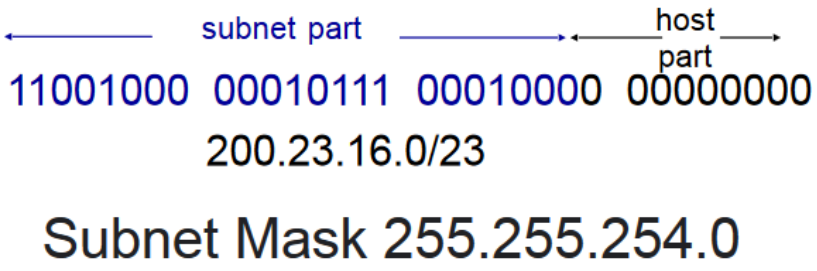
9/15 – Networking Layer pt. II

IP Addressing: CIDR

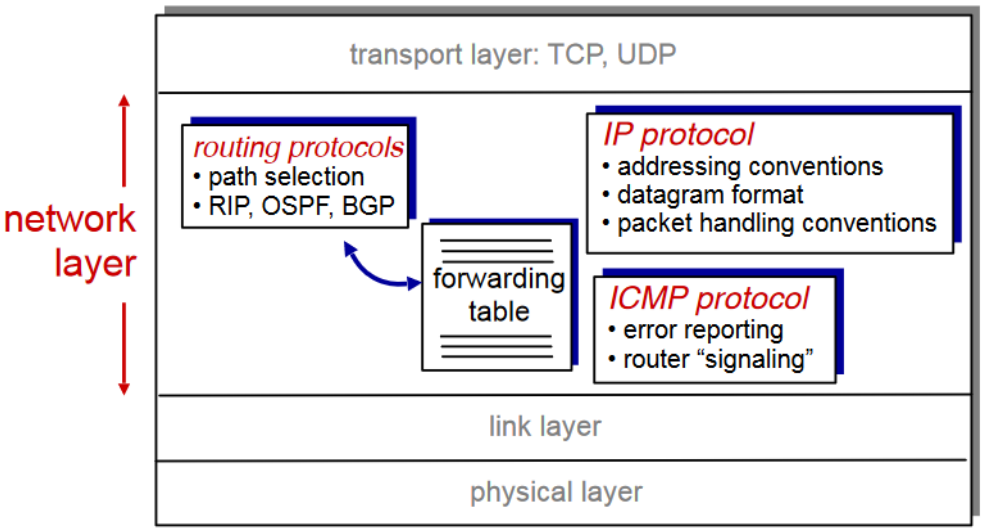
CIDR: Classless InterDomain Routing (pronounced “cider”)

- Subnet portion of address of arbitrary length
- Address format: a.b.c.d/x, where is # of bits in subnet portion of address



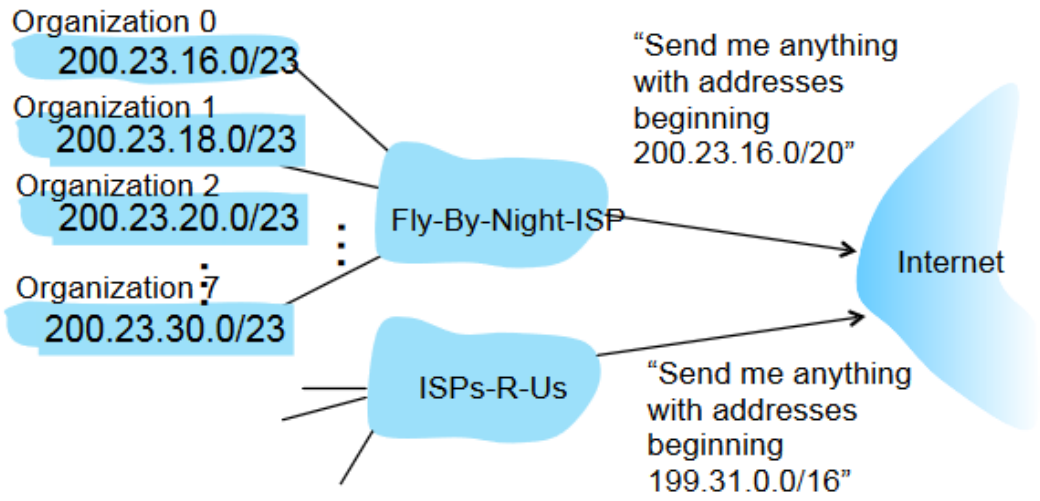
The Internet network layer:

- Contains host, router network layer functions

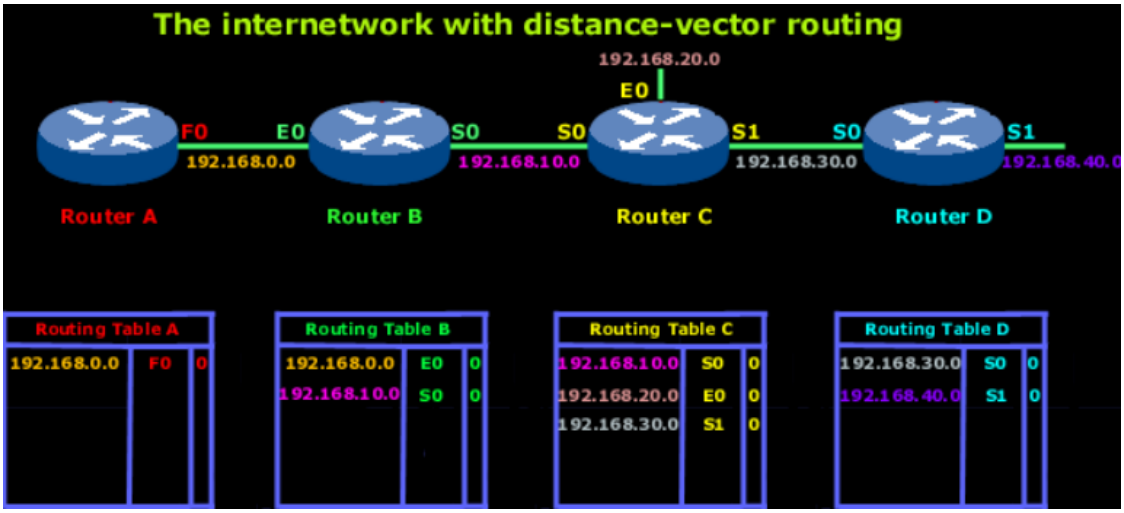


Hierarchical addressing: route aggregation

- Hierarchical addressing allows efficient advertisement of routing information



- Next, back to the problem of distance-vector algorithm
- i.e. IANA -> ARIN -> COMCAST -> UVA -> ME



- in actuality, tables would be populated with more values

Distance Vector Algorithm:

- Bellman-Ford equation (dynamic programming)

Let $d_x(y) = cost\ of\ least - cost\ path\ from\ x\ to\ y$

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

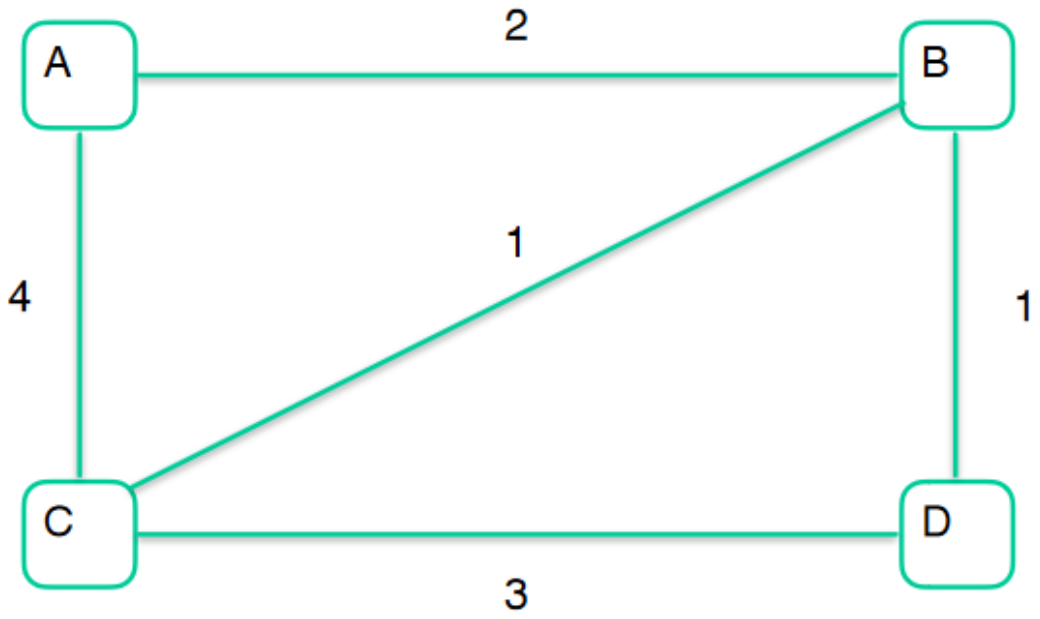
- $c(x,v) = cost\ to\ neighbor\ v$
- $d_v(y) = cost\ from\ neighbor\ v\ to\ destination\ y$

- Each node knows the cost to each neighbor, and must maintain its neighbors' distance vectors

key idea:

- From time to time, each node sends its own distance vector estimate to neighbors
- When x receives a new distance vector estimate from neighbor, it updates its own distance vector using the Bell-Ford equation
- The "estimate" $D_x(y)$ converges to the actual value $d_x(y)$

Example: network with 4 nodes



- First, initialize the cost of neighbors

Node A table:

	A	B	C	D
A	0	2	4	∞
B	∞	0	∞	∞
C	∞	∞	0	∞
D	∞	∞	∞	0

Node B table:

	A	B	C	D
A	0	∞	∞	∞
B	2	0	1	1
C	∞	∞	0	∞
D	∞	∞	∞	0

Node C table:

Node D table:

	A	B	C	D
A	0	∞	∞	∞
B	∞	0	∞	∞
C	4	1	0	3
D	∞	∞	∞	0

	A	B	C	D
A	0	∞	∞	∞
B	∞	0	∞	∞
C	∞	∞	0	∞
D	∞	1	3	0

- Next, tell your neighbors about your distance vector

Updated Node B table (with information from Node A):

	A	B	C	D
A	0	2	4	∞
B	2	0	1	1
C	∞	∞	0	∞
D	∞	∞	∞	0

- Next, process the information and update your distance vector
- For node B, there are no updates to make b/c it already has the shortest path to each node
    - However, for node A, there are updates to make after it has received information from node B

Updated Node A table:

	A	B	C	D
A	0	2	3	3
B	2	0	1	1
C	∞	∞	0	∞
D	∞	∞	∞	0

Example Calculation / Update of table (x = A, y = D):

$$D_a(c) \leftarrow \min \{c(a,c) + D_c(d), \; c(a,b) + D_b(d)\}$$

$$D_a(c) \leftarrow \min \{3 + \infty, \; 2 + 1\} = 3$$

- Because A's distance vector was changed, it is going to get sent to other nodes in the network

- This process is repeated until all nodes have their tables completed with updated values

Final node A table:

	A	B	C	D
A	0	2	3	3
B	2	0	1	1
C	3	1	0	2
D	∞	∞	∞	0

Final node B table:

	A	B	C	D
A	0	2	3	3
B	2	0	1	1
C	3	1	0	2
D	3	1	2	0

Final node C table:

	A	B	C	D
A	0	2	3	3
B	2	0	1	1
C	3	1	0	2
D	3	1	2	0

Final node D table:

	A	B	C	D
A	0	∞	∞	∞
B	2	0	1	1
C	3	1	0	2
D	3	1	2	0

But how does this relate to forwarding tables?

- This process helps to figure out what interface to forward on, since each of these routers has a subnet

How do I determine my next hop router?

- If we focus on a single column, we can determine where the routing table will send the next hop

	D
A	3
B	1
C	2
D	0

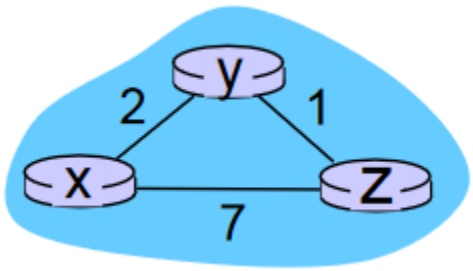
Can't send to my self

+2 = 3

+4 = 6

+ ∞ = ∞. not a neighbor can't sen

Quiz:



Node X table:

	X	Y	Z
X	0	2	7
Y	∞	0	∞
Z	∞	∞	0

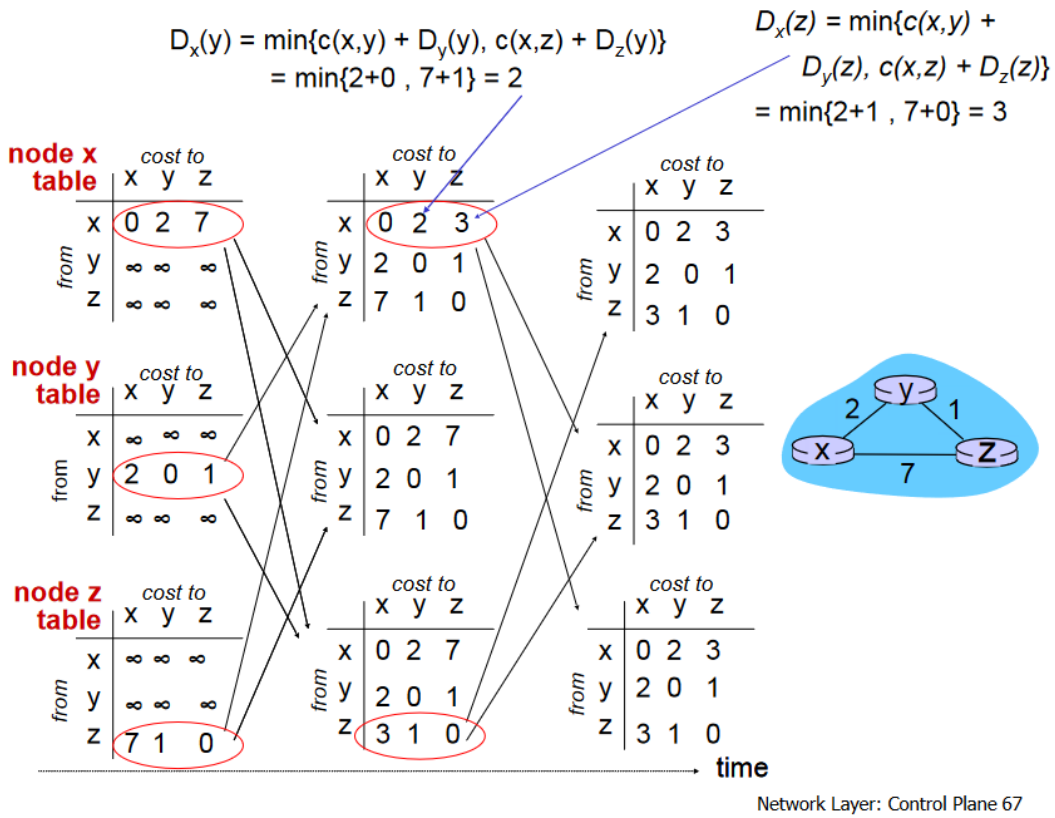
Node Y table:

	X	Y	Z
X	0	∞	∞
Y	2	0	1
Z	∞	∞	0

Node Z table:

	X	Y	Z
X	0	∞	∞
Y	∞	0	∞
Z	7	1	0

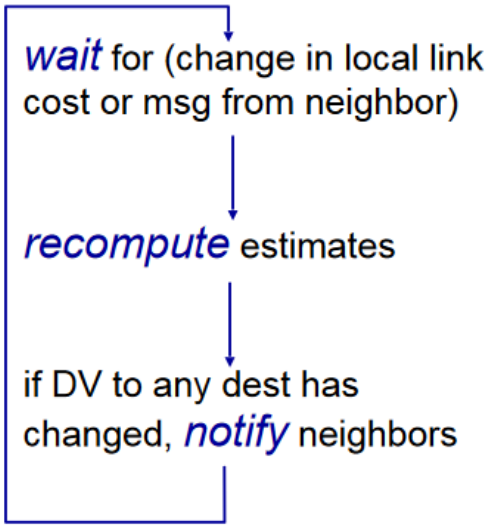
- After sharing information between neighbors, the table will update so that the distance from X to Z is 3 (smaller than 7)



Network Layer: Control Plane 67

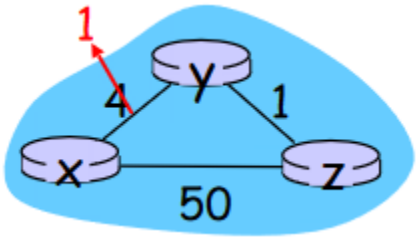
More info. on distance vector algorithm:

- Iterative, asynchronous: each local iteration caused by:
  - Local link cost change
  - Distance vector update message from neighbor
- Distributed:
  - Each node notifies neighbors only when its distance vector changes
    - Neighbors then notify their neighbors if necessary
- For each node:



  -

Link cost changes:



- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If distance vector changes, notify neighbors
    - t_0: y detects link-cost change, updates its distance vector, informs its neighbors
    - t_1: z receives update from y, updates its table, computes new least cost to x, sends its neighbors its distance vector
    - t_2: y receives z's update, updates its distance table. Y's least costs do not change, so y does not send a message to z
        - "good news travels fast"
- Node detects local link cost change
- *Bad news travels slow* – "count to infinity problem"
- 44 iterations before algorithm stabilizes: see text
- (example in slides)

Solution: poisoned reverse

- If z routes through y to get to x:
    - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
    - (example in slides)

Destination-based forwarding:

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011000 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Longest prefix matching: when looking for forwarding table entry for given destination address, use longest address prefix that matches destination address

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

- Often performed using ternary content addressable memories (TCAMs)
    - Content addressable: present address to TCAM: retrieve address in one clock cycle, regardless of table size
    - Cisco catalyst: can hold up to 1M routing table entries in TCAM

Comparison of LS (Dykstra) and DV algorithms:

- Message complexity:
    - LS: with n nodes, E links, O(nE) messages sent
    - DV: exchange between neighbors only

        - Convergence time varies
- Speed of convergence:
    - LS: O(n^2) algorithm requires O(nE) messages
        - May have oscillations
    - DV: convergence time varies
        - May be routing loops
        - Count-to-infinity problem
- Robustness: what happens if router malfunctions?
    - LS:
        - Node can advertise incorrect link cost
        - Each node computes only its own table
    - DV:
        - DV node can advertise incorrect path cost
        - Each node's table used by others
            - Error propagates through network

Making routing scalable:

- Our routing study thus far – idealized
    - All routers identical
    - Network "flat"
        - …. not true in practice
    - Scale: with billions of destinations
        - Can't store all destinations 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

- *DHCP example done at the end of class (unsuccessfully)