D	4, E
4	2, A	3, E	1, A	2, D	4, E



h = 4

Distance Vector Routing



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

(b)

(a) A subnet (b) Input from A, I, H, K, and the new routing table for J

The Count-to-Infinity Problem

Failures can cause DV to "count to infinity" while seeking a path to an unreachable node

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

Good news. a new path to *A* spreads quickly

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
	⋮				
	•	•	•	•	

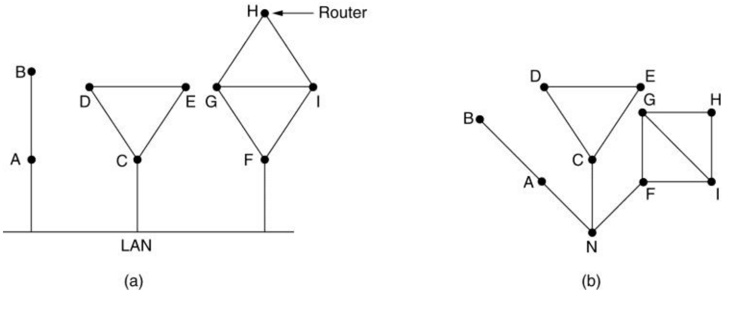
etc… to infinity; bad news travels slow

Link State Routing

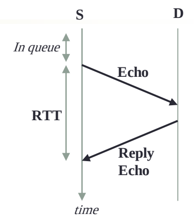
- Discover neighbors and learn their network addresses
- Each router measures the distance (in delay, hop count, etc.) between itself and its adjacent routers
- The router builds a packet containing all these distances. The packet also contains a sequence number and an age field.
- Send this packet to all other routers
- Once a router receives all the link state packets from the network, it can reconstruct the complete topology and compute a shortest path between itself and any other node using Dijsktra's algorithm.

Learning about Neighbors

- Send HELLO packet on point-to-point lines
- If routers are connected to a LAN, the LAN can be represented as a node

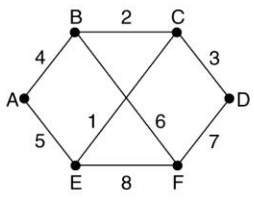


Measuring Line Cost



- Send ECHO packet
- Wait for response
- Measure round-trip-time
- To take load into account: start timer when packet is queued
- To ignore the load: start timer when packet reaches the front of the queue

Building Link State Packets



(a)

Link State Packets

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
E	8

(b)

Distributing Link State Packets

- Use flooding
- Packet contains sequence number
- When packet is received
  - If new, forward to all except coming from
  - If duplicate, discard
  - If old, rejected

Distributing Link State Packets

Problems

- Sequence number wrap around
  - Use 32-bit sequence numbers
- Router crashes, seq. no. starts over
- Seq. no. corrupted: 65540 instead of 4
  - Include age, decremented once per second

Hierarchical Routing

Our routing study thus far - idealization
- all routers identical
- network "flat"

➔ not true in practice

scale: with 200 million destinations:
- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network

R. Ouni 51

Hierarchical Routing

- Routing tables grow with network size
- More router memory
- More CPU time to scan them
- More bandwidth to send updates
- For large networks, better to do routing hierarchically
- Hierarchy can be in multiple levels
  - regions
  - clusters
  - zones
  - groups ...

R. Ouni 52

Hierarchical Routing

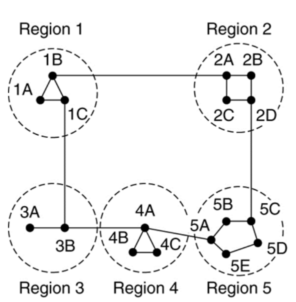
Problems can be solved:

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol (LS, DV)
  - Within AS: "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router

- To connect different ASs: edge routers add task for being responsible for routing outside AS ➔ inter-autonomous system routing protocol.

Hierarchical Routing



Full routing table has 17 entries

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

(b)

Hierarchical table for 1A

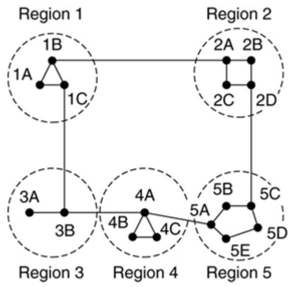
Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Hierarchical routing table has 7 entries

Hierarchical Routing

- The gain in space is not free
- Increased path length for some hosts
- Example
  - best route from 1A to 5C is via R2
  - with hierarchical routing all traffic to R5 is via R3
  - because it is better for most dests in R5

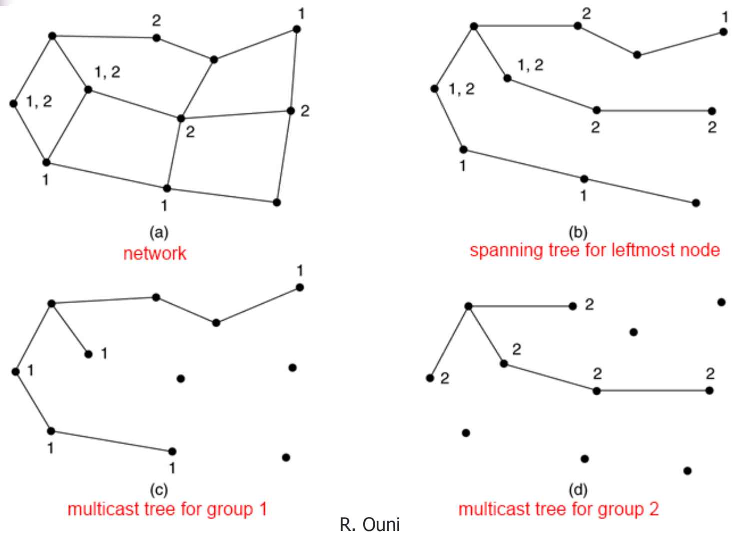


Homework Problem: Prove! ☺

Multicast Routing

- Multicasting
  - sending message to a group of nodes
  - routing algorithm called multicast routing
- Why multicasting?
  - distributed processing
  - broadcasting is inefficient, sometimes insecure
- Require group management
  - create, destroy groups
  - processes to join, leave groups

Multicast Routing



External References

- Data and Computer Communications, Stallings, 8/E
  - Dijkstra and Bellman-Ford algorithm descriptions and examples