# Computer Networks
# COL 334/672

Intra-domain Routing

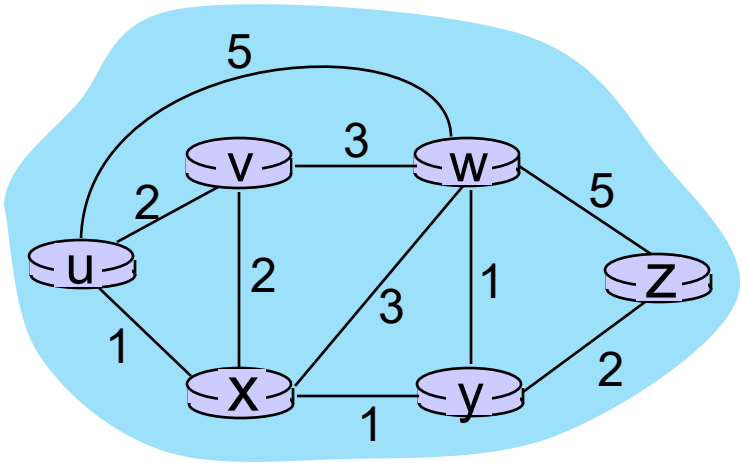
*Slides adapted from KR*

Sem 1, 2025-26

# Intra-domain Routing

How to determine shortest path from one node to all other nodes in a graph?

Graph Abstraction




graph: $G = (N,E)$

$N$: set of routers = { $u, v, w, x, y, z$ }

$E$: set of links ={ $(u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)$ }

$c_{a,b}$: cost of *direct* link connecting $a$ and $b$

e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

→ Bellman Ford

Floyd Warshall,

→ Dijkstra

→ graph is dynamic
  ↳ node can come & go
  ↳ edge can
  ↳ edge weight can change

→ Algorithm should be simple & efficient

→ Scalable ⇒ Decentralized

# Distance vector algorithm

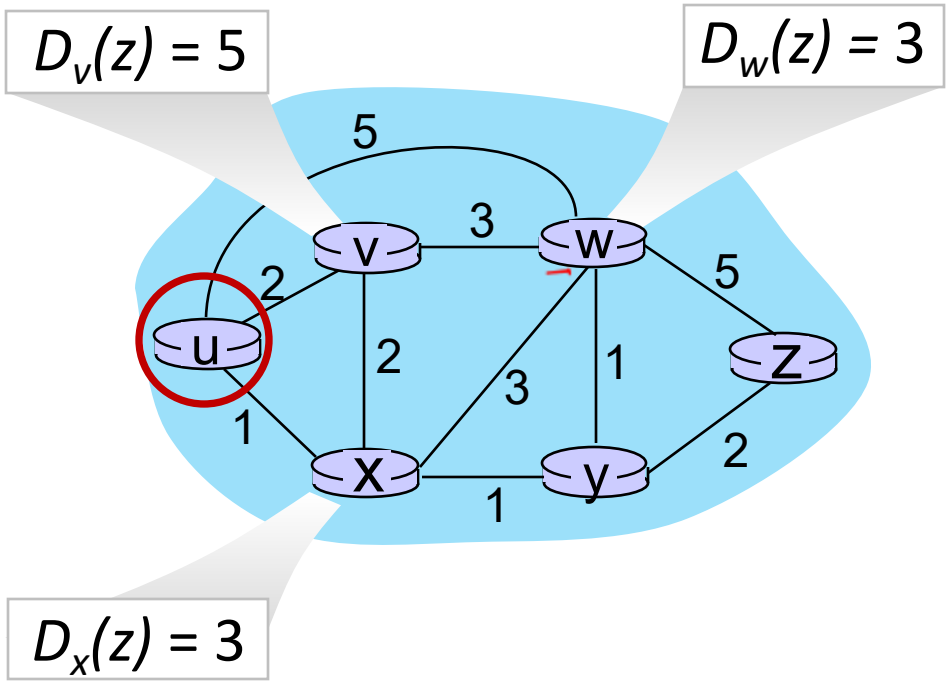
known:

Based on *Bellman-Ford* (BF) equation (dynamic programming):

$$D_x(y) = \min_v \{c(x,v) + D_v(y)\}$$

v: all neighbors of x

Suppose that *u*'s neighboring nodes, *x,v,w,* know that for destination *z*:




$$\min\{c(u,v) + D_v(z),\ c(u,x) + D_x(z),\ c(u,w) + D_w(z)\}$$

$$= \min\{2+5,\ 1+3,\ 5+3\} = 4$$

# Distance vector algorithm

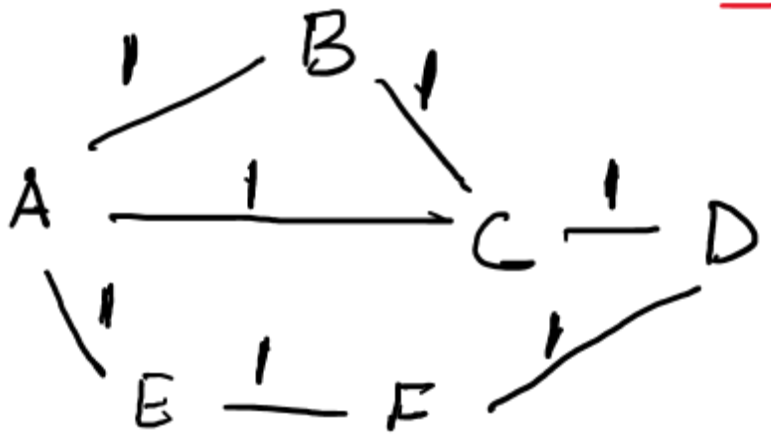
key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when *x* receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow min_v\{c_{x,v} + D_v(y)\} \text{ for each node } y \in N$$

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

# Distance vector: Example




It #1

A sends: (A,0), (B,1), (C,1), (E,1)

Hears: B: (B,0), (A,1), (C,1)

C: (C,0), (A,1), (D,1), (B,1)

E: (E,0), (A,1), (F,1)

A sends: (A,0), (B,1), (C,1), (E,1), (D,2), (F,2)

Hears: B: (B,0), (A,1), (C,1), (E,2), (D,2)

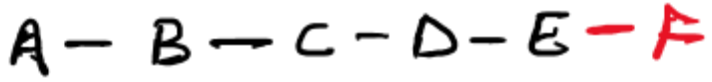
C: (C,0), (A,1), (B,1), (D,1), (F,2), (B,2)

E: (E,0), (A,1), (B,2), (F,1), (D,2), (C,2)

| DST | COST | NXT HOP |
|---|---|---|
| A | 0 | A |
| B | 1 | B |
| C | 1 | C |
| D | 2 | C |
| E | 1 | E |
| F | 2 | E |

# Good News Travels Fast, Bad News Travels Slow!

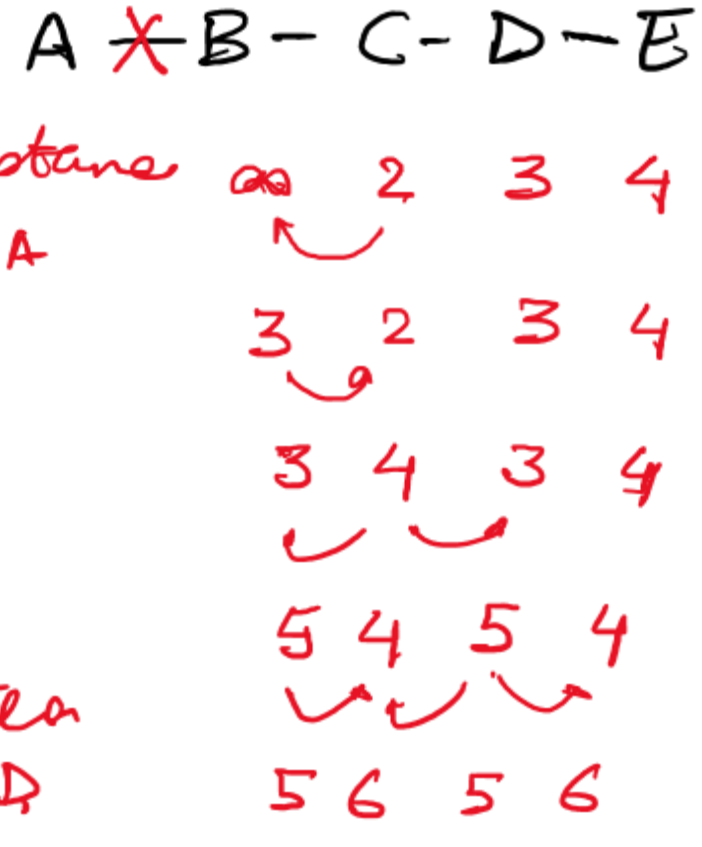
- Assume a new node F comes up in the network
- How long does it take for the A to update their routing table?

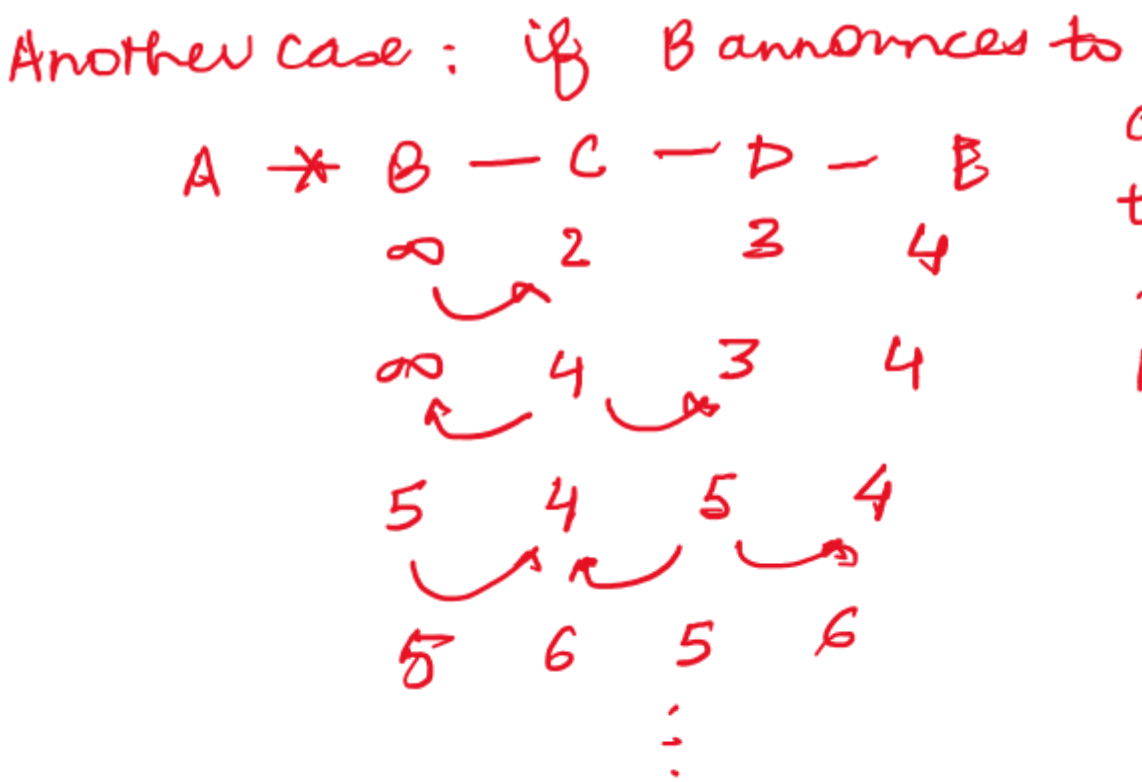



- In general, good news spreads at rate of one hop per exchange
  - What is the maximum time it can take?
- What happens in case of link failure?

# Good News Travels Fast, Bad News Travels Slow!

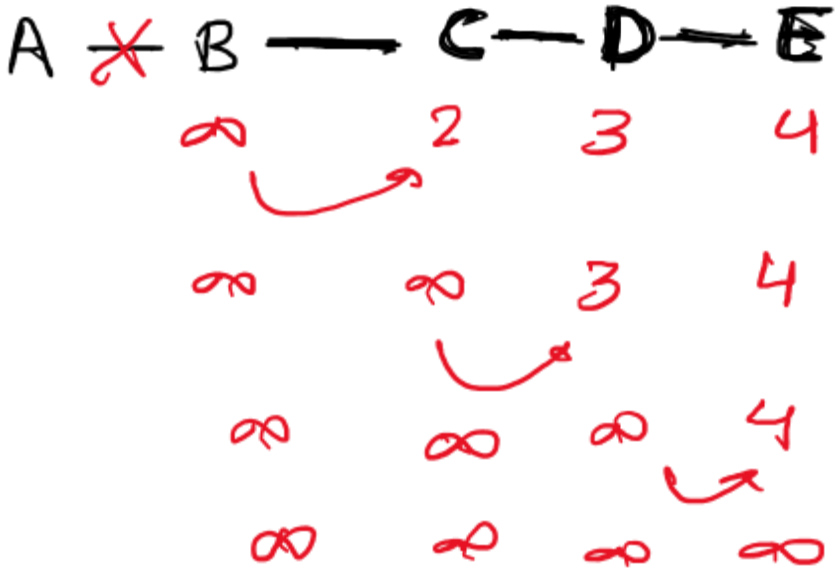
- Assume the A-B link goes down
- How does the routing table gets updated at B?
- At other nodes?



# How to Handle *Count to Infinity* Problem?

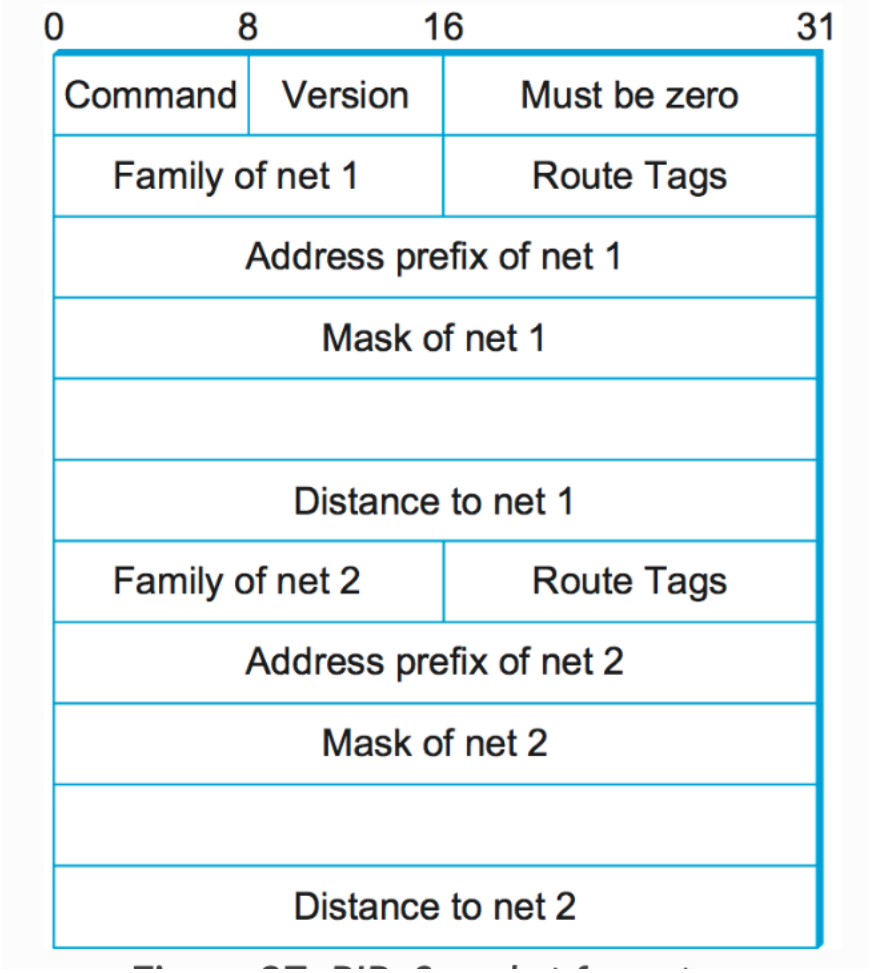
- Split horizon
  - Don’t announce the route back to the next hop
  - Does it always work?



# How to Handle *Count to Infinity P*roblem?

- Split horizon
  - Don’t announce the route back to the next hop
- Make infinity smaller
  - Routing Information Protocol (RIP): Based on hop count
  - Used a maximum length of 15
  - **Limitation**: Doesn’t work for cases when actual path length is greater than 15

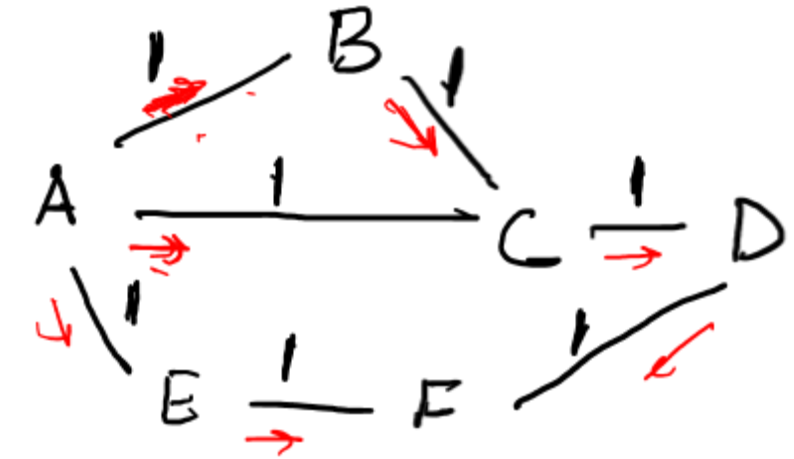
# RIP Message Format




RIPv2 Protocol Message Format

# Link State Routing Algorithm

- Each node knows the entire network topology including link costs:
  - accomplished via "link state broadcast"
  - all nodes have same info
- Link state broadcast:
  - (x, {list of neighbors, cost}, seq #, TTL)
  - reliable broadcast using acknowledgements
- Once a node has the entire topology:
  - Use Dijkstra's algorithm to find the shortest path




{A, ([B, 1), (B, 1), (C, 1)], 1001, 60}

Seq #: assess the liveliness of msg
→ only forward LSBs of A if seq # > cur seq # (A)

# Dijkstra's link-state routing algorithm

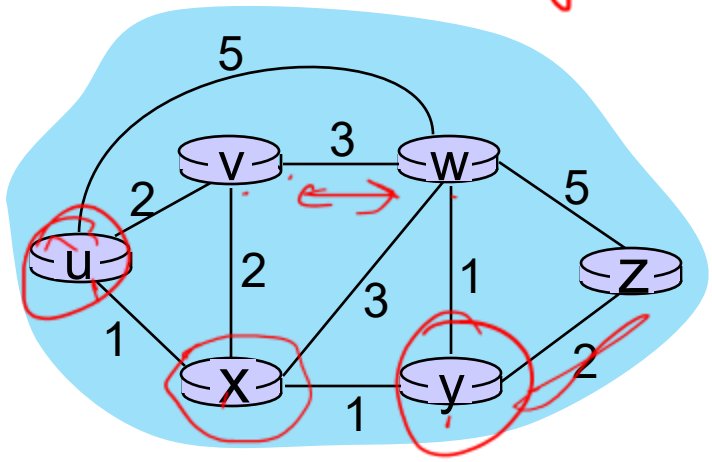
```
1  Initialization:
2    N' = {u}                    /* compute least cost path from u to all other nodes */
3    for all nodes v
4      if v adjacent to u        /* u initially knows direct-path-cost only to  direct neighbors  */
5        then D(v) = c_u,v       /* but may not be minimum cost!                                   */
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 least-path-cost to v is either old least-cost-path to v or known
14     least-cost-path to w plus direct-cost from w to v */
15   until all nodes in N'
```

# Dijkstra's algorithm: an example

LSA: periodic
Triggered if update in N/w

predecessor of v

| Step | N' | v<br>D(v),p(v) | w<br>D(w),p(w) | x<br>D(x),p(x) | y<br>D(y),p(y) | z<br>D(z),p(z) |
|---|---|---|---|---|---|---|
| 0 | u | 2, u | 5, u | 1, u | ∞ | ∞ |
| 1 | ux | 2, u | 4, x | | 2, x | ∞ |
| 2 | uxv | | 3, v | | 2, x | ∞ |
| 3 | uxvy | | 3, v | | | 4, y |
| 4 | uxvyw | | | | | 4, y |
| 5 | uxvywz | | | | | |




After adding node a to N'
∀ neighbors b of a not in N'

$$D(b) = \min\left(D(b),\ D(a) + c(a,b)\right)$$

Two popular link state routing protocols: Intermediate System to Intermediate System (IS-IS), Open Shortest Path First (OSPF)

# Setting Link Weights

- How to set link weights?
  - Static: 1, bandwidth of link, cost of the link
  - Adaptive: load on the network
- How to set weights based on load on the network?
  - Send ECHO packets to measure RTT
  - RTT depends on queuing delays (load on the network)
- Is it a good strategy?
  - Can lead to path oscillations, routing loops
  - Variations in end-to-end delay and packet reordering
- Link weight setting is an art!
  - Knobs for traffic engineering
  - Reversal of cause and effect