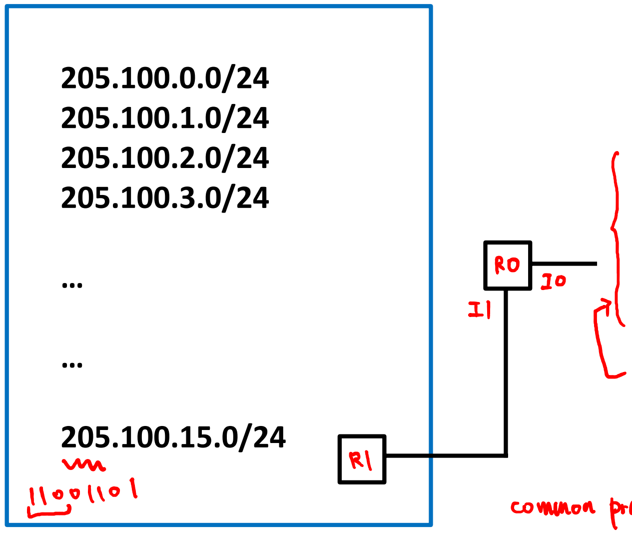


11001101

class C

Routing Table @ R0

Dest.	Mask	Next-hop	Int
205.100.0.0	/24	R1	I1
205.100.1.0	/24	R1	I1
⋮			
205.100.15.0	/24	R1	I1
205.100.0.0	/20	R1	I1

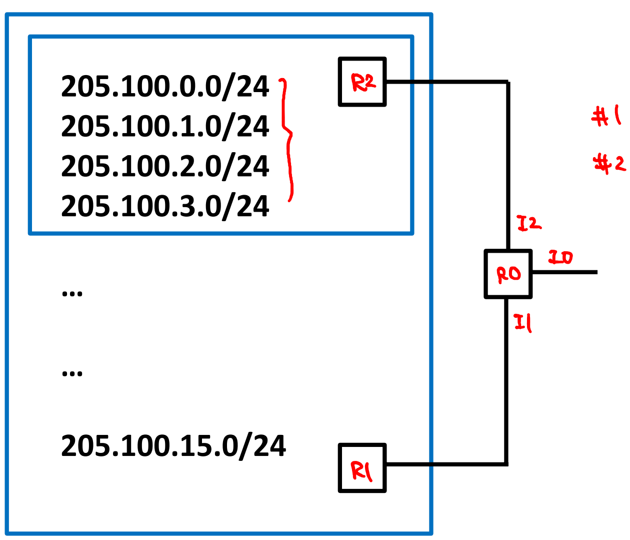
supernet

common prefix of 20 bits

205.100.0.0	0000 0000
1.0	0000 0001
2.0	0000 0010
3.0	0000 0011
⋮	⋮
15.0	0000 1111

CIDR

Classless
Inter
Domain
Routing



Routing Table @ R0 $\quad 2^{24-20} = 16$

#1	205.100.0.0	/20	R1	I1
#2	205.100.0.0	/22	R2	I2

class C /24

$2^{24-22} = 4$

E.g.

IP pkt with dest IP address 205.100.2.1

it matches both entries

=> pick entry #2

" Longest Prefix Match "

Supernet Addressing

- Supernet Addressing allows the addresses assigned to a single organization to span multiple classed address blocks
  - Classless Addressing

- Why was classless addressing adopted?
  - Class B is too large for most organizations
    - At the rate Class B numbers were being assigned, Class B prefixes would be exhausted quickly
  - Class C is too small

  - Supernet Addressing is short-term solution
  - Long-term solution: IPv6 with much bigger address space

Supernet Example

- Example:
  - A supernet contains 16 Class C blocks:
    - From 11001101 01100100 00000000 00000000 (205.100.0.0)
    - Up to 11001101 01100100 00001111 00000000 (205.100.15.0)
  - The common network prefix is 11001101 01100100 0000
  - supernet address = 11001101 01100100 00000000 00000000

  - network mask = 11111111 11111111 11110000 00000000
  - supernet address = IP address AND network mask

  - Slash notation for this supernet is 205.100.0.0/20

Effect of Supernet Addressing on Routing

- With supernet addressing, routing is according to the supernet address (or, the network prefix) of an IP address, not its class
  - This is known as CIDR (Classless Inter-Domain Routing)
    - CIDR collapses a block of contiguous Class C address blocks into a single entry in the routing table
  - Example: 205.100.0.0/20
    - Pre-CIDR: destination network with 16 contiguous Class C address blocks requires 16 entries in the routing table
    - Post-CIDR: destination network with 16 contiguous Class C address blocks only requires 1 entry in the routing table

Longest Prefix Match

- With CIDR, multiple routing table entries may match a given IP destination address
- Example: routing table may contain
  - 205.100.0.0/20 corresponds to a given supernet
  - 205.100.0.0/22 corresponds to another supernet resulted from aggregation of a smaller number of addresses
  - Longest Prefix Match
    - Packet must be routed using the most specific route

      → routing table entry corresponding to the smallest supernet

      → longest prefix match
  - Several fast longest-prefix matching algorithms are available

Routing Scheme

Goal: find the best route between two nodes in the network
?

objective function: min # routers ; fastest route ; least expensive ; ......

assign cost to each link: hop count ; delay ; #

to find the "best" route

= to find a path with smallest path cost ($= \Sigma$ link cost)

(path length)

shortest path

Shortest Path Routing Scheme

① Routing Protocol

* to distribute routing related info between nodes
  - what info?
  - how to ?

② Routing Algorithm

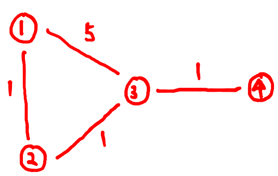
* Executed at each node (router)
* How to use routing related info to compute shortest path and update routing table if needed

1. Distance Vector Routing (DV) vs Link State Routing (LS)

- Each node maintains three lists:
  1. A list of next hops to each destination node along the shortest path
     - routing table node index H_{ij}
  2. A list of distances to each destination node (shortest distance)
     - distance vector path cost D_{ij}
  3. A list of link costs to each neighbor node
     - ∞ link cost to each non-neighbor node

C_{ij}

E.g.



At node 1:

dest j	H_{1j}	D_{1j}	C_{1j}
1	1	0	0
2	2	1	1
3	2	2	5
4	2	3	∞

H_{1j}: routing table

D_{1j} → Distance Vector (global info)

C_{1j} → Link state (local info)

- Only distance vectors are exchanged between neighbor nodes

routing related info (global info)

how to distribute (local exchange)

- Each node will update its lists upon receiving new distance vectors from neighbor nodes

Bellman-Ford Algorithm

Bellman Ford

- Initialization Step

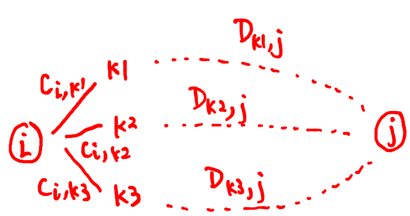
  H, D, C

- Send Step
  - Each node sends its distance vector to its immediate neighbors across the local links (DV report)

Bellman-Ford Algorithm

- Looping
  - If node i receives a distance vector from neighbor k or sees a link cost change to neighbor k, it re-calculates the shortest path to each destination j:
    - $D_{ij} = \min_k \{ C_{ik} + D_{kj} \}$ ✓
    - $H_{ij} = \arg\min_k \{ C_{ik} + D_{kj} \}$: k such that $(C_{ik} + D_{kj})$ is the minimal one
  - If a new D_{ij} or H_{ij} is found, go to Send Step (this is called triggered update)
  - Otherwise, periodic broadcast

E.g.



$C_{i,k1} + D_{k1,j}$

$C_{i,k2} + D_{k2,j}$ } pick the smallest one

* $C_{i,k3} + D_{k3,j}$

E.g. $H_{ij} = k3$

if $(C_{i,k3} + D_{k3,j})$ is the smallest