Module 5
Routing Algorithms

Mr. A. Swaminathan

Routing

- Network Layers
  - Routing
  - Congestion control
  - Internetworking
- The process of forwarding the packets in a network to reach its intended destination.

Goals of routing

- Correctness
- Simplicity
- Robust
- Stability
- Fairness
- Optimality

The purpose of any routing protocol

- to dynamically communicate information about all network paths used to reach a destination
- to select the *best* path to reach a destination network.

Routing Classification

- Adaptive / Dynamic
- Gathers information at run time locally from neighbour routers
- Changes routes periodically
  - For every t seconds
  - load changes
  - Topology changes

- Non-Adaptive (Static)
- Route is predetermined offline and downloaded to the routers when the network is booted

Routing Types

- Distance Vector routing
  - Routing information Protocol
- Link state routing
  - Open Shortest Path First OSPF
- Shortest path routing
- Flooding
- Flow based routing

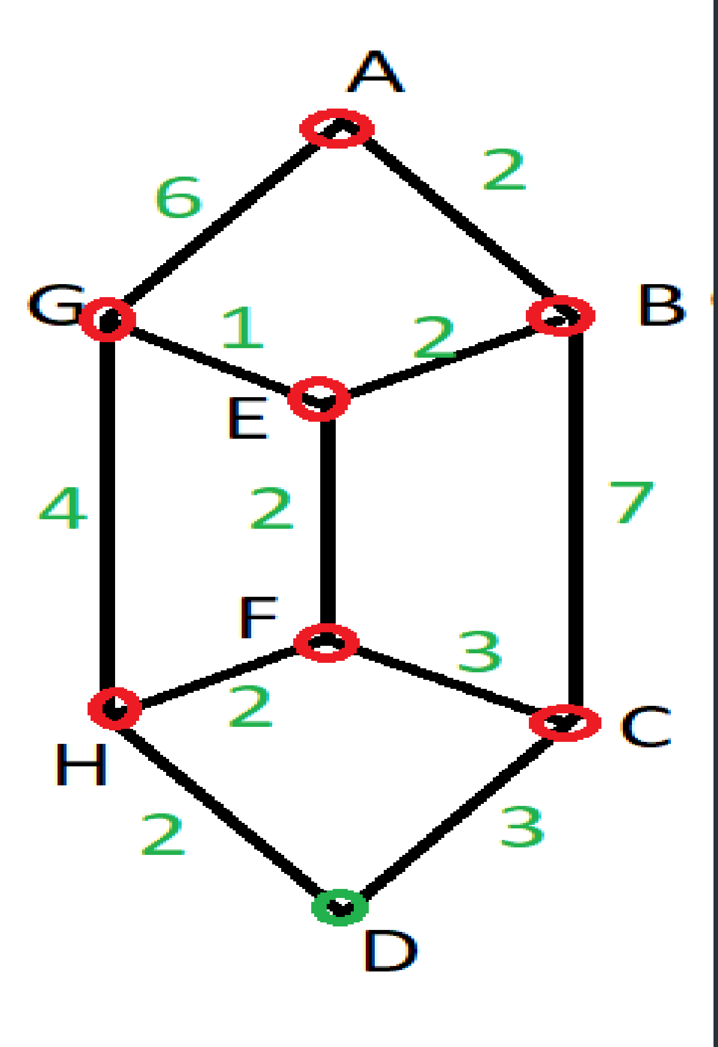
Routing types

- Hierarchical routing
- Routing for mobile host

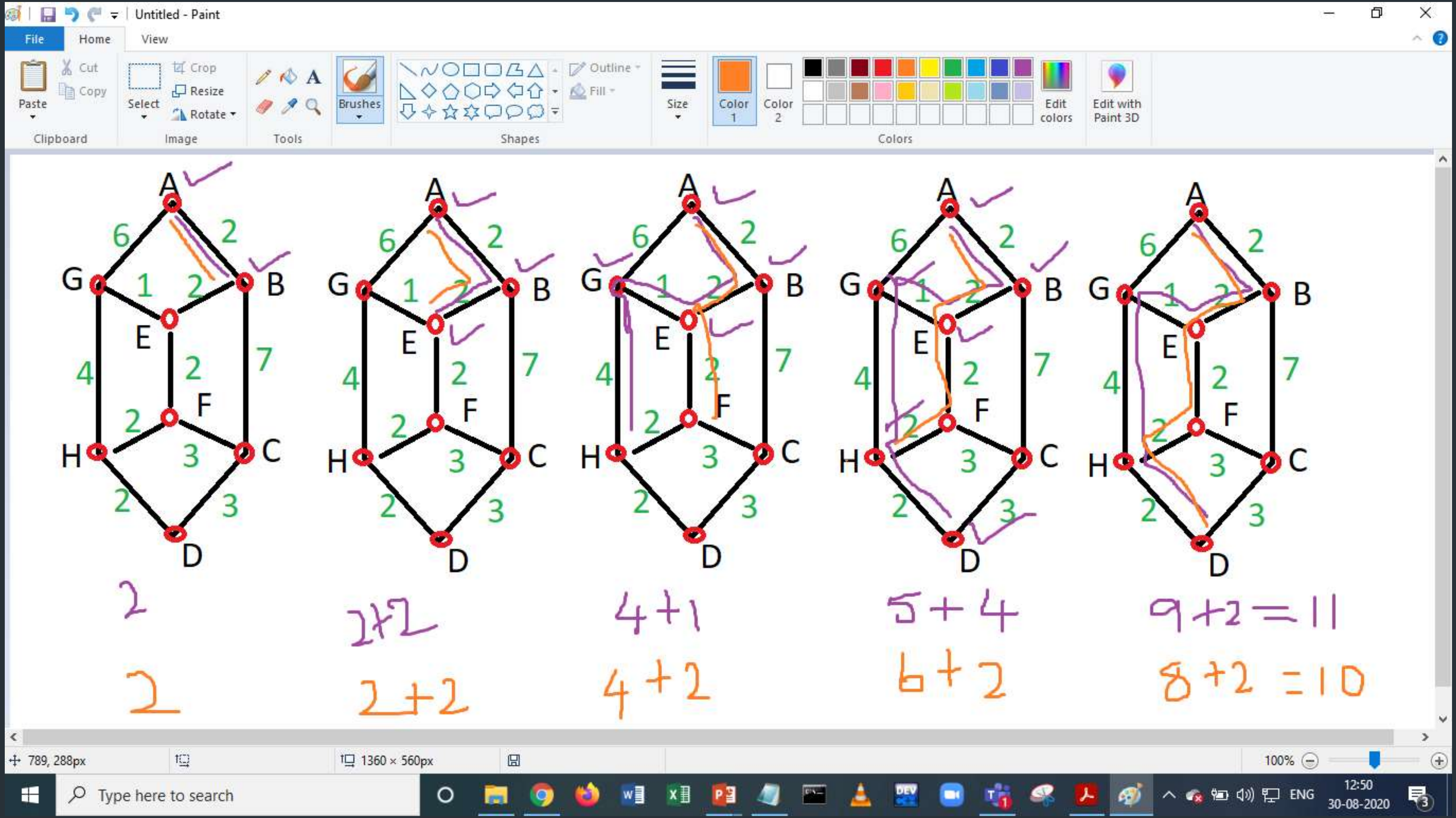
Shortest path routing algorithm

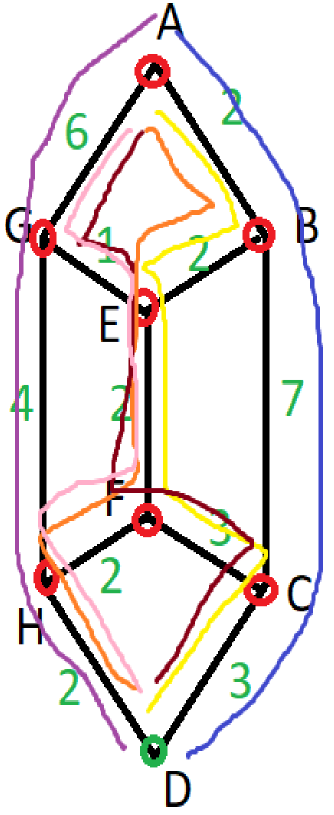
- Finds the shortest path between the pair of routes
- The cost of link depends upon
  - Distance
  - Bandwidth
  - Average traffic
  - Communication cost
  - Delay

Dijkstra's Algorithm



	A	B	C	D	E	F	G	H
A	0							
B		0						
C			0					
D				0				
E					0			
F						0		
G							0	
H								0


Untitled - Paint
File
Home
View
Paste
Cut
Copy
Clipboard
Select
Crop
Resize
Rotate
Image
Tools
Brushes
Outline
Fill
Shapes
Size
Color 1
Color 2
Colors
Edit colors
Edit with Paint 3D
A B C D E F G H
6 2 1 2 4 2 7 2 3 2 3
2
2
2+2
2+2
4+1
4+2
5+4
6+2
9+2=11
8+2=10
789, 288px
1360 × 560px
100%
Type here to search
ENG
12:50
30-08-2020



A - B - C - D = 12

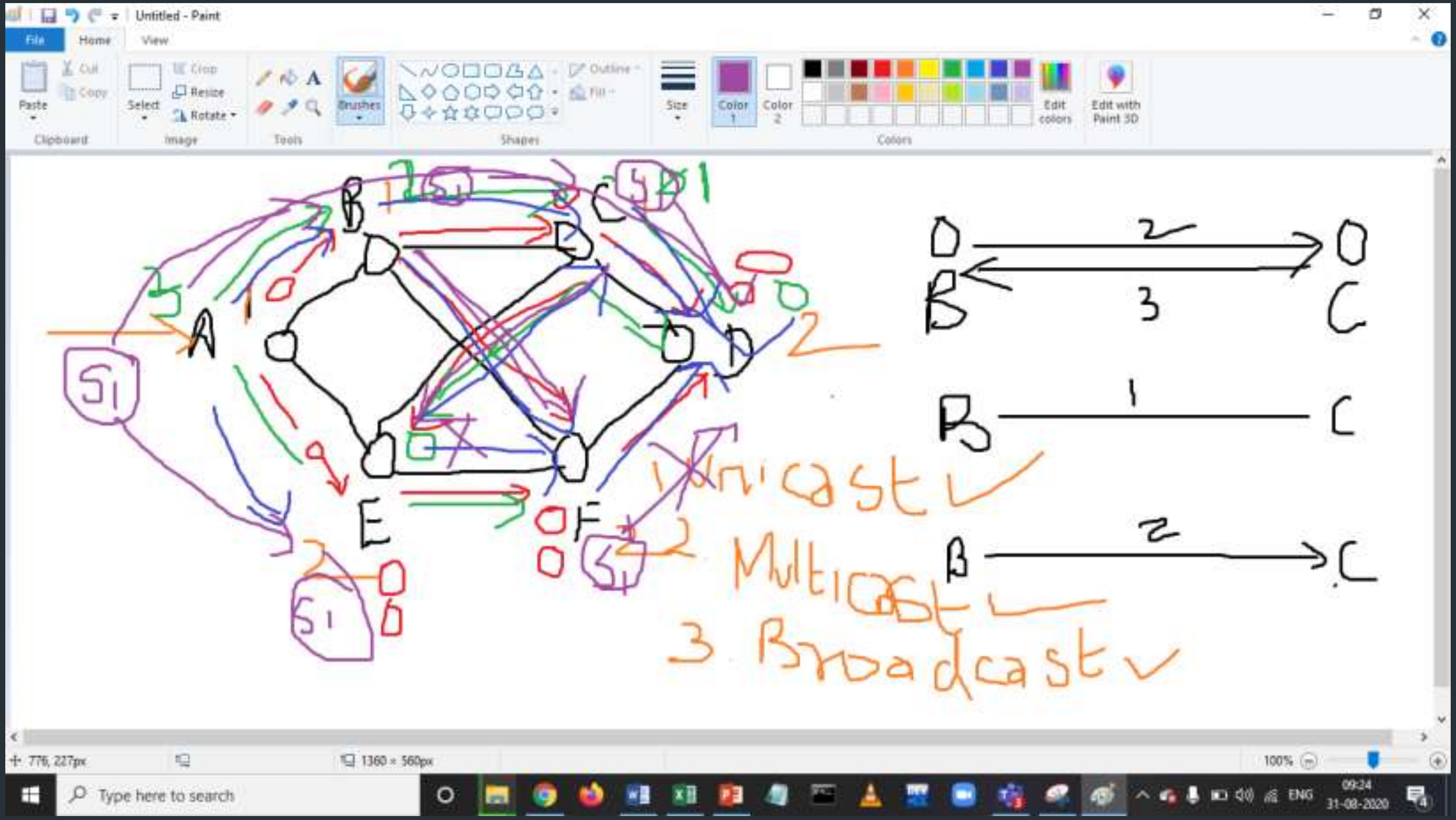
A - G - H - D = 12

A - B - E - F - C - D = 12

A - B - E - F - H - D = 10

A - G - E - F - C - D = 13

A - G - E - F - H - D = 13

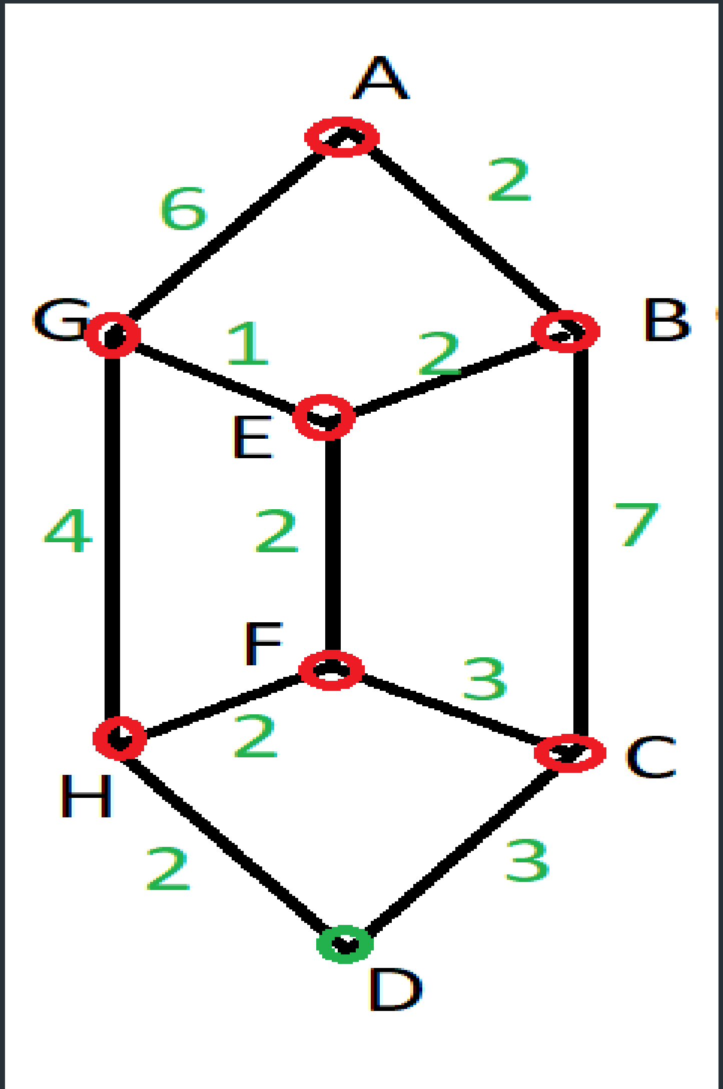


1. Unicast ✓

2. Multicast ✓

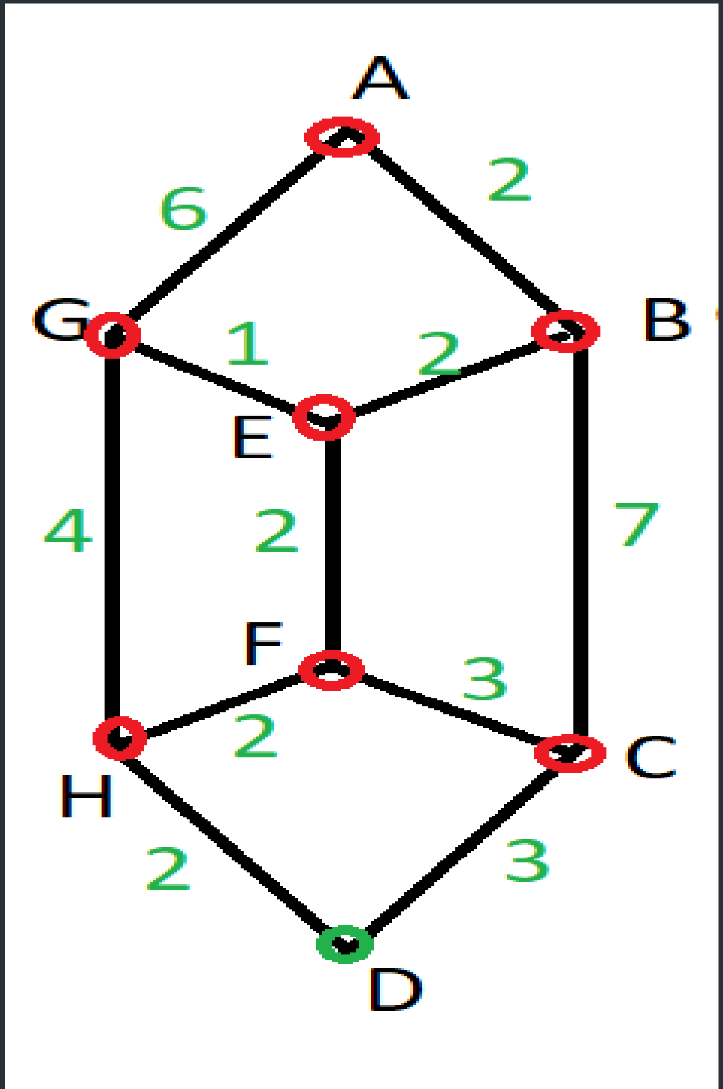
3. Broadcast ✓

Dijkstra's Algorithm



	A	B	C	D	E	F	G	H
A	0	2	∞	∞	∞	∞	6	∞
B	2	0	7	∞	2	∞	∞	∞
C	∞	7	0	3	∞	3	∞	∞
D	∞	∞	3	0	∞	∞	∞	2
E	∞	2	∞	∞	0	2	1	∞
F	∞	∞	3	∞	∞	0	∞	2
G	6	∞	∞	∞	1	∞	0	4
H	∞	∞	∞	2	∞	2	4	0

Dijkstra's Algorithm



	A	B	C	D	E	F	G	H
A	0	2					6	
B	2	0	7		2			
C		7	0	3		3		
D			3	0				2
E		2			0	2	1	
F			3			0		2
G	6				1		0	4
H				2		2	4	0

Flooding

- Every incoming packets is sent to every outgoing path. (Broadcast) except the source.
- Disadvantages
  - Large no. of duplicate packets
  - Bandwidth
- How to eliminate the duplicate
  - Hop counter
    - Decrement each router
    - Discard packet if counter is '0'
  - Apply sequence number in packet
    - It avoid sending the same packet to a router for second time
    - Maintain the source list in each router
  - Selective Flooding
    - Select approximately right direction

Flow based Routing

- Routing based on
    - Topology
    - Network traffic/Load
    - Quick sending
    - Finds shortest path
        - If path is Network traffic/Load
        - Find next shortest path

Flow based technology

- Subnet topology (different routes)
- Traffic matrix
- Line capacity matrix (Bandwidth)

Distance Vector Routing

- Least cost between pair of nodes
- Bellman Ford Algorithm
- One routing table
- $d_x(y)=\min\{cost(x,v)+d_v(y)\}$
- d_x = Source
- Y = destination
- V = intermediate node

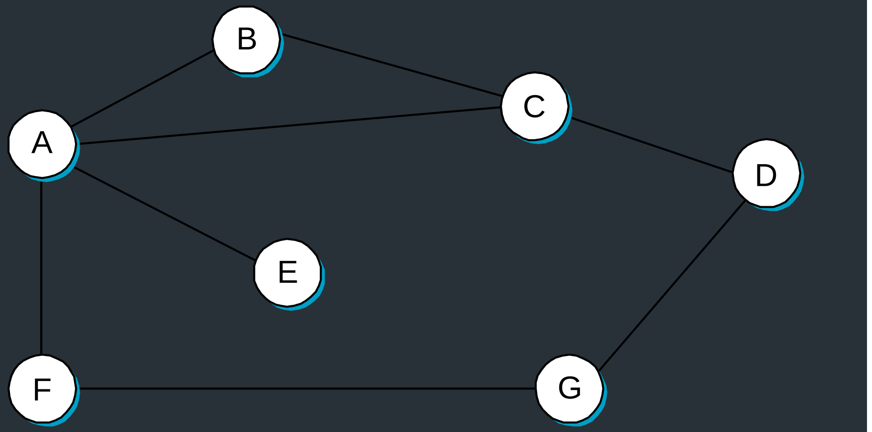
Distance Vector Routing

- Completely decentralized
- No node has complete information about the costs of all network links
- Gradual calculation of path by exchanging information with neighbors

Specifics

- Each node constructs a one-dimensional array containing the “distances” or “costs” to all other nodes (as it relates to its knowledge) and distributes it to its immediate neighbors.
- Key thing -- each node knows the cost of links to its neighbors.
- If no link exists between two nodes, the cost of a direct link between the nodes is “infinity”.

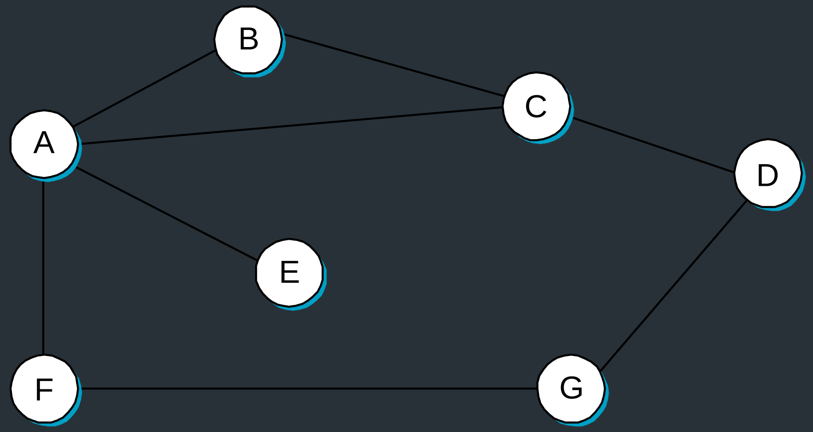
An Example



- Internal Information at each node ------>

	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Routing Tables

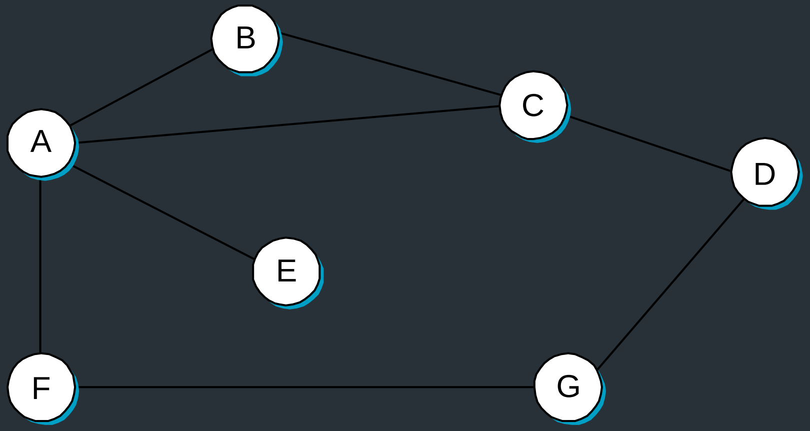


- With this information, routing table at A is -->

	Cost	Next Hop
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	F
G	∞	-

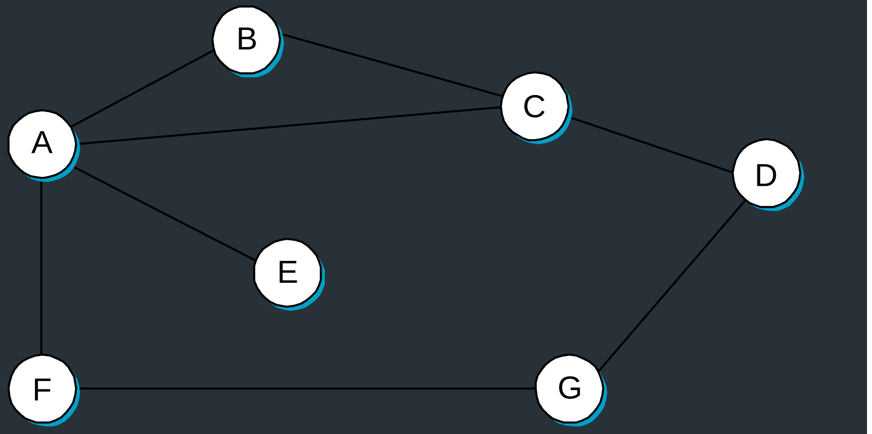
Evolution of the table.

- Each node sends a message to neighbors with a list of distances.
- F --> A with G is at a distance 1
- C --> A with D at distance 1.



	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Final Distance Matrix



	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

Convergence

- In the absence of topological changes -- few exchanges between neighbors before complete routing table is formed.
- This table is consistent.
- Convergence is achieved.
- Notice -- no centralized authority

Routing updates

- When are routing updates sent ?

1. Periodic updates
   - Even if nothing has changed, send periodically. Main reason is to let other nodes know that the sender is alive.
   - Refresh information that might be needed if some of the routes were to become unavailable.
2. Triggered updates
   - When a node receives an update from one of its neighbors which may lead to a change in its routing tables (could be due to change in link cost).

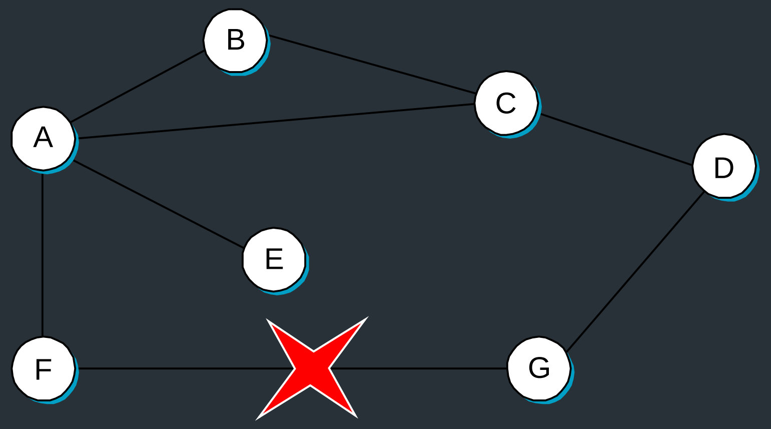
- Note: typically order of periodicity is seconds to several minutes.

Link/Node Failures

- Nodes that first notice send new lists of distances to neighbors.
- How do they detect failures ?
  - Route updates don’t arrive
  - Probing with test packets.

Example Revisited

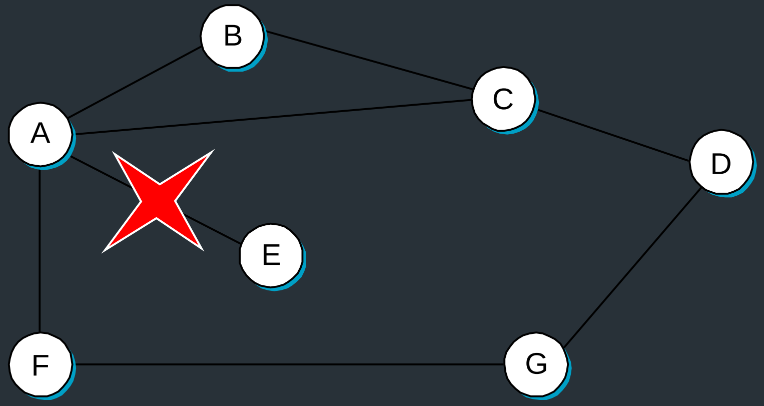
- Let link from F to G fail.
- F sets new distance to G to ∞; sends update to A.
- A was initially routing to G via F. So it now sets link cost to G to ∞.
- Next update from C; A learns that C has 2 hop path to G.
- A now can reach G in 3 hops via C.
- A sends an update to F. Thus, F now, can reach G via A in 4 hops.



Count to Infinity

- A discovers that link to E is lost.
- If before A's message (saying that link cost to E is ∞) is received, if B or C advertise that they can reach E in two hops, then A can be confused.
- Another possibility, B gets A's update followed by C's update which says that E is reachable in 2 hops.
- So B tells A this, and A thinks it can now reach E via B in 3 hops.
- This information reaches C who now thinks that it can reach E in 4 hops via A.

- **The process continues and thus, the system does not stabilize.**
- **This is the count to infinity problem.**



Split Horizon

- One solution would be to approximate ∞ to say 16 hops.
- With Split Horizon, when a node sends a routing table update to its neighbors, it “does not” send those routes it learned from “a particular” neighbor, back to that neighbor.
  - For example, B had E, 2, A. When it sends a route update to A, it does not include this.
- With split horizon with poison reverse, this update is reported but the link weight is set to ∞.
  - For example B sends (E, ∞) to A.

Does this work ?

- Typically, in static networks where link failures/node failures are rare, this may be enough.
- Speed of convergence is why, link state routing may be preferable -- it takes a while before routes converge.

Purging routing entries

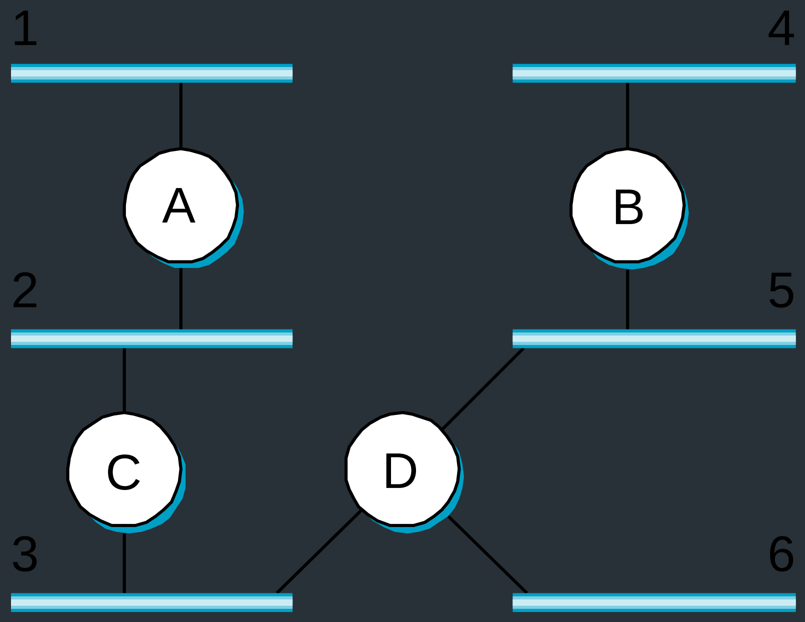
- Each routing entry has a time-to-live (or TTL) field.
- A counter -- initially set to MaxTTL.
- This is then decremented and if TTL = 0, then, time to purge the entry.

RIP

- Stands for Routing Information Protocol
- Built based on distance vector routing.
- In the Internet, goal of routers is to learn how to forward packets to various networks.
- Send/receive updates (Routing information) from neighbors
- Routing table & update time – 30 sec

An Example of RIP

- Metric : hop count
- Rumors based protocol
- Loop issue – star hybrid
- Routers advertise the cost of reaching networks.
- In this example, C’s update to A would indicate that C can reach Networks 2 and 3 with cost 0, Networks 5 and 6 with cost 1 and Network 4 with cost 2.



Other RIP Details

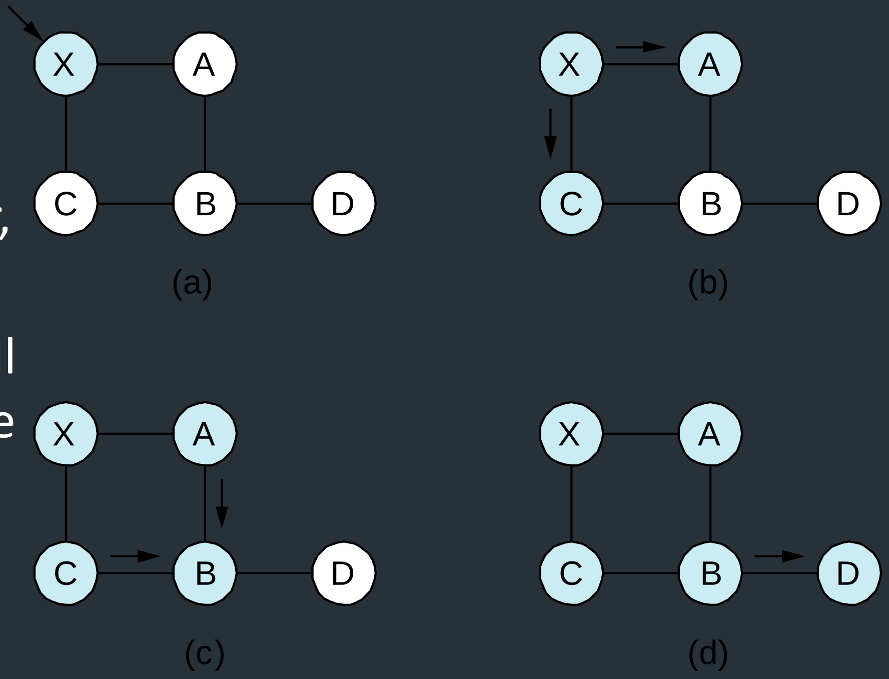
- Routing tables are exchanged every 30 seconds using the RIP advertisement.
- If a router does not hear from its neighbor once every 180 seconds, the neighbor is deemed unreachable.
- The router that detects this will modify its routing table and propagate the information.

RIP implementation

- RIP packets are sent using UDP.
- Typically, there is a routing daemon (routed) that is an application layer process that provides access to routing tables.
- Allows for the access of the tables.
- Use “netstat -rn” to view routing table at host.

An Example

- X receives LSP from some node Y.
- X checks to see if it already has an update from Y. If it does, it compares the sequence number in the new LSP to the one stored.
- If New seq no < Old sequence number, then, discard LSP.
- Else -- store LSP and send the LSP to all neighbors except the one that sent the LSP.
- If no update from Y, keep it.



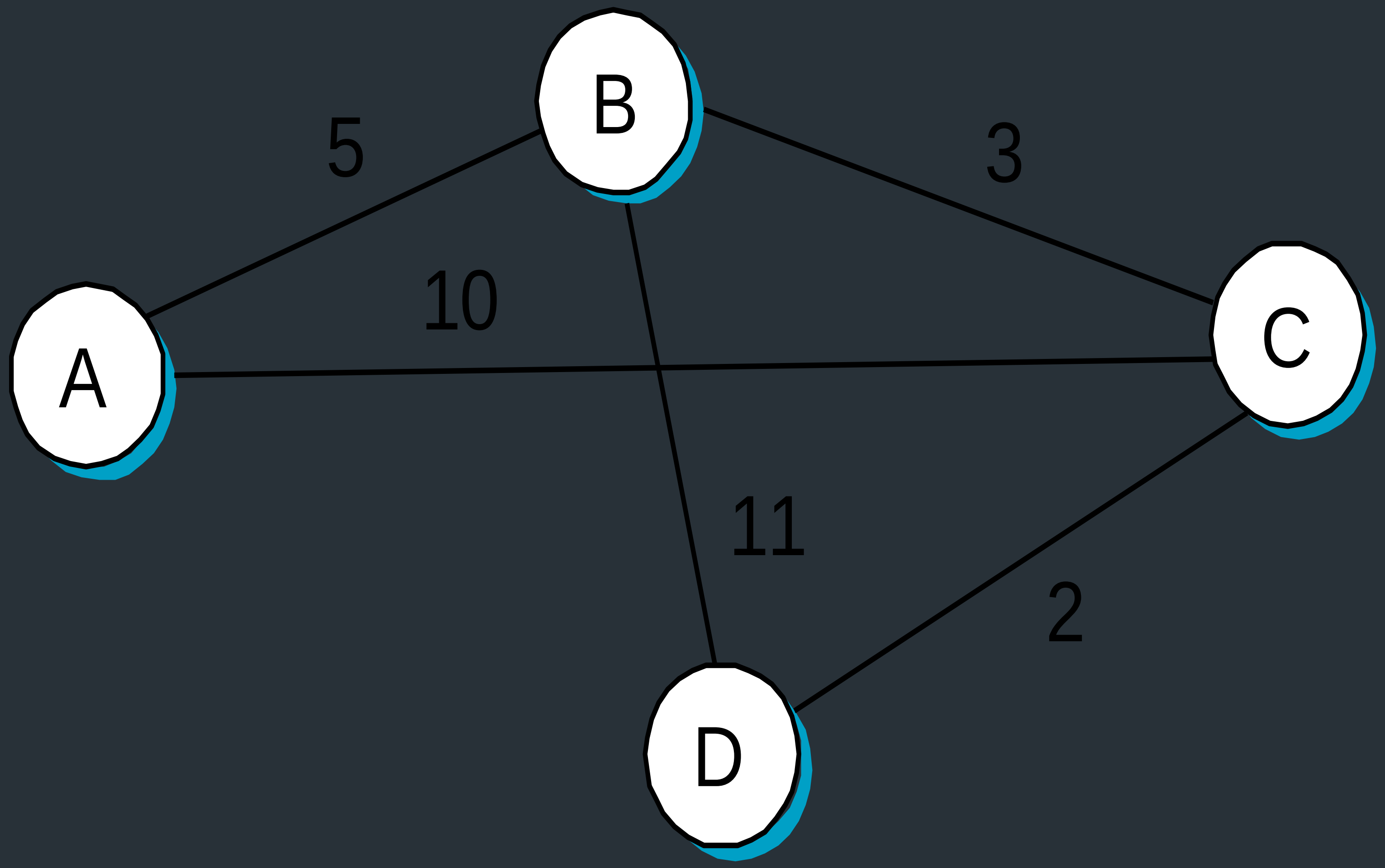
Dissemination of LSPs

- LSPs are sent periodically (upon the expiry of a timer) or may be triggered due to a change in topology (as in RIP).
- The only topology change that triggers the creation of a new LSP is a change to one of the directly connected links.
  - Failures detected by link layer protocol by using what are known as “HELLO” packets -- probes to determine if neighbor is alive.
- To minimize overhead, LPSs are not created unless needed --periodicity is of the order of hours.
- Sequence numbers help in identifying new info and TTL helps in ensuring that packets don’t stay in the network indefinitely.

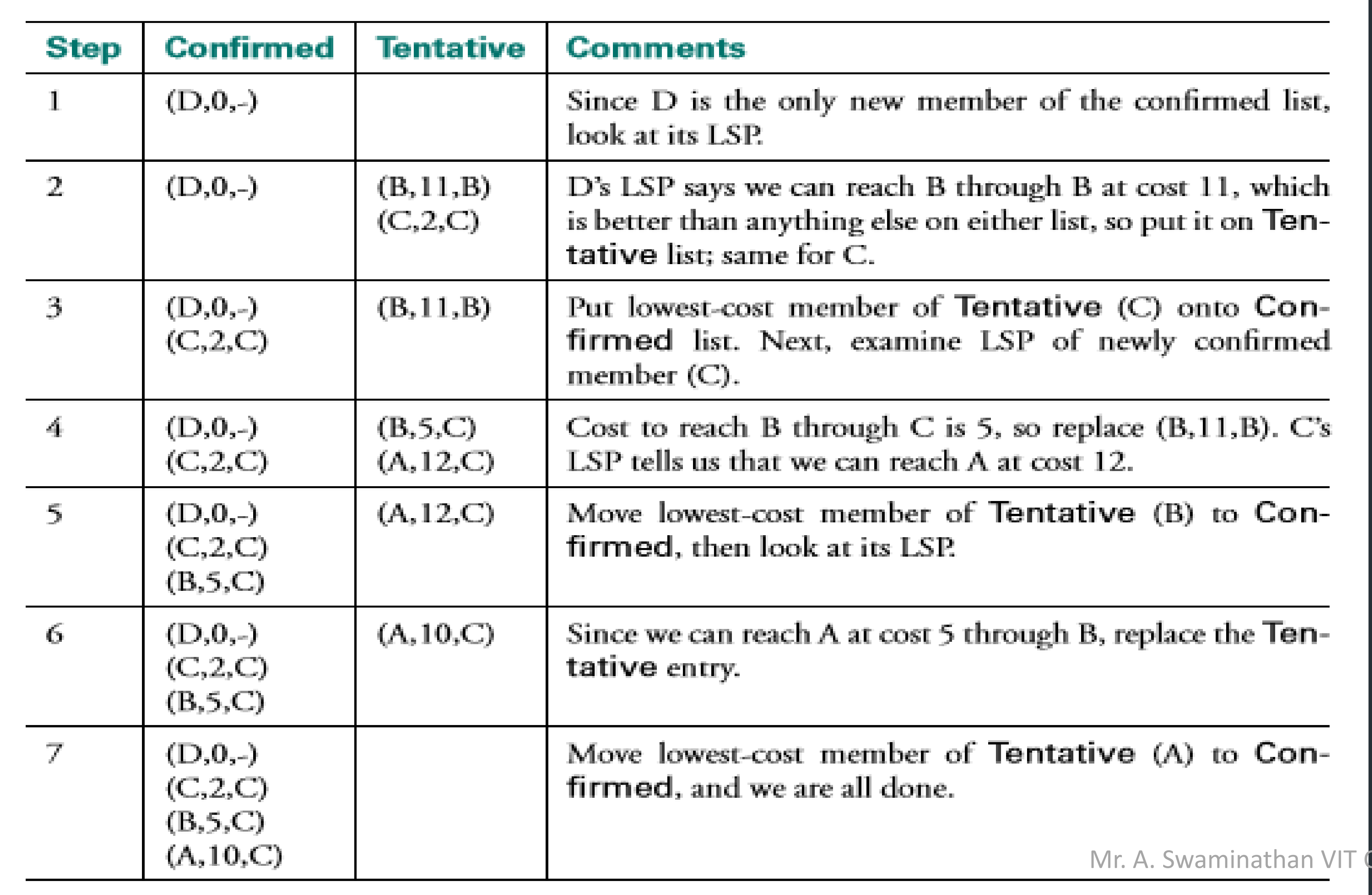
Graph abstraction

- Used for computation of shortest path using Dijkstra’s.
- Let N denote the set of nodes in the graph.
- l(i,j) denotes the non-negative cost or weight associated with the edge between nodes i and j.
- l(i,j) = ∞ if there is no edge between i and j.
- Remember -- each node has entire map of network.

An Example



Steps for building routing table of node D



Step	Confirmed	Tentative	Comments
1	(D,0,-)		Since D is the only new member of the confirmed list, look at its LSP.
2	(D,0,-)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better than anything else on either list, so put it on Tentative list; same for C.
3	(D,0,-) (C,2,C)	(B,11,B)	Put lowest-cost member of Tentative (C) onto Confirmed list. Next, examine LSP of newly confirmed member (C).
4	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
5	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to Confirmed, then look at its LSP.
6	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative entry.
7	(D,0,-) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative (A) to Confirmed, and we are all done.

Link State Routing

- Initial state : similar to distance vector i.e., state of link to neighbors known (up/down).
- Goal : To find the path of least cost to destination.
- Basic Idea -- Every node knows how to reach its neighbors. If this info is dissemination to every node, every node ultimately has the info. to build the complete map of the network.
- Open Shortest Path First:

Link State Information

- Each node creates a link-state packet (LSP) that contains:
  - ID of the node that created LSP
  - a list of directly connected nodes and the cost to each node.
  - sequence number
  - TTL

} → **for reliability**

Open Shortest Path First

- Link state protocol
- It is more complex than EIGRP.
- Features
  - Open standard protocol
  - it supports features such as Variable length subnet mask (VLSM)/ Classless Inter-Domain Routing (CIDR) etc
  - Highly scalable
  - Does not have hop count limit
  - It works in multivendor environment
  - Minimizes works between neighbors

OSPF

- route computation using Dijkstra’s algorithm
- It supports hierarchical design
- larger network divided into smaller networks
- Even though these areas are separated but it works with single OSPF autonomous system
- EIGRP is different from other it works on multiple autonomous system
- Though the areas are independent the EIGRP requires redistribution and vice versa

OSPF (Open Shortest Path First)

- “Open”: publicly available

- Uses Link State algorithm
  - link state (LS) packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm

OSPF “Advanced” Features (not in RIP)

- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different Type Of Service (eg, satellite link cost set “low” for best effort; high for real time)
- Security: all OSPF messages authenticated (to prevent malicious intrusion); TCP connections used
- Hierarchical OSPF

OSPF

- Advantages
  - decrease routing overhead and flow of updates
  - limit network problems such as instability to an area
  - Speed up convergence
- disadvantages
  - planning and Configuration is difficult

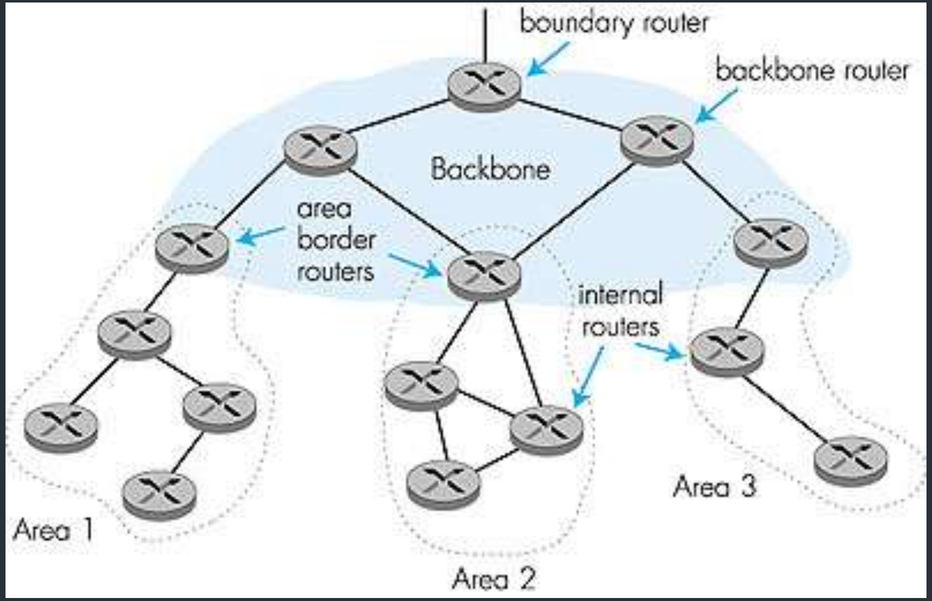
OSPF “Advanced” Features (not in RIP)

- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different Type Of Service (eg, satellite link cost set “low” for best effort; high for real time)
- Security: all OSPF messages authenticated (to prevent malicious intrusion); TCP connections used
- Hierarchical OSPF

Hierarchical OSPF

“summarize” distances to nets in own area, advertise to other Area Border routers.

run OSPF routing limited to backbone.

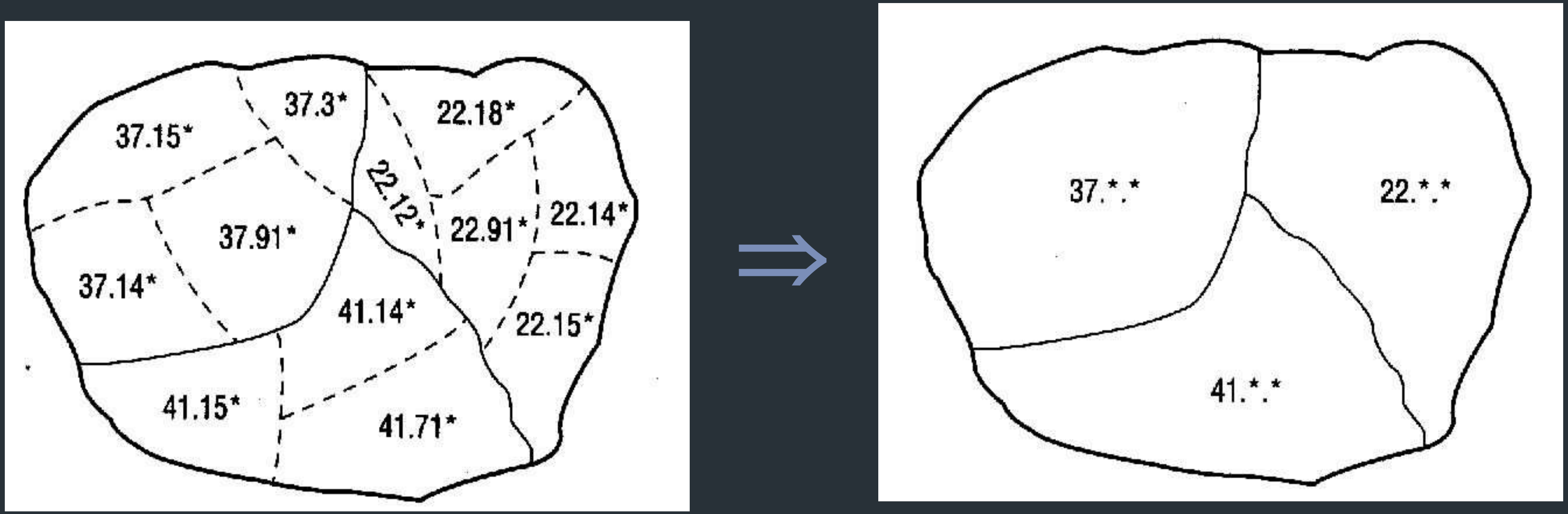


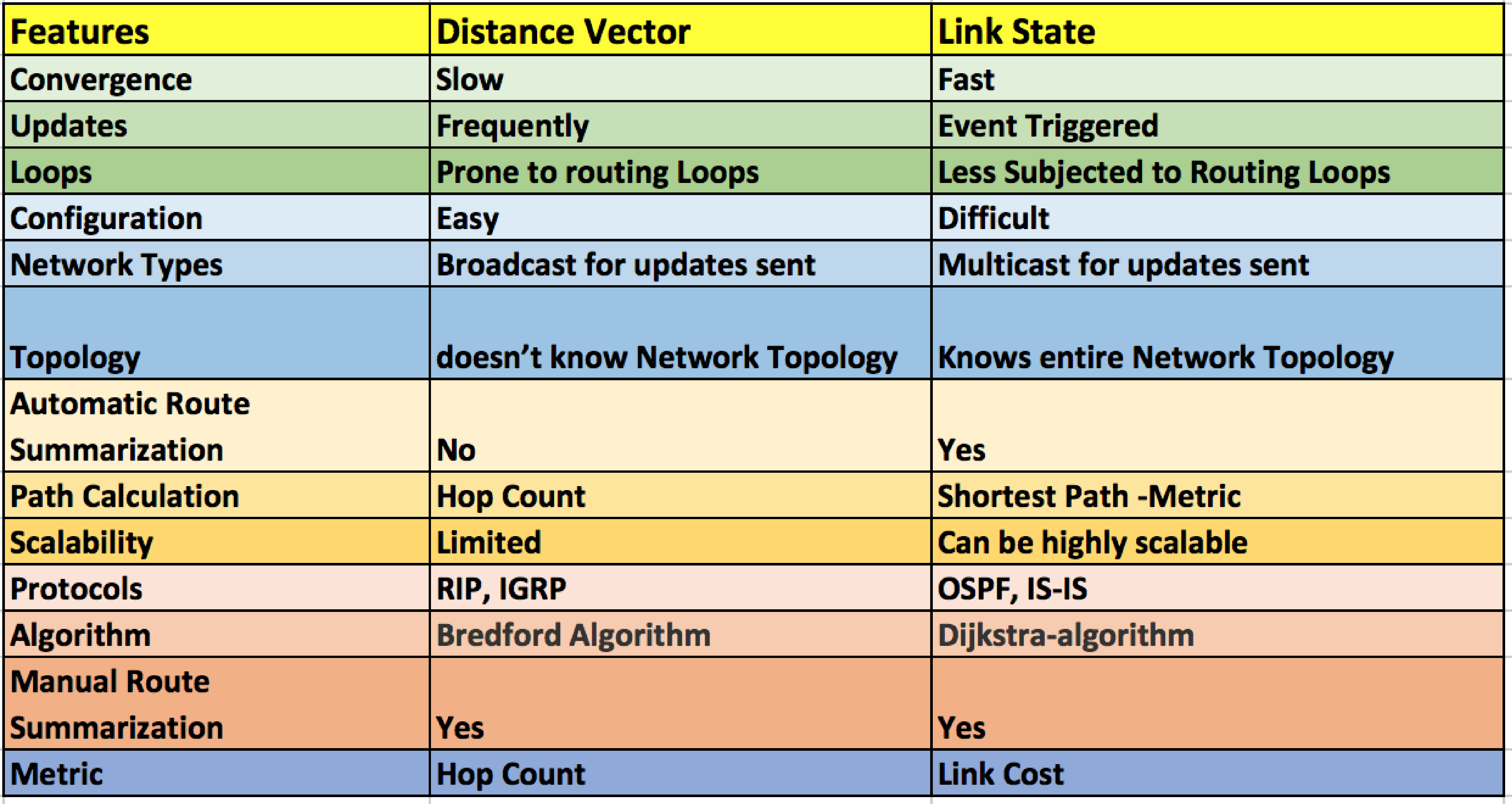
- Link-state advertisements only in area each nodes has detailed area topology;
- only know direction (shortest path) to nets in other areas.

Two-level hierarchy: local area, backbone.

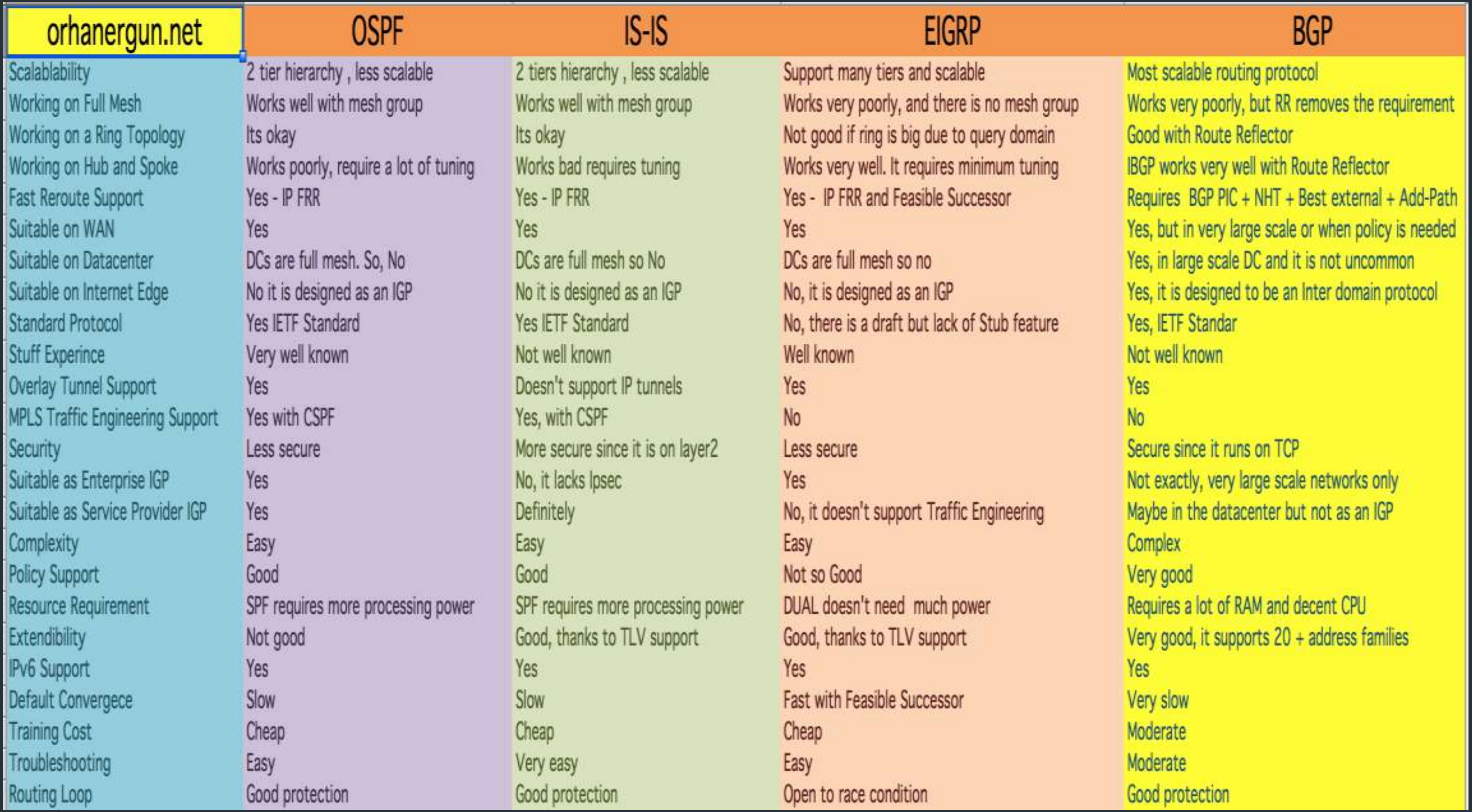
Why Hierarchy?

- Information hiding (filtered) => reduce computation, bandwidth, storage





Features	Distance Vector	Link State
Convergence	Slow	Fast
Updates	Frequently	Event Triggered
Loops	Prone to routing Loops	Less Subjected to Routing Loops
Configuration	Easy	Difficult
Network Types	Broadcast for updates sent	Multicast for updates sent
Topology	doesn’t know Network Topology	Knows entire Network Topology
Automatic Route Summarization	No	Yes
Path Calculation	Hop Count	Shortest Path -Metric
Scalability	Limited	Can be highly scalable
Protocols	RIP, IGRP	OSPF, IS-IS
Algorithm	Bredford Algorithm	Dijkstra-algorithm
Manual Route Summarization	Yes	Yes
Metric	Hop Count	Link Cost



orhanergun.net	OSPF	IS-IS	EIGRP	BGP
Scalablability	2 tier hierarchy , less scalable	2 tiers hierarchy , less scalable	Support many tiers and scalable	Most scalable routing protocol
Working on Full Mesh	Works well with mesh group	Works well with mesh group	Works very poorly, and there is no mesh group	Works very poorly, but RR removes the requirement
Working on a Ring Topology	Its okay	Its okay	Not good if ring is big due to query domain	Good with Route Reflector
Working on Hub and Spoke	Works poorly, require a lot of tuning	Works bad requires tuning	Works very well. It requires minimum tuning	IBGP works very well with Route Reflector
Fast Reroute Support	Yes - IP FRR	Yes - IP FRR	Yes - IP FRR and Feasible Successor	Requires BGP PIC + NHT + Best external + Add-Path
Suitable on WAN	Yes	Yes	Yes	Yes, but in very large scale or when policy is needed
Suitable on Datacenter	DCs are full mesh. So, No	DCs are full mesh so No	DCs are full mesh so no	Yes, in large scale DC and it is not uncommon
Suitable on Internet Edge	No it is designed as an IGP	No it is designed as an IGP	No, it is designed as an IGP	Yes, it is designed to be an Inter domain protocol
Standard Protocol	Yes IETF Standard	Yes IETF Standard	No, there is a draft but lack of Stub feature	Yes, IETF Standar
Stuff Experince	Very well known	Not well known	Well known	Not well known
Overlay Tunnel Support	Yes	Doesn't support IP tunnels	Yes	Yes
MPLS Traffic Engineering Support	Yes with CSPF	Yes, with CSPF	No	No
Security	Less secure	More secure since it is on layer2	Less secure	Secure since it runs on TCP
Suitable as Enterprise IGP	Yes	No, it lacks Ipsec	Yes	Not exactly, very large scale networks only
Suitable as Service Provider IGP	Yes	Definitely	No, it doesn't support Traffic Engineering	Maybe in the datacenter but not as an IGP
Complexity	Easy	Easy	Easy	Complex
Policy Support	Good	Good	Not so Good	Very good
Resource Requirement	SPF requires more processing power	SPF requires more processing power	DUAL doesn't need much power	Requires a lot of RAM and decent CPU
Extendibility	Not good	Good, thanks to TLV support	Good, thanks to TLV support	Very good, it supports 20 + address families
IPv6 Support	Yes	Yes	Yes	Yes
Default Convergece	Slow	Slow	Fast with Feasible Successor	Very slow
Training Cost	Cheap	Cheap	Cheap	Moderate
Troubleshooting	Easy	Very easy	Easy	Moderate
Routing Loop	Good protection	Good protection	Open to race condition	Good protection

References

- CISCO Packet Magazine, 2nd Quarter 2002 http://www.cisco.com/en/US/about/ac123/ac114/ac173/ac168/about_cisco_packet_issue_home.html
- 3Com University – Wireless LANs A Technology Overview www.3com.com/3comu
- National Institute of Standards and Technology Wireless Network Security http://csrc.nist.gov/publications/drafts/draft-sp800-48.pdf