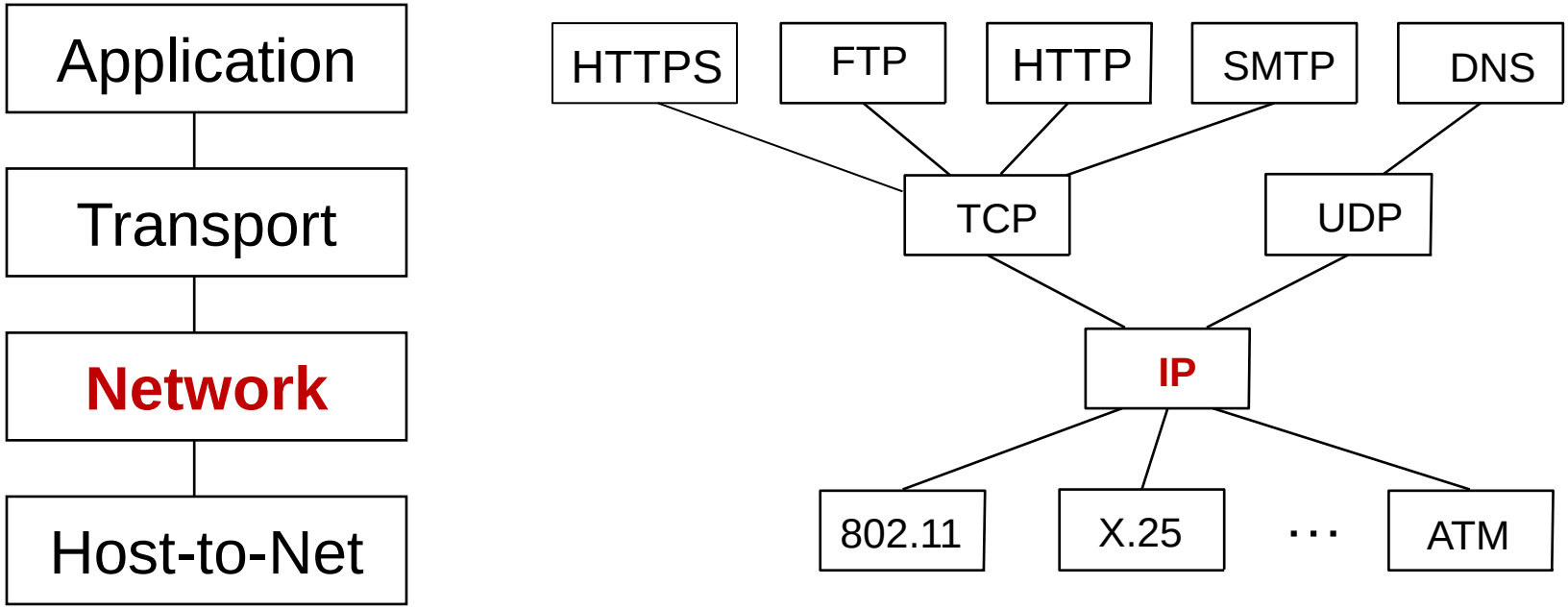
# Routing in Internet

Dr. Vijay Kumar Chaurasiya

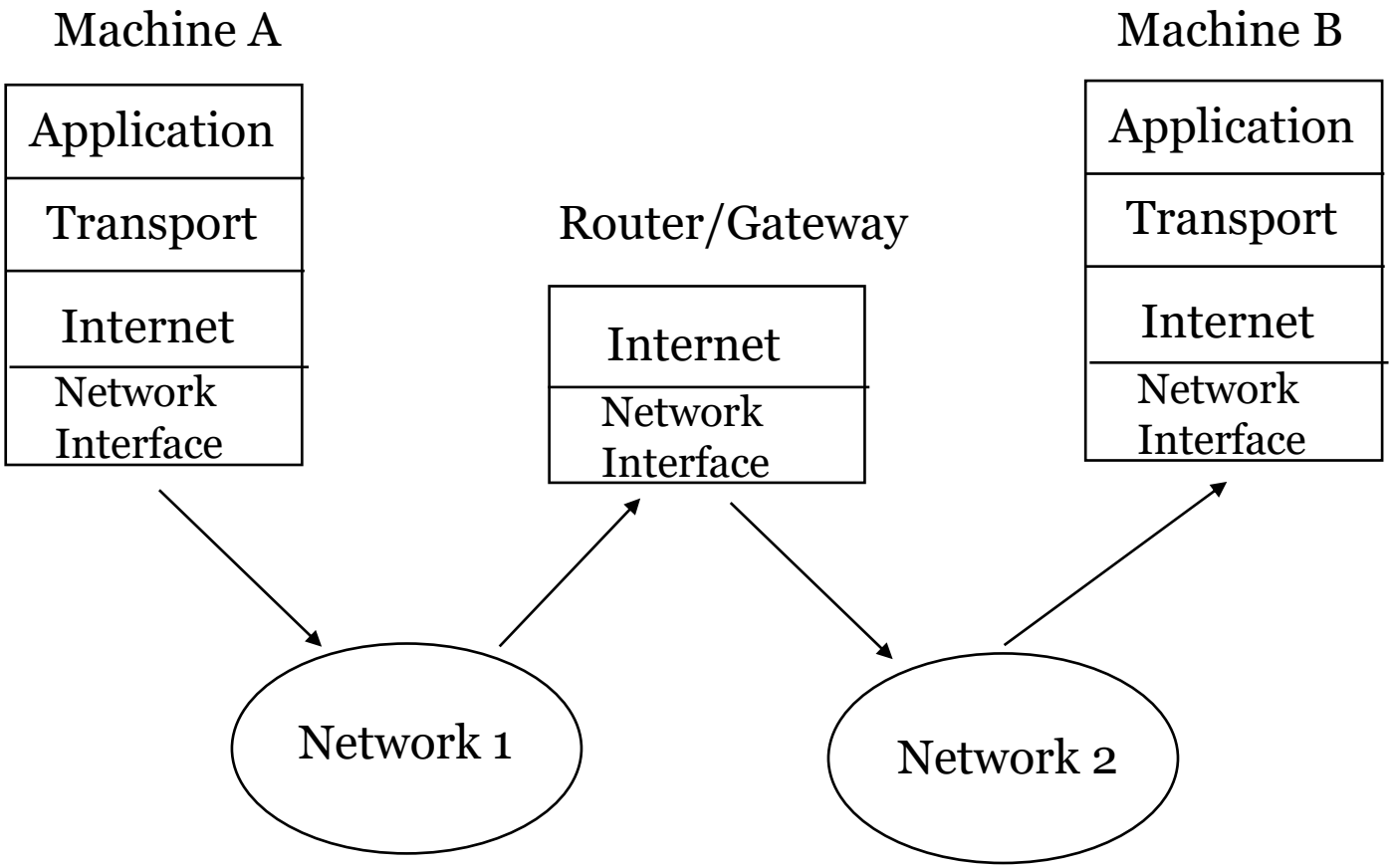
# Network Overview




- The main function of the network layer is to move packets from one endpoint to another.

# Routing Protocols

- Concerned with delivering a packets from source to destination.



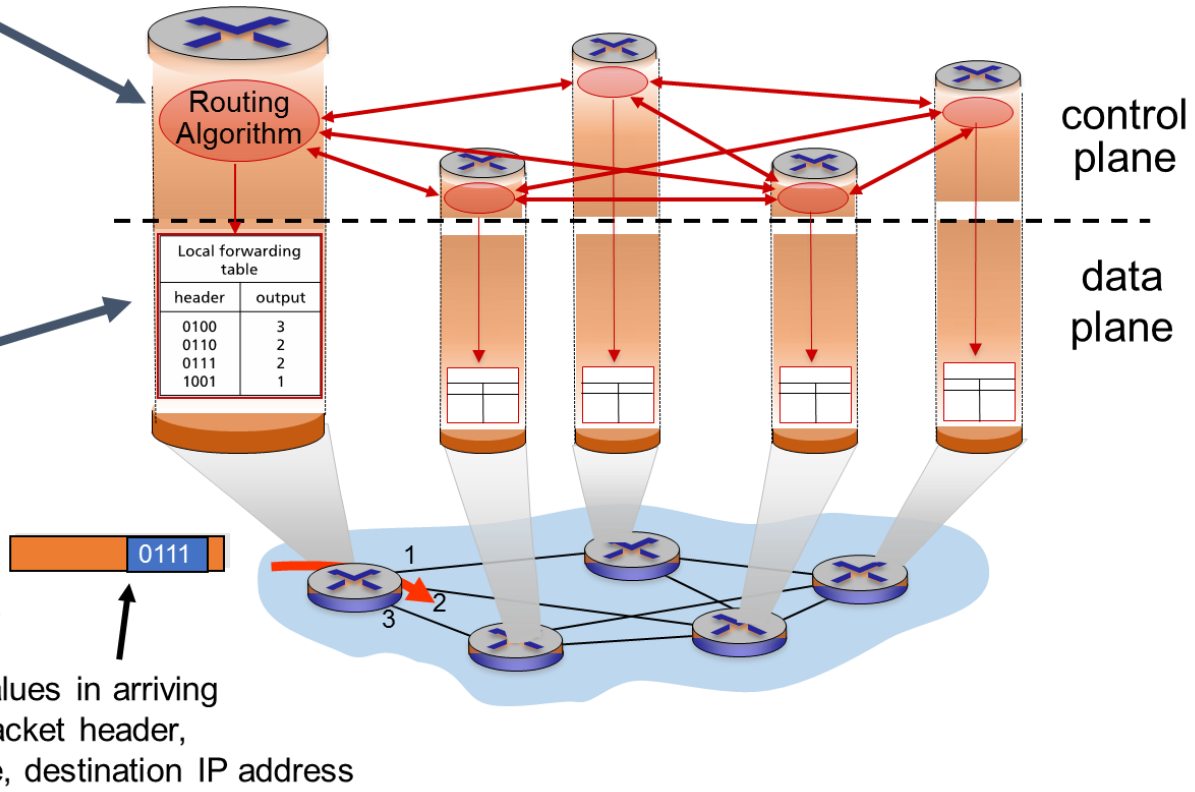
# Review: Network layer functions

| | |
|---|---|
| **Forwarding:** means to move packets from router ' s queue to appropriate router's output interface. | **Managed by Data Plane** |
| **Routing:** mean to determine the route taken by packets from source to destination provided by routing algorithms.<br>**Please Note:** The network layer solves the routing problem. | **Managed by Control Plane**<br>Two kinds of control planes:<br>• Distributed: per-router control (traditional)<br>• Centralized: logically centralized control (software defined networking) |

# Per-router control and data plane

**Distributed control plane:** Components in every router interact with other components to produce a routing outcome.

**Data plane** per-packet processing, moving packet from input port to output port



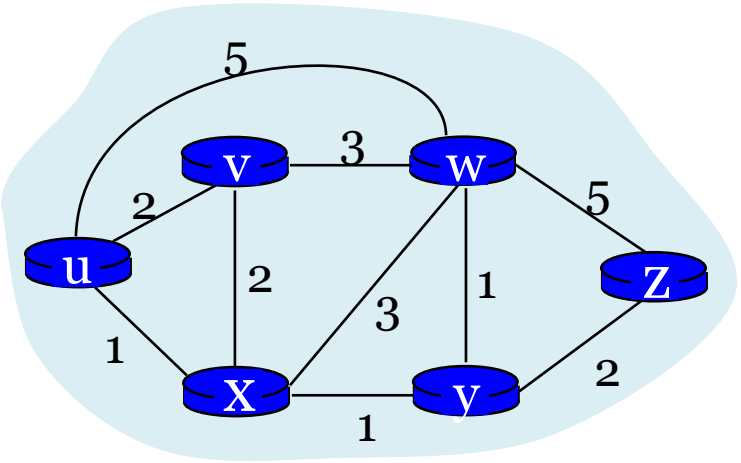
# Goal of Routing Algorithms

- Determine **best paths** from source to destination
- **"best"** = least cost
  - Least propagation delay
  - Least cost per unit bandwidth (e.g., Rs./Gb/s)
  - Least congested (workload-driven)
- **"Path"** = a sequence of router ports (links).
- Routing is a fundamental problem in networking.

# The graph abstraction

- Routing algorithms work over an abstract representation of a network i.e. the graph abstraction

  **Ex:** IIIT Allahabad campus
  u: CC1
  v: CC2
  …

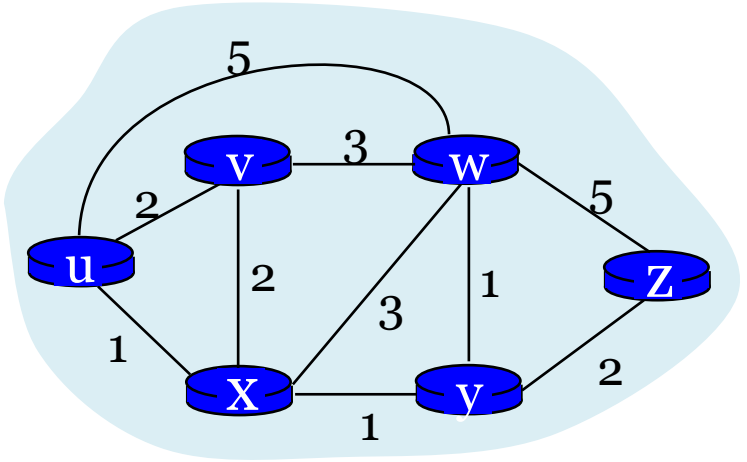



- Each router is a node in a graph.
- Each link is an edge in the graph.
- Edges may have weights (also called link metrics). Set by admin.

**Graph: G = (V,E)**
**V = set of routers = { u, v, w, x, y, z }**
**E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }**

# The graph abstraction (contd.)

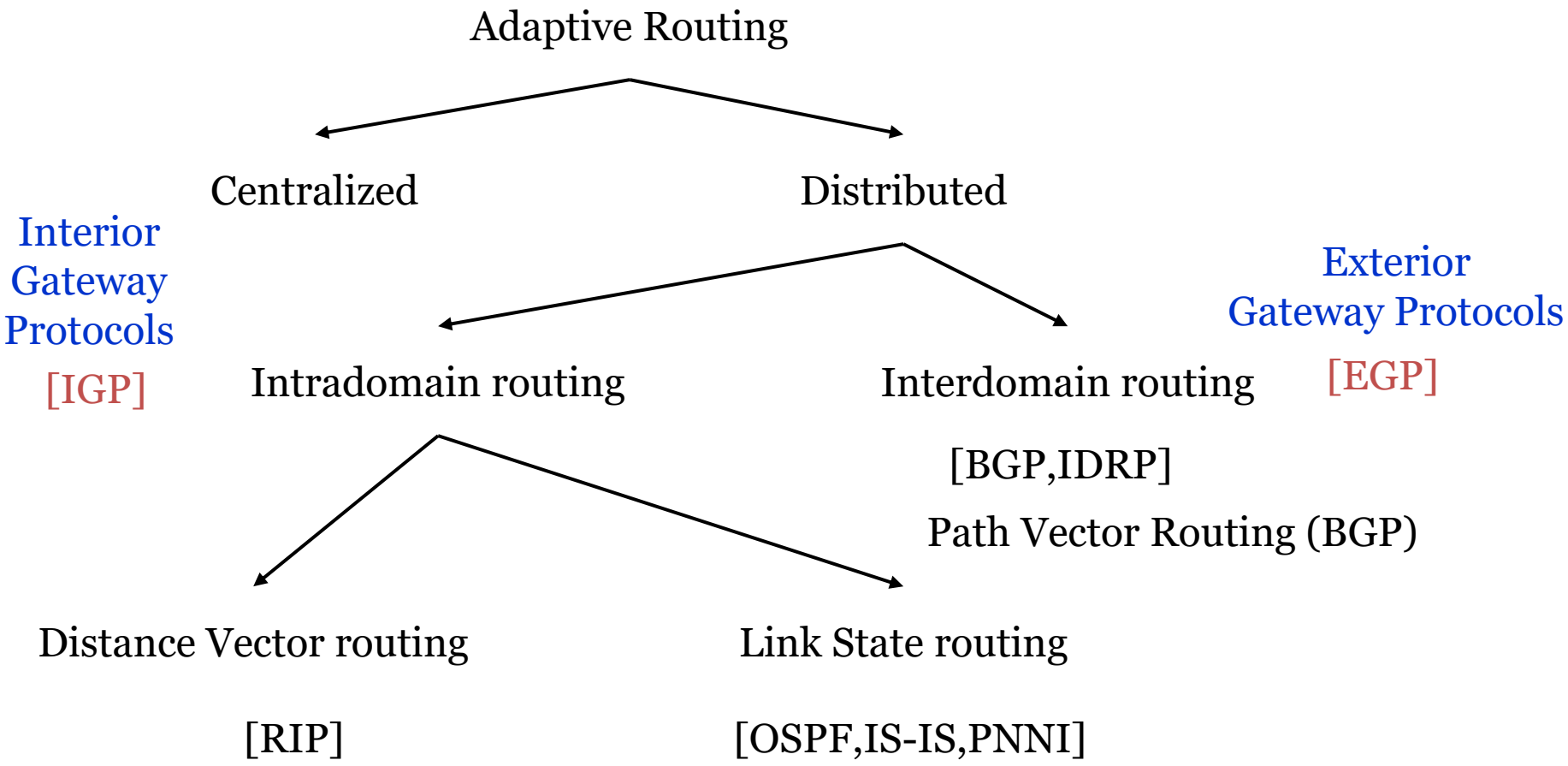
- Cost of an edge: c(x, y)
  - Examples: c(u, v) = 2, c(u, w) = 5
- Cost of a path = summation of edge costs
  - c(path x → w → y → z) = 3 + 1 + 2 = 6
- Outcome of routing: Routing table (each node should determine the least cost path to every other node)
- Q1: What algorithm should each node run to compute the least cost path to every node?
- Q2: What information should nodes exchange with each other to enable this computation?




**Routing Table**

| Destination Net Id | Next Hop | Cost |
|---|---|---|
| 172.23.0.0 | IF1 | 25 |
| 192.26.25.0 | IF7 | 12 |
| …. | … | … |

# Routing Classification



# Intra-domain Routing Protocols

Domain is also known as autonomous system. A domain (AS) is the set of networks under a single administrative control.

# Link State Routing Algorithm

OSPF

# Link State Routing Algorithm

- Distributed routing protocols
- Goal of routing algorithms: find least cost path in a graph abstraction of the network.
- Link state algorithm: Each router has full visibility of the graph, i.e., the "states" of all links.

- Q1: what algorithm runs at each node?
- Q2: what information is exchanged?

# Solution (Link State Algorithm)

1. Each router is responsible for meeting its neighbors and learning their names.
2. Each router constructs a link state packet (**LSP)** which consists of a list of names and cost to reach each of its neighbors.
3. The **LSP** is transmitted to ***ALL other routers***. Each router stores the most recently generated **LSP** from each other router.
4. Each router uses complete information on the network topology to compute the *shortest path route* to each destination node.

# Q1: The algorithm

## Dijkstra's algorithm

- Given a network graph, the algorithm computes the least cost paths from one node (source) to all other nodes.
- This can then be used to compute the forwarding (routing) table at that node.
- Iterative algorithm: maintain estimates of least costs to reach every other node. After k iterations, each node definitively knows the least cost path to k destinations.

## Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current estimate of cost of path from source to destination v
- P(v): (predecessor node) the last node before v on the path from source to v
- N': set of nodes whose least cost path is definitively known.

# A1: Dijsktra's Algorithm

Initially mark all nodes (except source) with infinite distance.

working node = source node

Sink node = destination node

While the working node is not equal to the sink

1. Mark the working node as permanent.
2. Examine all adjacent nodes in turn

   If the sum of label on working node plus distance from working node to adjacent node is less than current labeled distance on the adjacent node, this implies a shorter path. Relabel the distance on the adjacent node and label it with the node from which the probe was made.

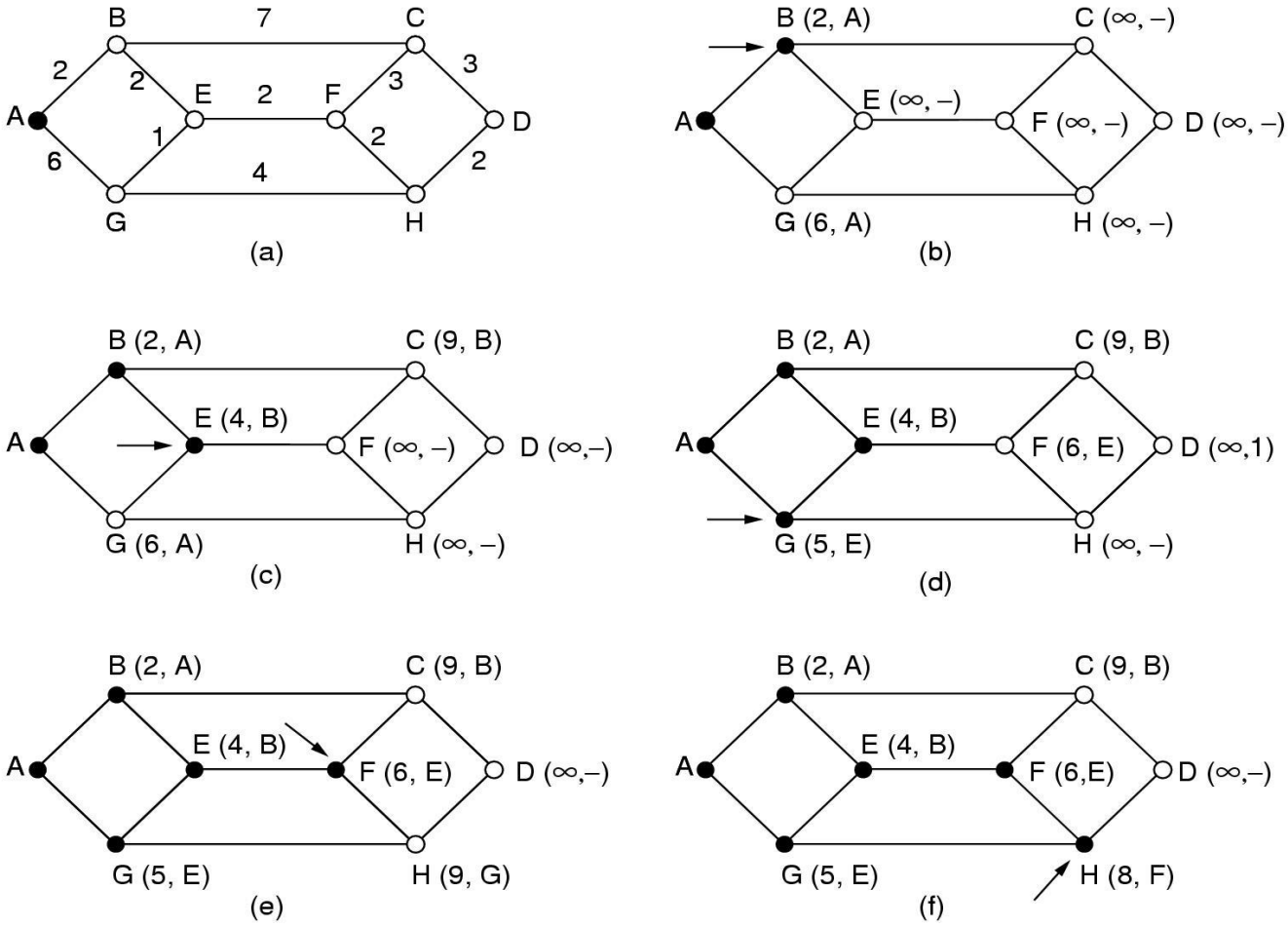
3. Examine all tentative nodes (not just adjacent nodes) and mark the node with the smallest labeled value as permanent. This node becomes the new working node.

Reconstruct the path backwards from sink to source.

# A1: Dijsktra's Algorithm

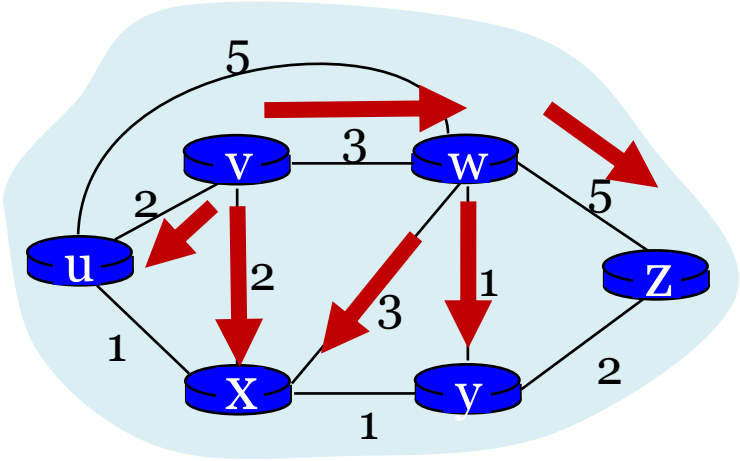
1. ***Initialization:***
2. N' = {u}
3. for all nodes v
4. if v adjacent to u
5. then D(v) = c(u,v)
6. else D(v) = ∞
7.
8. ***Loop***
9. find w not in N' such that D(w) is a minimum
10. add w to N'
11. update D(v) for all v adjacent to w and not in N' :
12. D(v) = min( D(v), D(w) + c(w,v) )
13. /* new cost to v is either old cost to v or known
14. shortest path cost to w plus cost from w to v */
15. ***until all nodes in N'***

# Routing: shortest path example: Source **A** and Destination **D**



# A2: Information exchange

- Link state flooding: the process by which neighborhood information of each network router is transmitted to all other routers
- Each router sends a link state advertisement (LSA) to each of its neighbors
- LSA contains the router ID, the IP prefix owned by the router, the router's neighbors, and link cost to those neighbors
- Upon receiving an LSA, a router forwards it to each of its neighbors: flooding

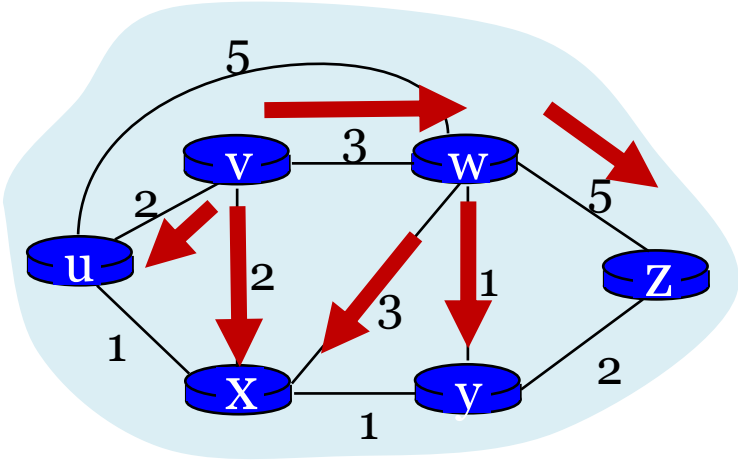


# Link State Packet

| Source Node Address (A) | |
|---|---|
| Seq. Number | |
| Age | |
| Neighbor 1 (B) | Cost 1 |
| Neighbor 2 (C) | Cost 2 |
| …. | ……. |

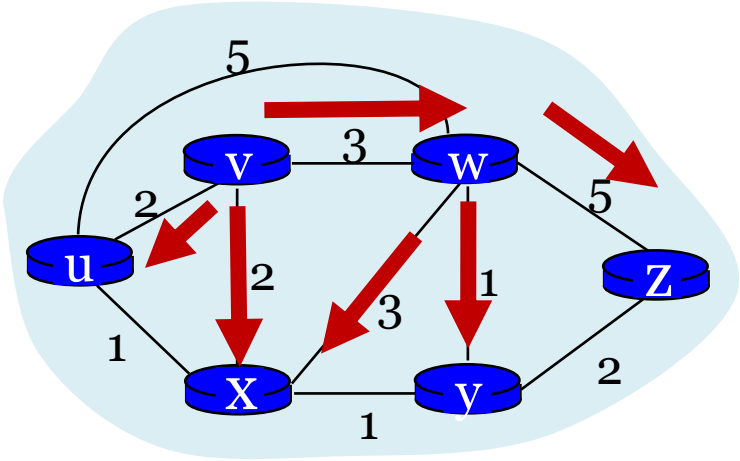
# A2: Information exchange

- Eventually, the entire network receives LSAs originated by each router.
- LSAs occur periodically and whenever the graph changes.
- **Example: if a link fails or if a new link or router is added etc..**
- The routing algorithm running at each router can use the entire network's graph to compute least cost paths



# A2: Information exchange

- Eventually, the entire network receives LSAs originated by each router.
- LSAs occur periodically and whenever the graph changes.
- **Example: if a link fails or if a new link or router is added etc..**
- The routing algorithm running at each router can use the entire network's graph to compute least cost paths



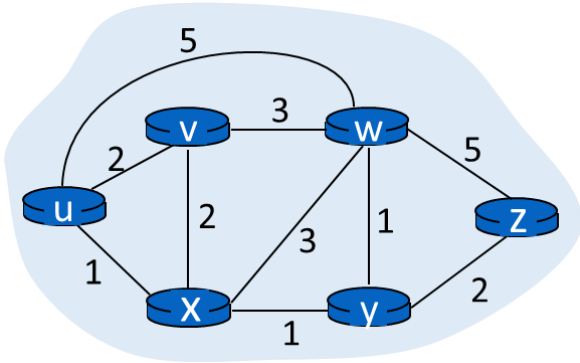
# Creating the Forwarding (Routing) Table

- After Dijkstra's algorithm, each node know the shortest part to every destination node.
- To find the router port to use for a given destination (router), find the predecessor of the node iteratively until reaching an immediate neighbor of the source `u`.
- The port connecting `u` to this neighbor will become the output port for this destination.

# Creating the Forwarding (Routing) Table

- Suppose we want forwarding entry for z.

| D(v),p(v) | D(w),p(w) | D(x),p(x) | D(y),p(y) | D(z),p(z) |
|---|---|---|---|---|
| 2,u | 3,y | 1,u | 2,x | 4,y |




Forwarding table at u:

| destination | link |
|---|---|
| z | (u,x) |

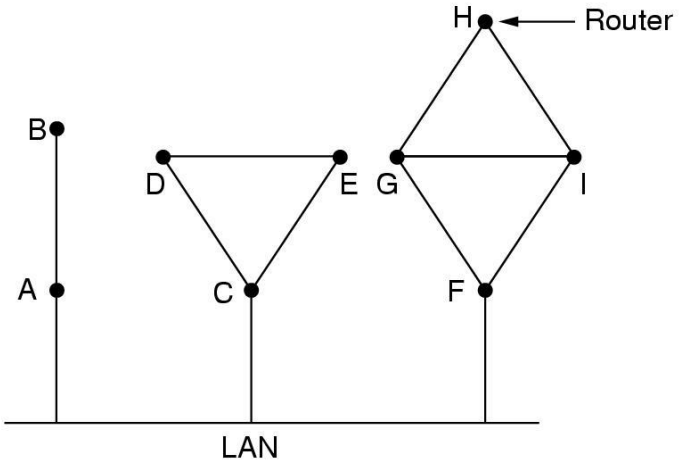
z: p(z) = y
y: p(y) = x
x: p(x) = u
x is an immediate neighbor of u

# Link State Algorithm

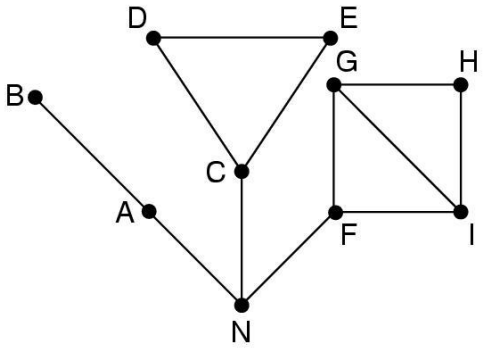
- Each router must
  - Discover its **neighbours** and **learn** their network addresses
  - Measure the delay or **cost** to each of its neighbours
  - Construct a **packet** with these distances
  - Send this packet to **all** other routers
  - Construct a local topology of the whole network based on the LSPs received.
  - Compute the **shortest path** to every other router

# Link State Algorithm

- Learning about neighbours:
  - Upon boot of router
    - Send HELLO packet on each point-to-point line
    - Routers are supposed to send reply with a globally unique name



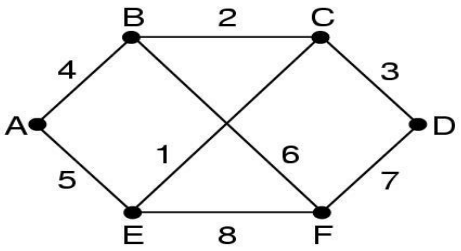

(a)




(b)

# Link State Algorithm

- Measuring line cost
  - Measure round-trip delay of HELLO Packet and its reply
- Building link state packets
  - Packet containing:
    - Identity of sender
    - Sequence number + age
    - For each neighbour: name




(a)

Link State Packets

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| Seq. | Seq. | Seq. | Seq. | Seq. | Seq. |
| Age | Age | Age | Age | Age | Age |
| B 4 | A 4 | B 2 | C 3 | A 5 | B 6 |
| E 5 | C 2 | D 3 | F 7 | C 1 | D 7 |
| | F 6 | E 1 | | F 8 | E 8 |

(b)

# Link State Algorithm

- Distributing link state packets
  - Trickiest part of algorithm
    - Arrival time for packets different
    - How to keep consistent routing tables
  - Basic algorithm
    - Flooding +
    - Sequence number (in each packet) to limit duplicates
  - Manageable problems
    - Wrap around of sequence numbers:
    - Wrong sequence number used:
      - lost in case of crash
      - Corruption

32 bits + 1 packet/sec ➔ 137 years

Age in each packet:

- Decremented during flooding, while used in router
- Age 0 ➔ info discarded

# Link State Algorithm

- Computing new routes:
  - With a full set of link state packets, a router can:
    - Construct the entire subnet graph
    - Run Dijkstra's algorithm to compute the shortest path to each destination
  - Problems for large subnets
    - Memory to store data
    - Compute time

# Open Shortest Path First (OSPF)

- OSPF runs *on top of* IP, i.e., an OSPF packet is transmitted with IP data packet header.
- Uses Level 1 and Level 2 routers
- Has: backbone routers, area border routers, and AS boundary routers
- LSPs referred to as **LSAs (Link State Advertisements)**
- Complex algorithm due to **five** distinct LSA types.

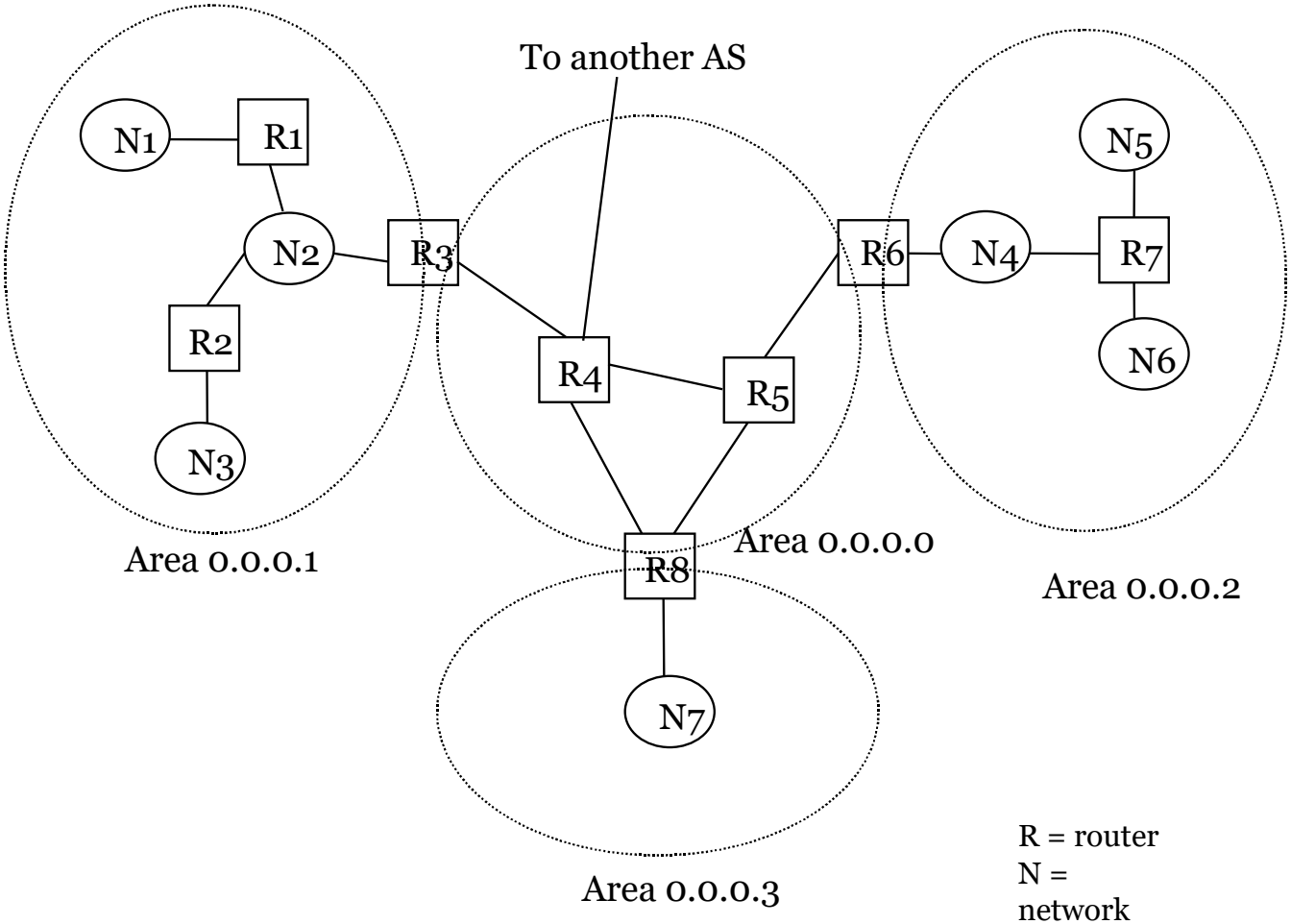
# five LSA types used by OSPF

- **Router LSA (Type 1) –** Contains a list of all links local to the router, and the status and "cost" of those links. Type 1 LSAs are generated by all routers in OSPF, and are flooded to all other routers within the local area.
- **Network LSA (Type 2) –** Generated by all Designated Routers in OSPF, and contains a list of all routers attached to the Designated Router.
- **Network Summary LSA (Type 3)** – Generated by all ABRs in OSPF, and contains a list of all destination networks within an area. Type 3 LSAs are sent between areas to allow inter-area communication to occur.

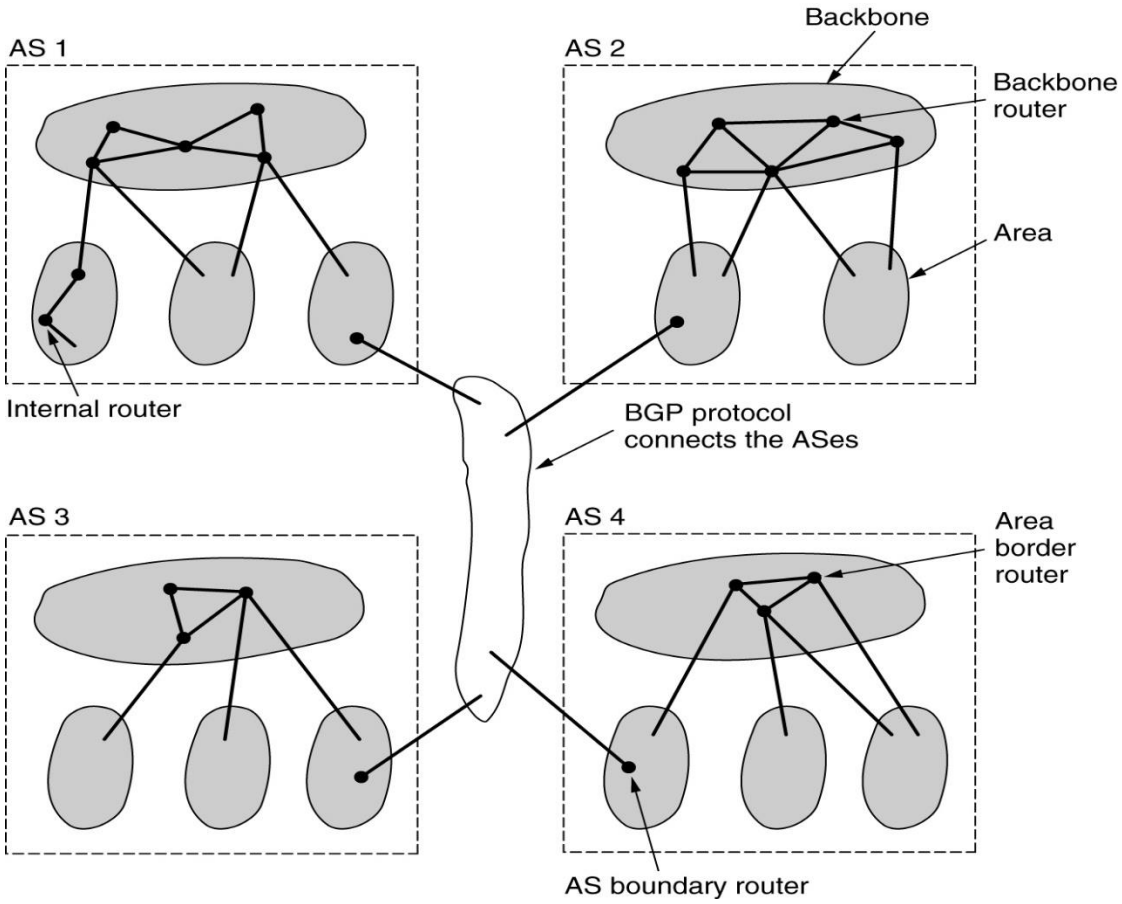
# **Five** LSA types used by OSPF

- **ASBR Summary LSA (Type 4)** – Generated by ABRs in OSPF, and contains a *route* to any ASBRs in the OSPF system. Type 4 LSAs are sent from an ABR into its local area, so that Internal routers know how to exit the Autonomous System.
- **External LSA (Type 5)** – Generated by ASBRs in OSPF, and contain routes to destination networks *outside* the local Autonomous System. Type 5 LSAs can also take the form of a **default route** to all networks outside the local AS. Type 5 LSAs are flooded to all areas in the OSPF system.

# OSPF Areas


To another AS
N1
R1
N2
R3
R2
N3
R4
R5
R6
N4
R7
N5
N6
R8
N7
Area 0.0.0.1
Area 0.0.0.0
Area 0.0.0.2
Area 0.0.0.3
R = router
N = network

# OSPF




The relation between ASes, backbones, and areas in OSPF.

# OSPF details

- The OSPF process builds and maintains three separate tables:
  - A **neighbor table** – contains a list of all neighboring routers.
  - A **topology table** – contains a list of *all* possible routes to all known networks within an area.
  - A **routing table** – contains the *best* route for each known network.

- Other characteristics of OSPF include:
  - OSPF supports only IP routing.
  - OSPF routes have an administrative distance is **110**.
  - OSPF uses **cost** as its metric, which is computed based on the bandwidth of the link. OSPF has no hop-count limit.

# Summary of link state protocols

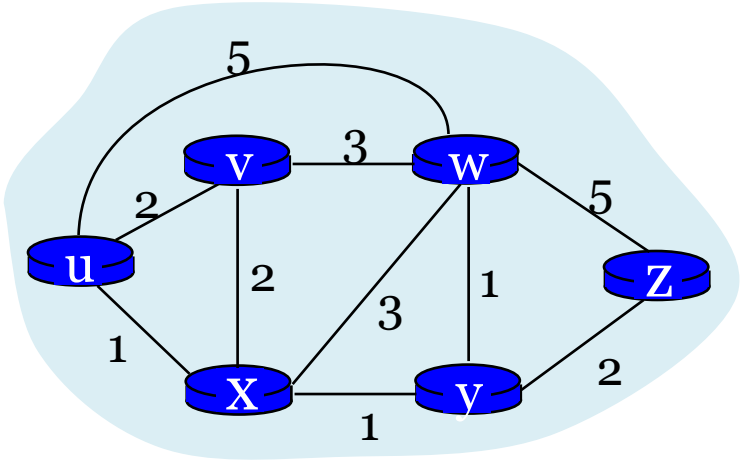
- Each router announces link state to the entire network using flooding.
- Each node independently computes least cost paths to every other node using the full network graph.
- Dijkstra's algorithm can efficiently compute these best paths.
  - Easy to populate the forwarding table from predecessor information computed during the algorithm.
  - Example of link state routing protocol is Open Shortest First Protocol (OSPF).

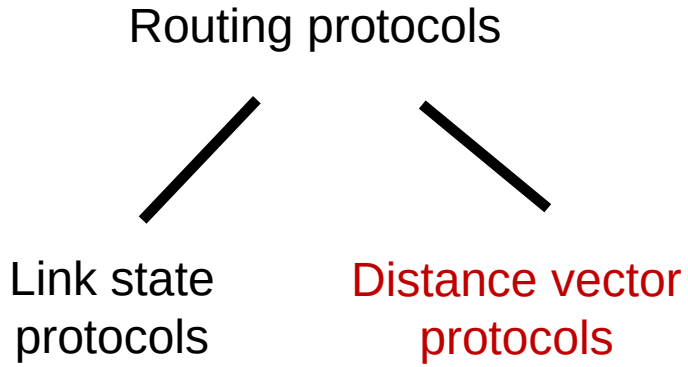
# Distance Vector Routing Protocols

RIP, RIPng

# Distance Vector Algorithm

- Distributed routing protocols.
- Goal of routing algorithms: find least cost path in a graph abstraction of the network.
- DV protocol: Each router maintains a vector of distances to all other routers; not the graph.
- Q1: what algorithm runs at each node?
- Q2: what information is exchanged?



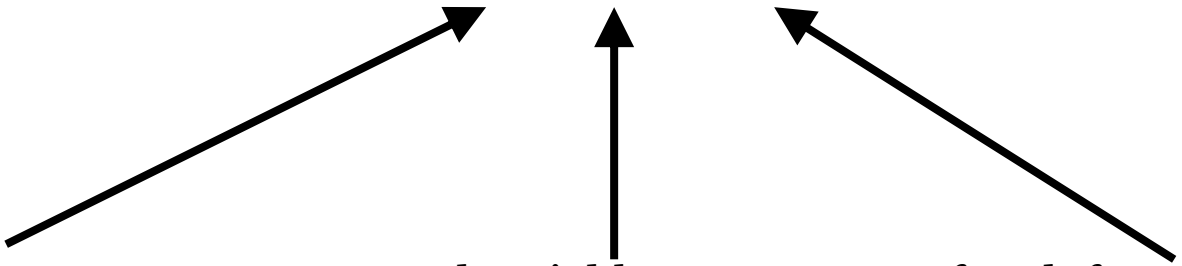



# A1: Algorithm

Bellman-Ford algorithm

- Each node initializes its own distance vector (DV) to edge costs.
- Each node sends its DVs to its neighbors.
- When a node x receives new DV from a neighbor v, it updates its own DV using the Bellman-Ford equation as follows:
- Given $d_x(y)$ := estimated cost of the least-cost path from x to y

Update $d_x(y) = \min_v \{c(x,v) + d_v(y)\}$



minimum taken over all neighbors v of x

cost to reach neighbor v directly from x

cost of path from neighbor v to destination y

# The Algorithm

- Historically known as the ***old*** ARPANET routing algorithm {or known as *Bellman-Ford algorithm*}.
- Basic idea: each network node maintains a Distance Vector table containing the ***distance*** between itself and **ALL** possible destination nodes.
- Distances are based on a chosen metric and are computed using information from the **neighbors'** distance vectors.
- Metric: *usually number of hops or delay.*

# A2: Exchanged info = Distance Vectors

- Nodes exchange distance vectors with their neighbors.
  - No flooding unlike link state protocols. Message not propagated further than the neighbors.
- $D_x(y)$ = estimate of least cost from x to y.
- Distance vector: $\mathbf{D}_x = [D_x(y): y \in N ]$.
- Node x knows cost of edge to each neighbor v: c(x,v).
- Node x maintains $\mathbf{D}_x$.
- Node x also maintains its neighbors' distance vectors.
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N ]$.
- Nodes exchange distance vector periodically and whenever the local distance vector changes (e.g., link failure, cost changes).

# Distance Vector Routing

Information kept by DV router

1. each router has an ID
2. associated with each link connected to a router, there is a link cost (static or dynamic) **the metric issue!**

- Distance Vector Table Initialization
  - Distance to itself = 0
  - Distance to ALL other routers = infinity number
  - Suppose a network have following nodes: A, B, C, D, E, F .

| Distance Vector of Node A (Initially) | |
|---|---|
| A | 0 |
| B | ∞ |
| C | ∞ |
| D | ∞ |
| E | ∞ |
| F | ∞ |

# Distance Vector Routing

1. Router transmits its *distance vector* to each of its neighbors.
2. Each router receives and saves the most recently received *distance vector* from each of its neighbors.
3. A router **recalculates** its distance vector when:
   a. It receives a *distance vector* from a neighbor containing different information than before.
   b. It discovers that a link to a neighbor has gone down (i.e., a topology change).

Note: The DV calculation is based on minimizing the cost to each destination.

# Features of DVR protocols

- Distributed algorithm
- Triggers when:
  - Change in cost to neighbour
  - Receive new table from neighbour
  - Periodic update
- Update local tables
- If changed:  forward distance vector to neighbours

# D V Algorithm

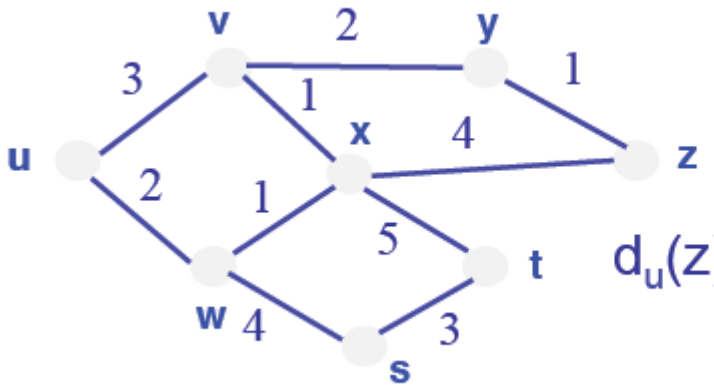
- Define distances at each node *X*

$$d_x(y) = \text{cost of least-cost } \textit{path} \text{ from } X \text{ to } Y$$

- Update distances based on neighbor's distance vector.

$$d_x(y) = \min \{c(x,v) + d_v(y)\}$$

over all neighbors *V*

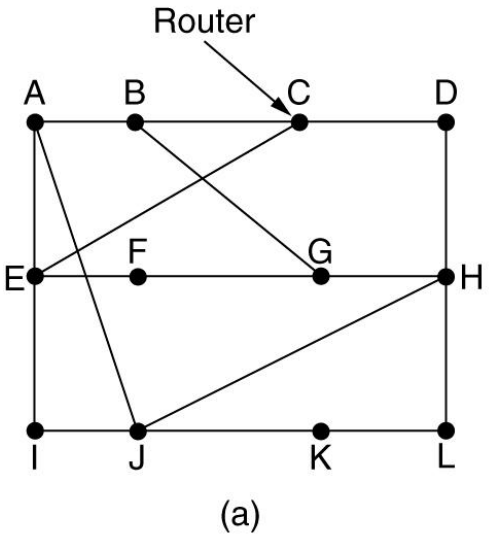



$$d_u(z) = \min\{c(u,v) + d_v(z), c(u,w) + d_w(z)\}$$

# D V Algorithm (Contd.)

- $c(x,v)$ = 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 $\boldsymbol{D}_x = [D_x(y) : y \varepsilon N]$
- Whenever node $x$ receives distance vectors from its neighbors'
  - For each neighbor $v$, $x$ maintains $\boldsymbol{D}_v = [D_v(y): y \varepsilon N]$
- 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 $y \varepsilon N$

# Distance Vector Routing




(a)

New estimated delay from J

| To | A | I | H | K | 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

Vectors received from J's four neighbors

New routing table for J

(b)

(a) A subnet.

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

# Initial Distance Vector of Each Node

| u | 0 |
|---|---|
| v | 2 |
| w | 5 |
| X | 1 |
| y | ∞ |
| Z | ∞ |

DV of Node u

| u | 2 |
|---|---|
| v | 0 |
| w | 3 |
| X | 2 |
| y | ∞ |
| Z | ∞ |

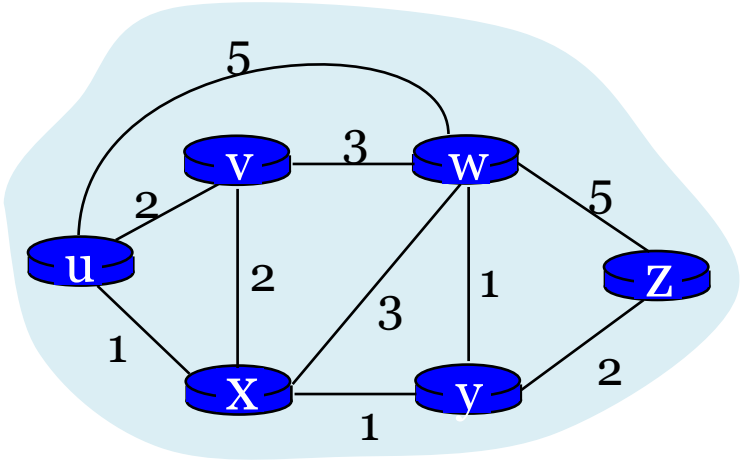
DV of Node v

| u | 5 |
|---|---|
| v | 3 |
| w | 0 |
| X | 3 |
| y | 1 |
| Z | 5 |

DV of Node w

| u | 1 |
|---|---|
| v | 2 |
| w | 3 |
| X | 0 |
| y | 1 |
| Z | ∞ |

DV of Node X




| u | ∞ |
|---|---|
| v | ∞ |
| w | 1 |
| X | 1 |
| y | 0 |
| Z | 2 |

DV of Node y

| u | ∞ |
|---|---|
| v | ∞ |
| w | 6 |
| X | ∞ |
| y | 2 |
| Z | 0 |

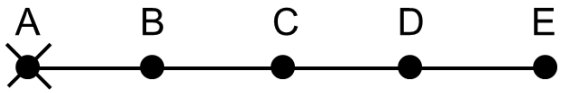
DV of Node Z

# Advantage of distance vector protocols

- Good news converge very fast for example:
  - Suppose the link cost reduces or a new better path becomes available in a network.
  - The immediate neighbors of the change detect the better path immediately.
  - Since their DV changed, these nodes notify their neighbors immediately.
    - And those neighbors notify still more neighbors.
    - … until the entire network knows to use the better path.
- Good news travels fast through the network.
- This is despite messages only being exchanged among neighbors.

# Disadvantage of distance vector protocols

- Routing loop may present in the network.
- Bad news may take infinite time to converge. For example:




| | | | | | |
|---|---|---|---|---|---|
| 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 | etc… to infinity |

Count to infinity problem

# Disadvantage of distance vector protocols

- Reacting appropriately to bad news requires information that only other routers have.
- B needs to know that C has no other path to A other than via B.
- Solutions:
  - Split Horizon: if X gets its route to Y via Z, then X will not announce $d_X(Y)$ in its message to Z
  - Poisoned reverse: if X gets its route to Y via Z, then X will announce $d_X(Y) = \infty$ in its message to Z
    - Effect: Z won't use X to route to Y
- However, this won't solve the problem in general (think why.)
- Fundamentally, DV protocols must exchange more information to react robustly to network changes and router errors.

# Routing: distance vector

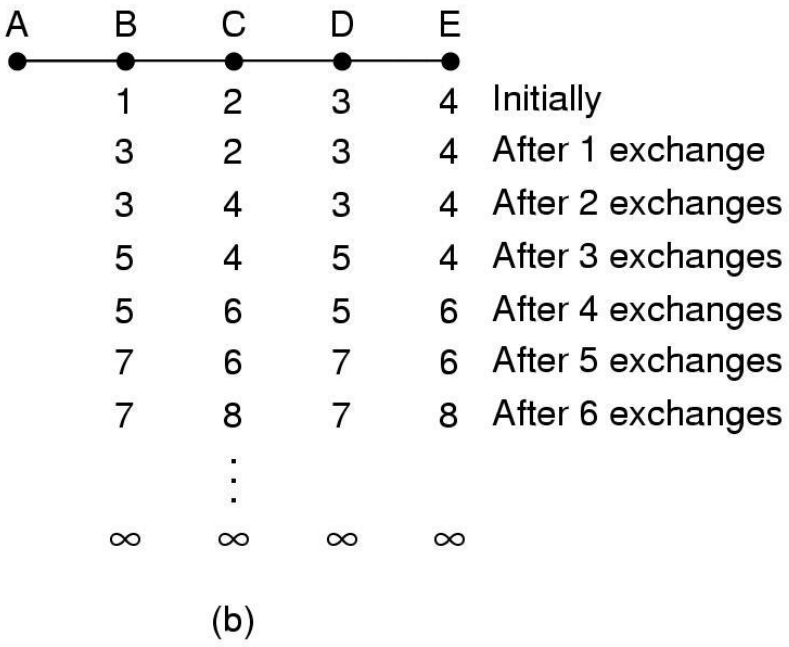
Bad news:

- A goes down

Loops!!

Slow!!

∞ = 5?



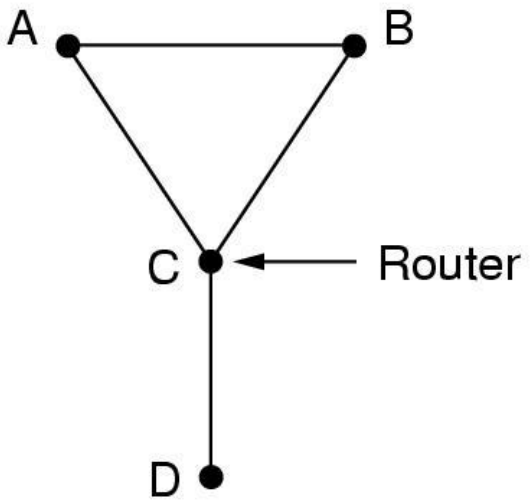

(b)

# Mitigation Strategies

- Hold downs
  - As metric increases, delay propagating information
  - Limitation: Delays convergence
- Loop avoidance
  - Full path information in route advertisement
- Split horizon
  - Never advertise a destination through its next hop
  - » C doesn't advertise B cost from C to A
- Poison reverse: Send negative information when advertising a destination through its next hop
  - C advertises B the cost of C to A with a metric of ∞
  - Limitation: Only works for loop's of size 2

# Routing: distance vector

- Split horizon hack:
  - Hack ➔ does not always work
- Example:
  - D goes down
  - A and B lie to C
  - A offers to B route to D
  - B offers to A route to D
  - Loops again!!!



# Routing Information Protocol (RIP)

- RIP had widespread use because it was distributed with BSD Unix in *"routed", a router management daemon*.
- **RIP** is the most used Distance Vector Protocol.
- RFC1058 in June 1988.
- Sends packets every 30 seconds or faster.
- Runs over UDP.
- Metric = hop count
- BIG problem is max. hop count =16

  ➔ RIP limited to running on small networks!!
- Upgraded to RIPv2

# Solution (Link State Algorithm)

1. Each router is responsible for meeting its neighbors and learning their names.
2. Each router constructs a link state packet (**LSP)** which consists of a list of names and cost to reach each of its neighbors.
3. The **LSP** is transmitted to ***ALL other routers***. Each router stores the most recently generated **LSP** from each other router.
4. Each router uses complete information on the network topology to compute the *shortest path route* to each destination node.

# Comparison of LS and DV

| Link State Algorithms | Distance Vector Algorithms |
| --- | --- |
| • Nodes have full visibility into the network's graph.<br>• Message complexity is high: each LSA is flooded over the whole network.<br>• In general, robust to network changes. Scope of incorrect info is limited to bad LSAs.<br>Deployed routing protocols:<br>1. OSPF (Open Shortest Path First).<br>2. IS-IS: (Intermediate System to Intermediate System). | • Only distances and neighbors are visible.<br>• Message complexity is low: DVs are exchanged among neighbors only.<br>• Brittle against bad news and router bugs: incorrect info can propagate throughout a network.<br>Deployed routing protocols:<br>1. RIP: Routing Information Protocol.<br>2. IGRP: Interior Gateway Routing Protocol<br>3. RIPng (for IPv6). |