CISC 7334X EW8
Distance Vector and Link State Routing

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

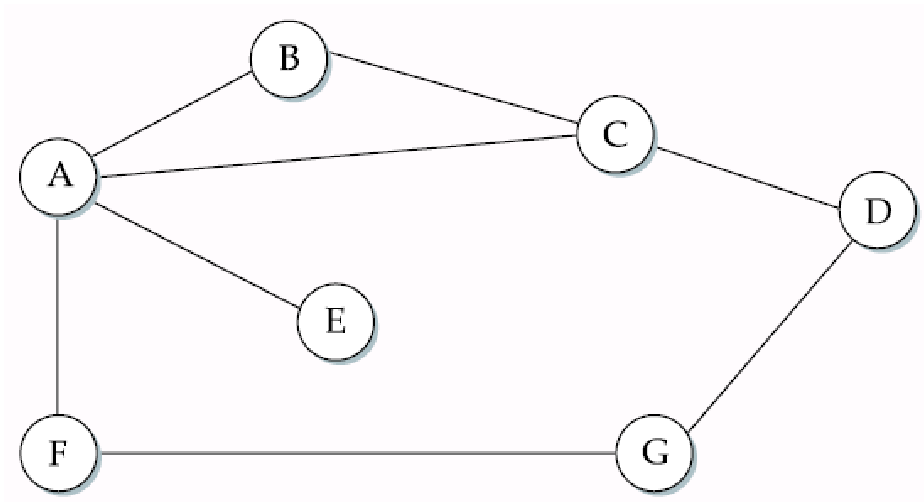
Acknowledgements

- Some pictures used in this presentation were obtained from the Internet
- The instructor used the following references
  - Larry L. Peterson and Bruce S. Davie, Computer Networks: A Systems Approach, 5th Edition, Elsevier, 2011
  - Andrew S. Tanenbaum, Computer Networks, 5th Edition, Prentice-Hall, 2010

Distance Vector

- Each node constructs a one dimensional array (a vector) containing the “distances” (costs) to all other nodes and distributes that vector to its immediate neighbors
- Starting assumption is that each node knows the cost of the link to each of its directly connected neighbors

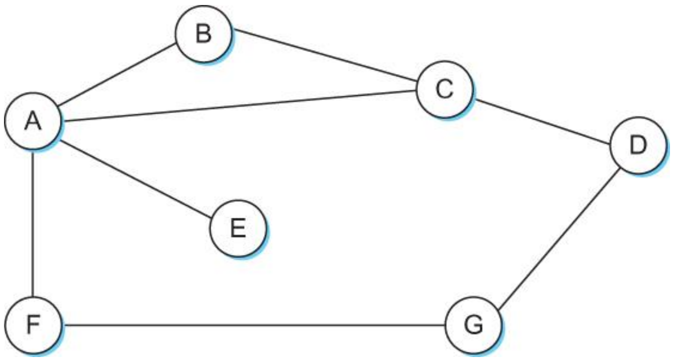
Distance From a Node to Other Nodes



- What is the (shortest) distance from A to B?
- What is the (shortest) distance from A to C?
- What is the (shortest) distance from A to D?

Distance Vector: Example of Initial Routing Table

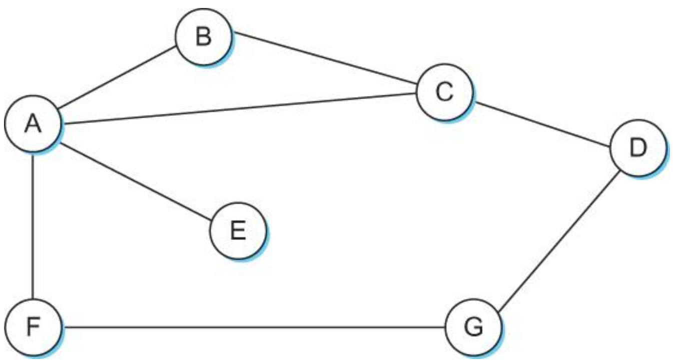
- Initial routing table at node A



Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Distance Vector: Example of Final Routing Table

- Final routing table at node A

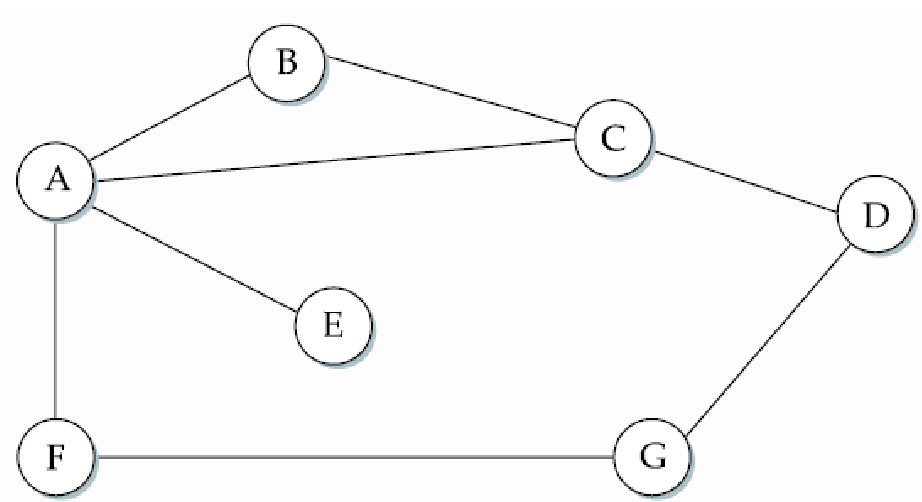


Distance vector: distances from A to the other nodes

Destination	Cost	NextHop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Exercise

- Given an internetwork below, construct the *initial* routing table for the distance vector routing algorithm at *router C* (by filling the provided table below)



Destination	Cost	Next Hop
A		
B		
D		
E		
F		
G		

Distance Vector Routing Algorithm

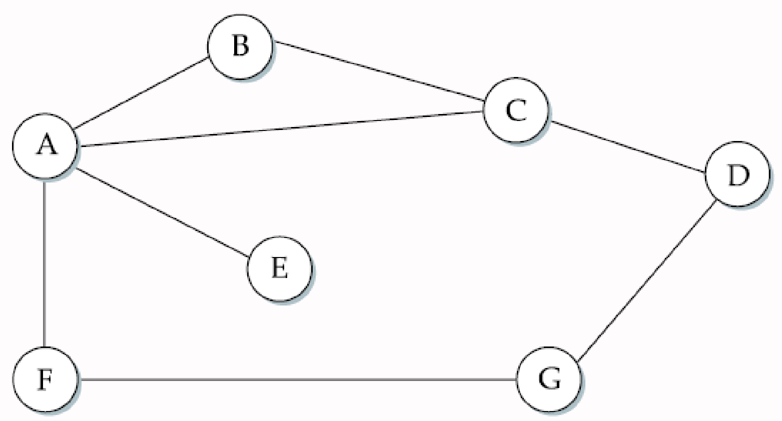
- Main idea
  - Every T seconds each router sends its table to its neighbor each router then updates its table based on the new information
- Problems
  - Fast response to good news, but slow response to bad news
  - Also too many messages to update

Distance Vector Routing Algorithm: More Details

- Each node maintains a routing table consisting of a set of triples
  - (Destination, Cost, NextHop)
- Exchange updates directly connected neighbors
  - periodically (on the order of several seconds)
  - whenever table changes (called *triggered update*)
- Each update is a list of pairs:
  - (Destination, Cost): from sending router to destination
  - Update local table if receive a "better" route
    - smaller cost
    - came from next-hop
- Refresh existing routes; delete if they time out

Table Update

- Example: Exchange updates between A and C



- Then A sends an update to C

Destination	Cost
B	1
C	1
D	∞
E	1
F	1
G	∞

C's initial routing table

Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	∞	-
F	∞	-
G	∞	-

C's updated routing table

Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	2	A
F	2	A
G	∞	-

Table Update from A at C

Destination	Cost
B	1
C	1
D	∞
E	1
F	1
G	∞

+ 1 =

Destination	Cost	Next Hop
B	2	A
C	2	A
D	∞	A
E	2	A
F	2	A
G	∞	A

Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	∞	-
F	∞	-
G	∞	-

Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	2	A
F	2	A
G	∞	-

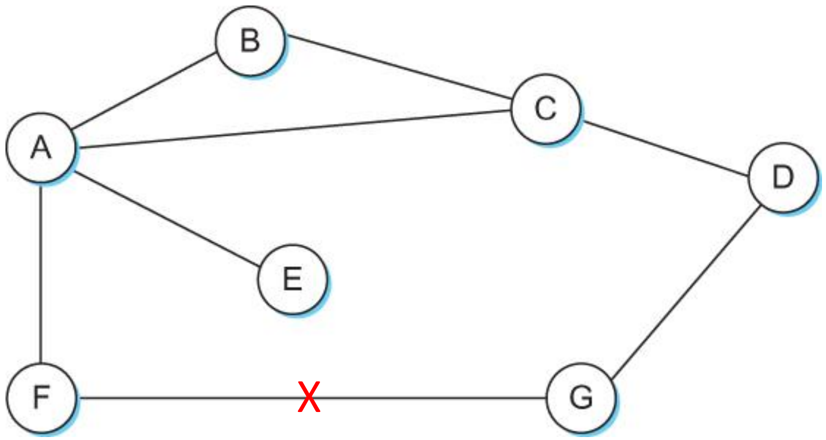
Convergence

- Process of getting consistent routing information to all the nodes
- Desired results: routing tables converges to a stable *global* table (no more changes upon receiving updates from neighbors)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

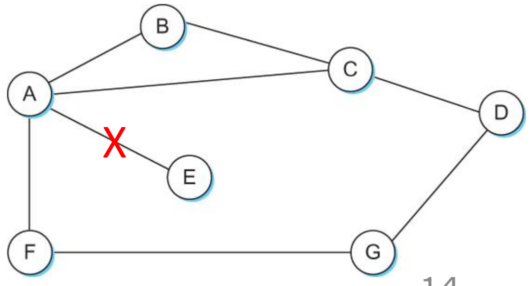
Link Failure: Example

- When a node detects a link failure
  - F detects that link to G has failed
  - F sets distance to G to infinity and sends update to A
  - A sets distance to G to infinity since it uses F to reach G
  - A receives periodic update from C with 2-hop path to G
  - A sets distance to G to 3 and
  - F decides it can reach G in 4

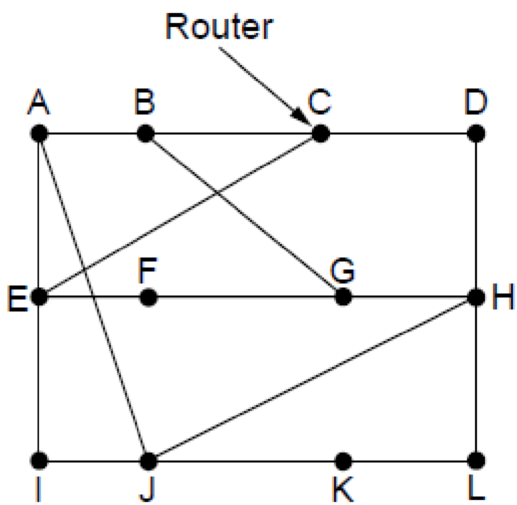


Count-to-infinity Problem

- Slightly different circumstances can prevent the network from *stabilizing*
  - Suppose the link from A to E goes down
  - In the next round of updates, A advertises a distance of infinity to E, but B and C advertise a distance of 2 to E
  - Depending on the exact timing of events, the following might happen
    - Node B, upon hearing that E can be reached in 2 hops from C, concludes that it can reach E in 3 hops and advertises this to A
    - Node A concludes that it can reach E in 4 hops and advertises this to C
    - Node C concludes that it can reach E in 5 hops; and so on.
    - This cycle stops only when the distances reach some number that is large enough to be considered infinite
    - *called* **count-to-infinity problem**



Distance Vector: Example



Network

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 vector for J	

Vectors received at J from Neighbors A, I, H and K

Count-to-Infinity Problem: Example

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

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 of a path to *A* spreads quickly

A	X	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
			⋮			
		•	•	•	•	

Bad news of no path to *A* is learned slowly

Count-to-infinity Problem: Solutions

- Use some relatively small number as an approximation of infinity
- For example, the maximum number of hops to get across a certain network is never going to be more than 16
  - Set infinity to 16
  - Stabilize fast, but not working for larger networks
- One technique to improve the time to stabilize routing is called *split horizon*

Split Horizon

- When a node sends a routing update to its neighbors, it does *not* send those routes it learned from each neighbor *back* to that neighbor
- For example, if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, and so whenever B sends a routing update to A, it does not include the route (E, 2) in that update

Split Horizon with Poison Reverse

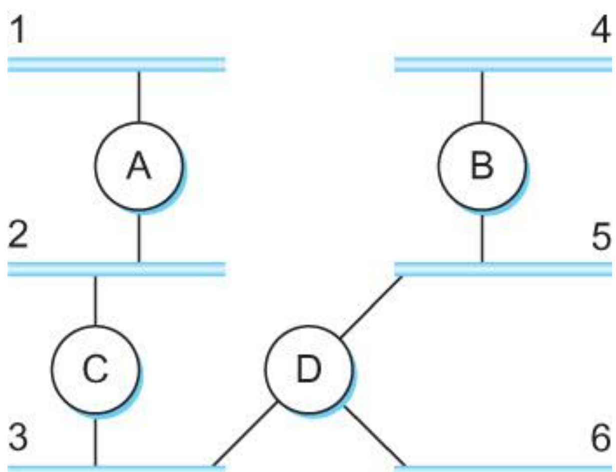
- In a stronger version of split horizon, called *split horizon with poison reverse*
  - B actually sends that back route to A, but it puts negative information in the route to ensure that A will not eventually use B to get to E
  - For example, B sends the route (E, ∞) to A

Routing Information Protocol

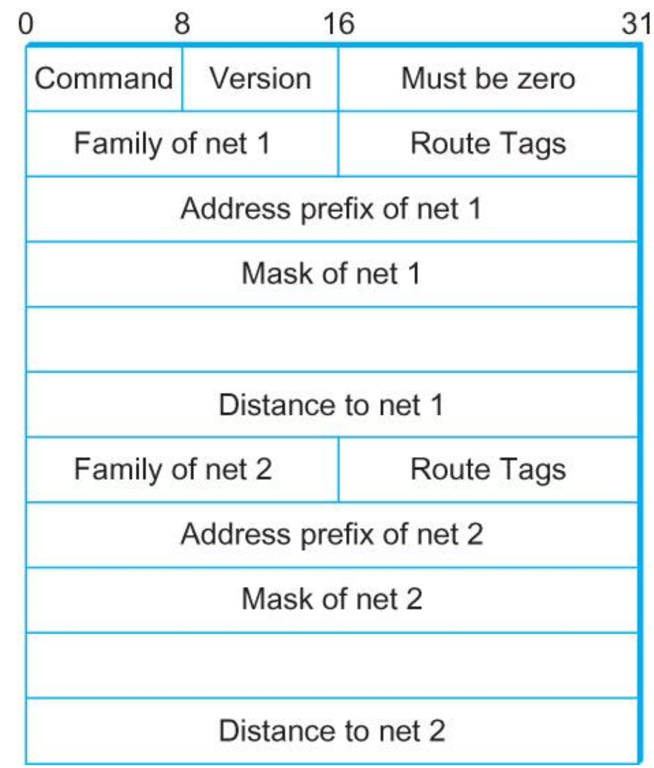
- Routing Information Protocol (RIP)
  - Initially distributed along with BSD Unix
  - Widely used
- Straightforward implementation of distance-vector routing

Routing Information Protocol (RIP)

- Distance: cost (# of routers) of reach a network
  - C → A
    - Network 2 at cost 0; 3 at cost 0
    - Network 5 at cost 1, 4 at 2



Example Network



RIPv2 Packet Format

Questions?

- Shortest path problem and algorithm
- Distance Vector Routing
  - Problems with distance vector routing?
- Implementation of Distance Vector Routing
  - Routing information protocol

Link State Routing

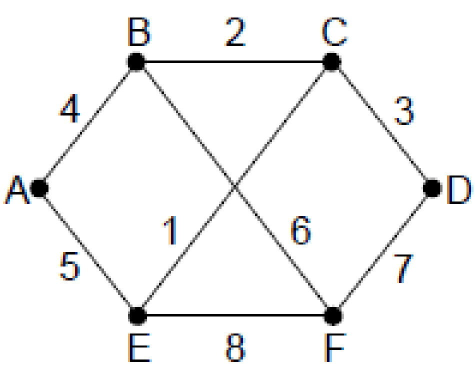
- Link state is an alternative to distance vector
  - More computation but simpler dynamics
  - Widely used in the Internet (OSPF, ISIS)
- Algorithm:
  - Each node floods information about its neighbors in LSPs (Link State Packets); all nodes learn the full network graph
  - Each node runs Dijkstra's algorithm to compute the path to take for each destination

Link State Routing

- Strategy: Send to all nodes (not just neighbors) information about directly connected links (not entire routing table).
- Link State Packet (LSP)
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet
- Reliable Flooding
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - generate new LSP periodically; increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP; discard when TTL=0

Link State Packet: Example

LSP (Link State Packet) for a node lists neighbors and weights of links to reach them



Network

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

LSP for each node

Reliable Flooding

- Reliable flooding triggered by
  - Timer
  - Topology or link cost change
- increment SEQNO
  - start SEQNO at 0 when reboot
  - SEQNO does not wrap
    - e.g., 64 bits
  - decrement TTL of each stored LSP
- discard when TTL=0

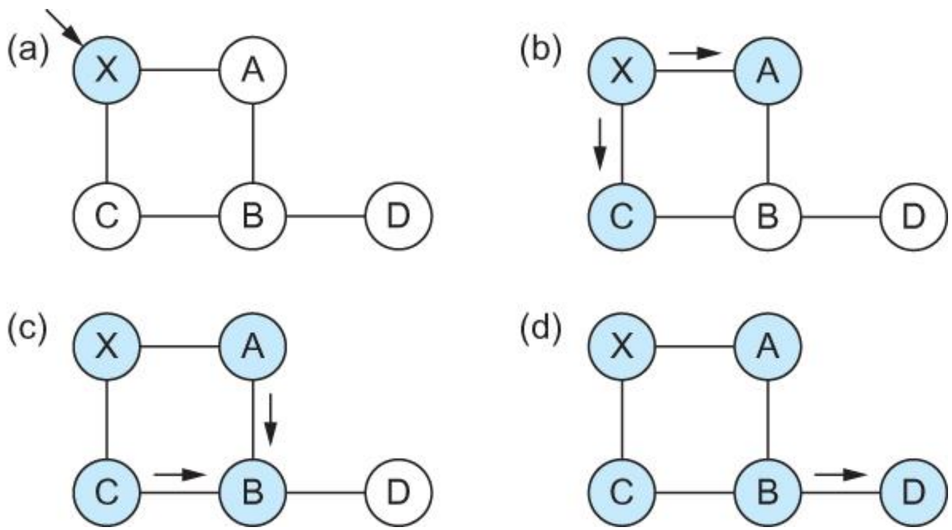
Reliable Flooding: Example

- Seq. number and age are used for reliable flooding
  - New LSPs are acknowledged on the lines they are received and sent on all other lines
  - Example shows the LSP database at router B

			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Reliable Flooding: Example

- Reliable Flooding



- Flooding of link-state packets. (a) LSP arrives at node X; (b) X floods LSP to A and C; (c) A and C flood LSP to B (but not X); (d) flooding is complete

Shortest Path Routing Algorithm

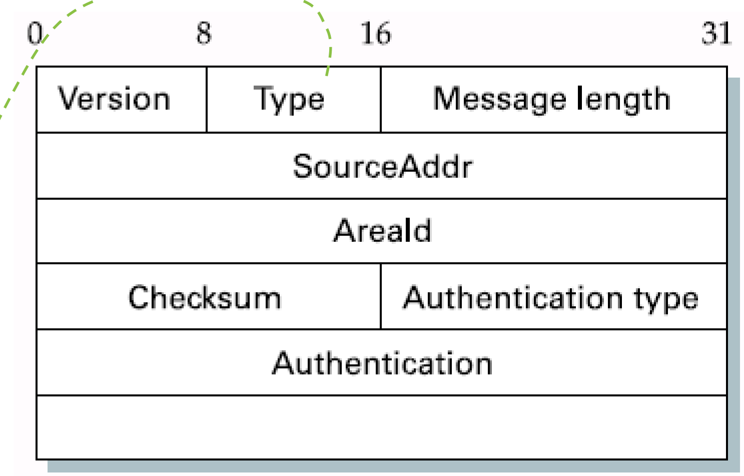
- In practice, each switch computes its routing table directly from the LSPs it has collected using a realization of Dijkstra's algorithm called the *forward search algorithm*
- Specifically each switch maintains two lists, known as **Tentative** and **Confirmed**
- Each of these lists contains a set of entries of the form (Destination, Cost, NextHop)

Link State in Practice

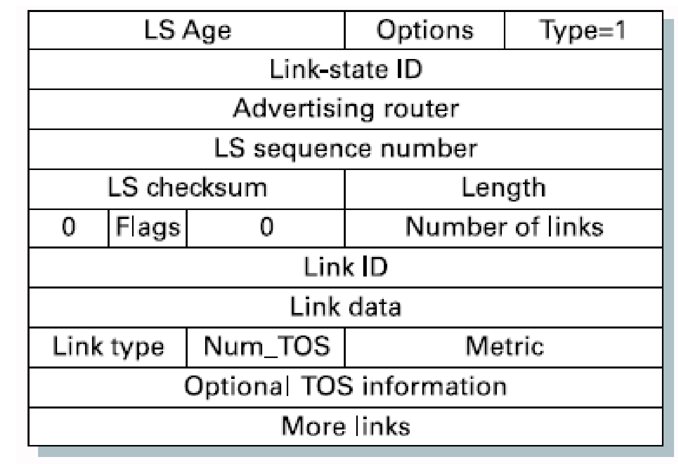
- Open Shortest Path First Protocol (OSPF)
  - "Open" → open, non-proprietary standard, created under the auspices of the IETF
  - "SPF" → Shortest Path First, alternative name of link-state routing
- Implementation of Link-State Routing with added features
  - Authenticating of routing messages
    - Due to the fact too often some misconfigured hosts decide they can reach every host in the universe at a cost of 0
  - Additional hierarchy
    - Partition domain into areas → increase scalability
  - Load balancing
    - Allows multiple routes to the same place to be assigned the same cost → cause traffic to be distributed evenly over those routes

Open Shortest Path First Protocol

OSPF Header Format



OSPF Link State Advertisement



Type	Packet name	Protocol function
1	Hello	Discover/maintain neighbors
2	Database Description	Summarize database contents
3	Link State Request	Database download
4	Link State Update	Database update
5	Link State Ack	Flooding acknowledgment

Questions

- Link State Routing

Questions?

- Distance Vector
  - Algorithm
  - Routing Information Protocol (RIP)
- Link State
  - Algorithm
  - Open Shortest Path First Protocol (OSPF)