07 - Routing

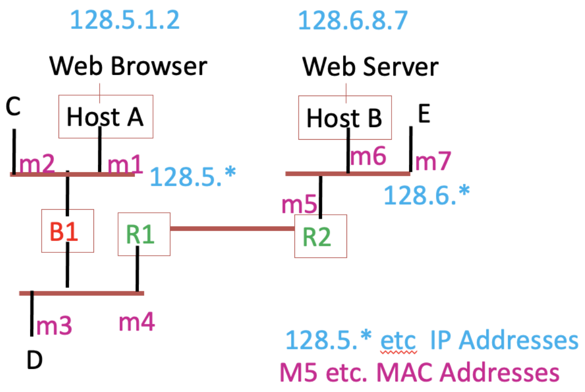
Bridge to Routers

- Data link headers came before data link relays (bridges/switches) so they had to adapt
- but routers were first-class creations. The debate is whether to use bridges universally or routers
- the big problem is no loops in switch topology → need possibly many intermediaries to communicate despite having fiber or really good hardware between stations
- switches also cannot handle **address incompatibility** (HDLC vs Ethernet (PPP)), **packet size incompatibility** and **bandwidth incompatibility** (FDDI vs Ether)

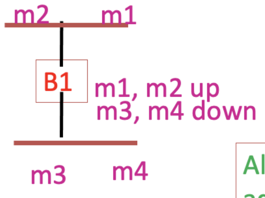
Abstraction

- Bridges extend LANs to extended LANs
- Routers connect extended LANs to WANs
- Consider the following example

Full Topology

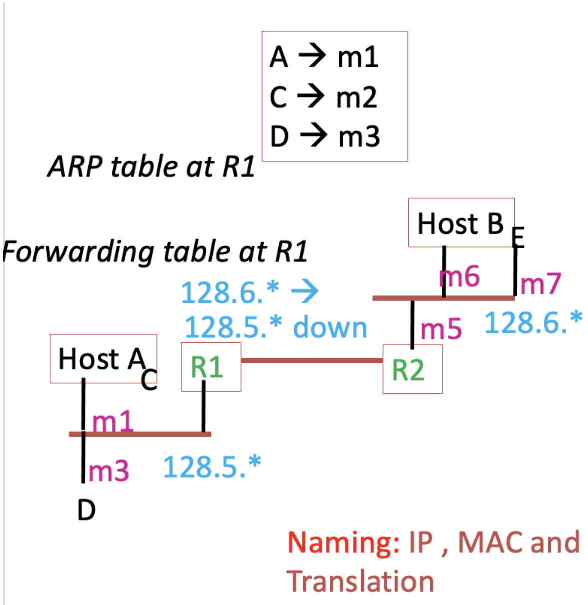


Bridge View

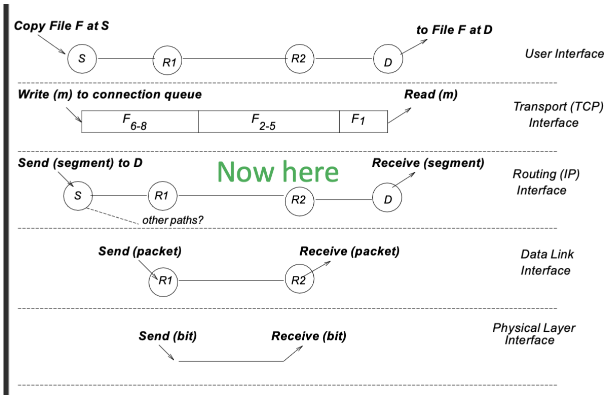


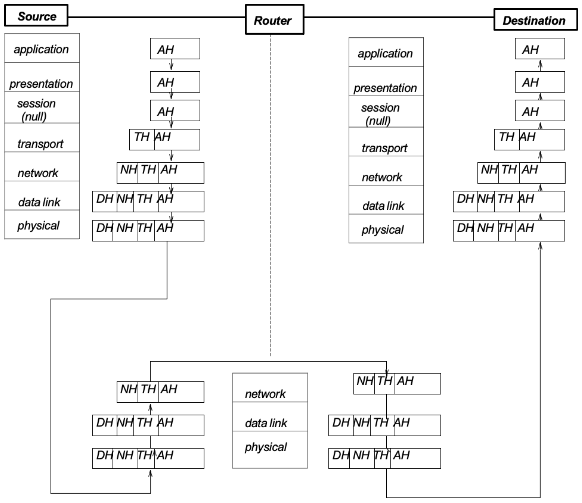
All routers are just MAC addresses to Bridges. R1 is just m4

Router View



OSI Overview





IP Forwarding

Terms

- ISP - Internet Service Provider, usually local with regional POPs
    - usually small ISPs connect with mega ISPs then to NAPs

- POP - Point of Presence, a physical location with a link that ISPs usually have per region
- Autonomous Systems - network managed by 1 manager, entirely contained in a LAN or alike
- NAP - Network Access Point, single connect for all ISPs
  - legacy, very cluttered and congested, instead peering
- Peering - Interconnect b/w ISPs w/o NAPs
  - megacorps like google, meta have their own WANs and peer with everyone with POPs everywhere

Internet

- IPs goal was to interconnect (internet) different network like DECNET, SNA, XEROX Net, Apple Talk
  - eventually all disappeared and IP works directly
- error message backward using ICMP (protocol)

History

- 1970s, ARPANET - linked govt and university ite (UCLA) in 1970s
  - was shutdown by gov too risky
- 1983, NSFNET - 1983 ARPANET splits up into MILNET and ARPANET. In 1984 NSF establishes NSFNET to be backbone. Campuses attached to backbone via regional networks (NYSERNET etc.) Strict hierarchy breaks down because of direct connections between providers
- late 1980s - multiple providers

Domain Name Server (DNS)

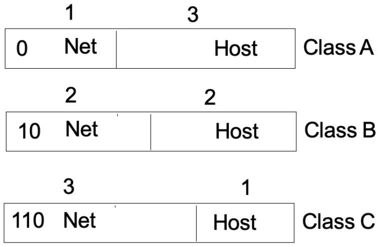
- servers that map domain names (urls) to 32-bit (5-byte) IP addrs
- hierarchical, local DNS knows translations for all in network devices, then subnet IPs and move up the hierarchy for wider prefixes
- then there are root DNSs which store common IP translations, e.g. google DNS IP: `1.1.1.1`

DHCP Server

- dynamic host control protocol
- when a station or node connects, it multicasts to DHCP server which allocates a local IP for the node, the prefix tells us the router IP
- this is slightly different when considering local vs public IPs

Original/Old Model

- small number of large networks (class A), moderate number of campus networks (class B), many LANs (class C)



- Find Dest - parse Network number of dest addr and check for class of addr
- Final hop reached? - if network number of dest = network number of this router' local interface(s) then deliver packet. Map to local address using ARP

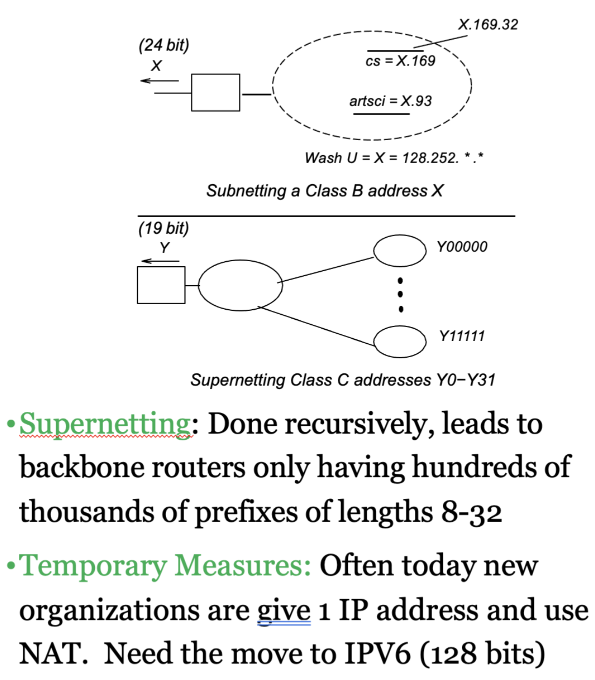
- lookup router table - lookup network number in routing table, if exists → forward. else → send to default router (e.g., many addrs in campus network → have default router and then route internally in campus network)

Challenges

- inefficient address usage - any org that need >255 addrs needed class B → quickly ran out of class B addrs
- routing table growth - response to above → allocate more class C addrs → each core router needed much larger routing tables
- sol - change IP forwarding to longest matching prefix

Subnetting/Supernetting

- slash at the end of an IP tells us how many of the prefix bits to consider to route
- this simplifies comms between networks/LANs by only considering the prefix that matters because all network devices within the network have the same IP prefix
- e.g., 128.32.0.0/16 ⇒ all network devices share the same first 16 bits 128.32
- can encode with slash bits or subnet mask using bitwise & between ip and mask - use 1s for all leading bits that matter i.e. in ints: 255.255.0.0 ⇒ [0-255].[0-255].x.x



Subnetting a Class B address X

Supernetting Class C addresses Y0–Y31

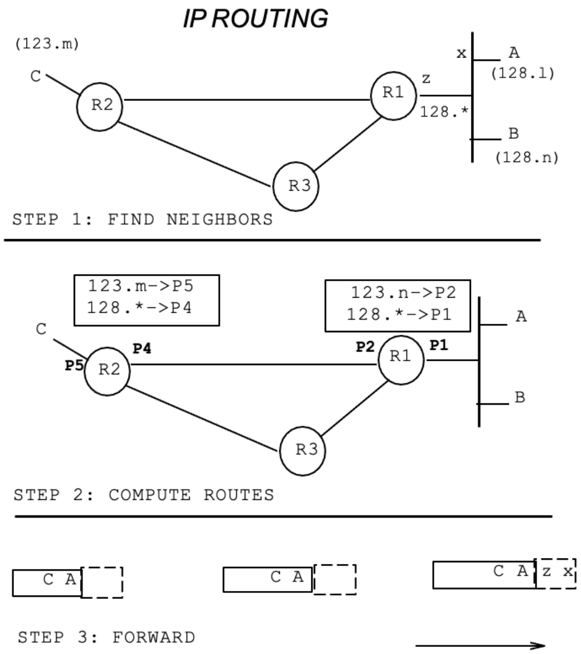
- Supernetting: Done recursively, leads to backbone routers only having hundreds of thousands of prefixes of lengths 8-32
- Temporary Measures: Often today new organizations are give 1 IP address and use NAT. Need the move to IPV6 (128 bits)

New IP Forwarding

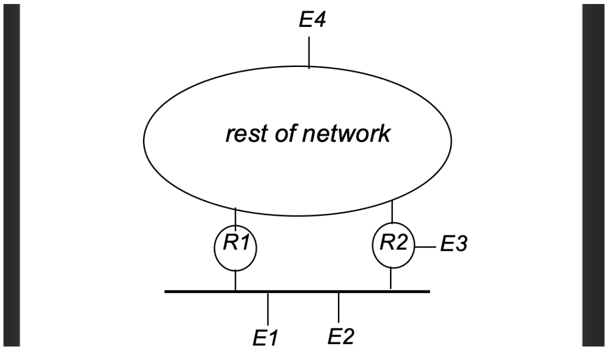
- CIDR - classless inter-domain routing - no more fixed length prefixes (IP classes from original model)

1. lookup - find longest matching prefix P of dest IP addr from forwarding table
2. default or local
    1. if P is nil → forward on default
    2. elif P is associated with local interface → deliver and map to local addr using ARP,
    3. else → forward to next hop associated with P

- router internally has a crossbar switch which is used to bridge packets to the correct data link terminal. i.e. line

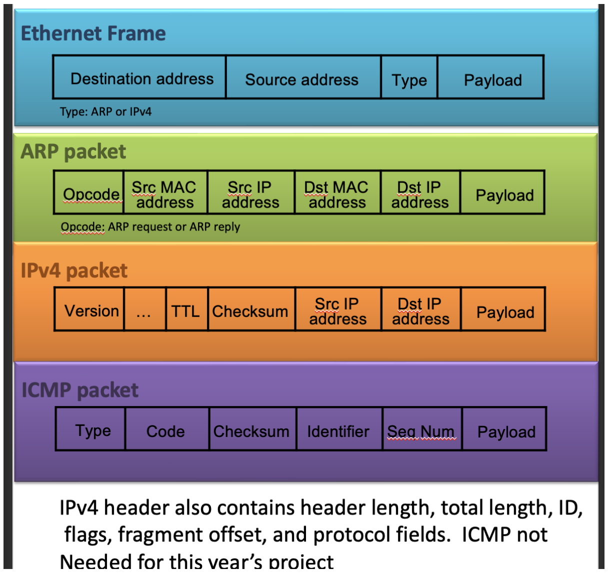
- the forwarding table maps ip prefixes to output links, this is constructed by routers comms with each other telling them that ips are on a specific direction → maps to specific data links → maps to specific mac addrs/ethernet terminals
- NOTE: IP does not yet know the mac addrs it needds to send → see prob and sol below
- 

IP Solution to End-node Problem

- Given top. 

- Prob1 - Routers need Data Link Addresses of endnodes
  - Sol1 - ARP for MAC address of destination
- Prob 2 - Endnodes need DL address of 1 router
  - Sol2 - a service called called DHCP gives you the IP address of one router (auto-configuration)
- Prob3 - E1 and E2 should be able to communicate without a router
  - Sol3 - two endnodes know they are on same subnet by comparing masks. Then ARP
- Prob4 - E1 to E3 traffic should go through R2
  - Sol4 - send to router and router sends redirect if packet returns on interface it entered router. (Ignore this code in project),

Routing Packet Structure



IPv4 header also contains header length, total length, ID, flags, fragment offset, and protocol fields. ICMP not Needed for this year's project

ARP

- sends signal on connected mac terminal to lan, router picks up and propagates back its mac addr, now client knows which router to send to the next hop
  ARP, or Address Resolution Protocol, is a network protocol used to map an Internet Protocol (IP) address to a physical machine address that is recognized in the local network. This is particularly important in IPv4 networks, where devices communicate using IP addresses, but the actual data transmission occurs over the physical network using MAC (Media Access Control) addresses.

How ARP Works:

1. **ARP Request**: When a device wants to communicate with another device on the same local network, it needs to know the MAC address corresponding to the target device's IP address. If the sender does not have this information in its ARP cache (a table that stores IP-to-MAC address mappings), it broadcasts an ARP request packet to all devices on the local network. This packet contains the sender's IP and MAC addresses, as well as the target IP address for which it is seeking the MAC address.
2. **ARP Reply**: All devices on the local network receive the ARP request, but only the device with the matching IP address will respond. This device sends back an ARP reply, which includes its MAC address. The reply is sent directly to the sender's MAC address.
3. **Updating ARP Cache**: Upon receiving the ARP reply, the sender updates its ARP cache with the new IP-to-MAC address mapping, allowing for faster communication in future interactions without needing to broadcast another ARP request.

ARP Cache:

The ARP cache is a temporary storage area where the mappings of IP addresses to MAC addresses are kept. Entries in the ARP cache can expire after a certain period, requiring the device to send a new ARP request if it needs to communicate with that IP address again.

Types of ARP:

1. **Proxy ARP**: This allows a router to respond to ARP requests on behalf of another device that is on a different network. This can help devices communicate across different subnets.
2. **Gratuitous ARP**: This is a type of ARP request sent by a device to announce its IP address to the network. It can be used to update other devices' ARP caches or to detect

IP address conflicts.

Security Considerations:

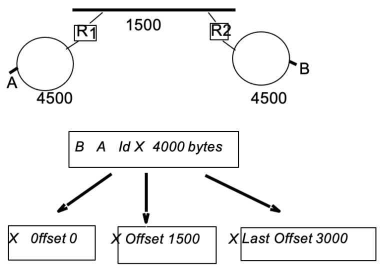
ARP is inherently insecure because it does not include any authentication mechanisms. This makes it susceptible to attacks such as ARP spoofing, where a malicious actor sends false ARP messages to associate their MAC address with the IP address of another device, potentially allowing them to intercept or manipulate network traffic.

Conclusion:

ARP is a fundamental protocol in networking that enables devices to discover each other's MAC addresses based on their IP addresses, facilitating communication within local networks. Understanding ARP is crucial for network configuration, troubleshooting, and security.

Bandwidth Incompatibility (Path MTU)

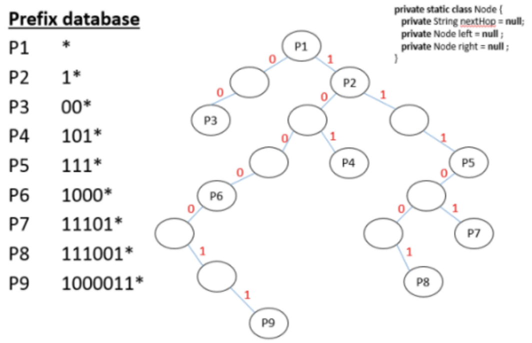
- sps packet is larger than the data link bandwidth



- og IP said to fragment and reassaablle but was too expensive
- modern end node find right size known as **Path MTU** and sends request for this size instead of asking routers to fragment

Fast IP Lookups

- Sol1 - Use unibit prefix trie as lookup table - too slow, 32 steps in worstcase

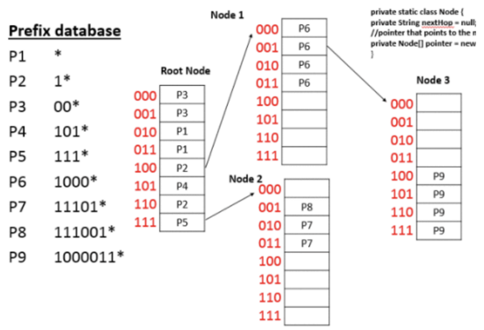


Prefix database

P1	*
P2	1*
P3	00*
P4	101*
P5	111*
P6	1000*
P7	11101*
P8	111001*
P9	1000011*

```
private static class Node {
   private String nextHop = null;
   private Node left = null ;
   private Node right = null ;
}
```

- Sol2 - Multibit trie - to slow and tm memory



Prefix database

P1	*
P2	1*
P3	00*
P4	101*
P5	111*
P6	1000*
P7	11101*
P8	111001*
P9	1000011*

Root Node

000	P3
001	P3
010	P1
011	P1
100	P2
101	P4
110	P2
111	P5

Node 1

000	P6
001	P6
010	P6
011	P6
100	
101	
110	
111	

Node 2

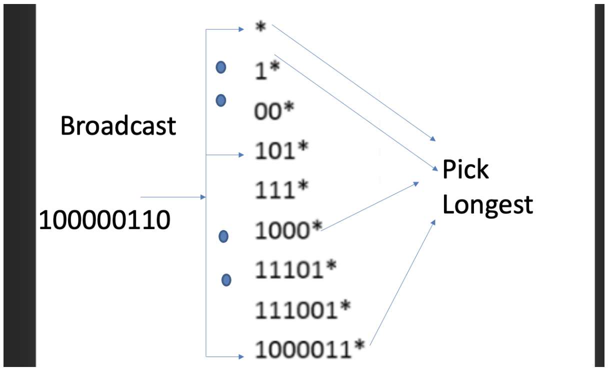
000	
001	P8
010	P7
011	P7
100	
101	
110	
111	

Node 3

000	
001	
010	
011	
100	P9
101	P9
110	P9
111	P9

```
private static class Node {
private String nextHop = null;
//pointer that points to the n
private Node[] pointer = new
}
```

- Sol3 - ternary CAM (content addressable memory) - memory where each bit can be 0, 1, or * that can be searched in parallel because we search by content instead of id or position- but requires tm power at high speed access, low longevity of memory



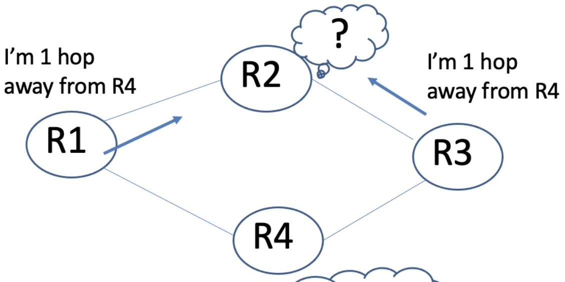
- diff router size ue diff types: compressed multi-bit tries, ternary CAMs

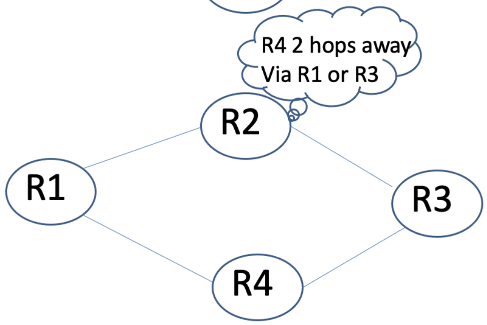
Route Computation

- Flavors:
  - Intradomain routing - within an autonomous entity
    - Distance vector - problems with count to infinity
    - Link State - often used
  - Interdomain routing - between ISPs

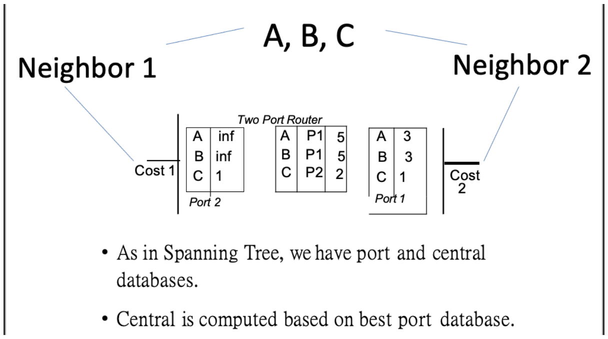
Distance Vector, Gossip

- Routers begin with comms between themselves and figure out neighbors and hops away from routers >1 hop → propagate
- bad if a router fails then no way to know neighbors until they tell u they are there so assume the are and lead to infinite hop count
  - consider that each router stores a routing table with distance to each router (via rumor, we don't actually know the state), sps the following link where A⟷B⟷C where B⟷C fails. Then, C's table is cleared, B realizes it can't reach C directly, but looks at other neighbors and sees A can reach C (A sets the distance to C as dist to B + dist to C), so B updates with A's DV when it's actually not true. Because B's table changed, it propagates changes to neighbors. Now A realizes B's dist to C changed so it updates its count → infinite count on both A & B.
- Now all vectors know their distances to all other routers





Distance Vector (DV) Database


A, B, C
Neighbor 1
Neighbor 2
Two Port Router
Cost 1
Port 2
Port 1
Cost 2

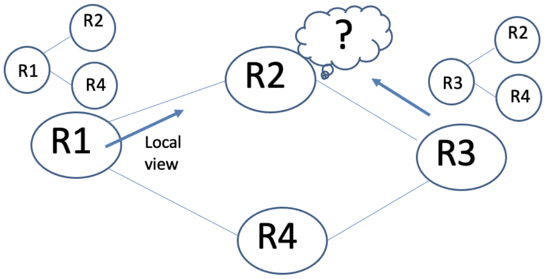
A	inf
B	inf
C	1

A	P1	5
B	P1	5
C	P2	2

A	3
B	3
C	1

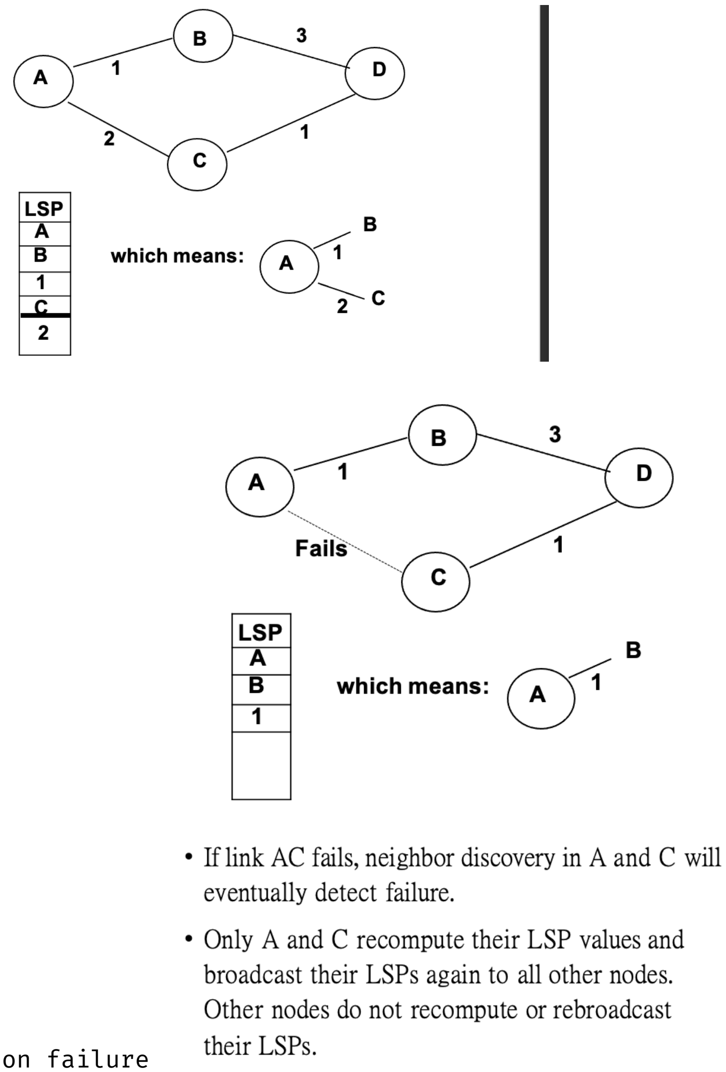
- As in Spanning Tree, we have port and central databases.
- Central is computed based on best port database.

Link State



- each router store its nuclear neighbors as map
- Then each node floods its LSP to all other nodes
- now use any shortest path - Djikstra
- broadcast local state via Link State Packets (LSPs)
- but this causes underutilization - Djikstra enables only shortest path → many links of high bandwidth unused, so we should compute routes using distributed algos

LSP Generation



- If link AC fails, neighbor discovery in A and C will eventually detect failure.
- Only A and C recompute their LSP values and broadcast their LSPs again to all other nodes. Other nodes do not recompute or rebroadcast their LSPs.

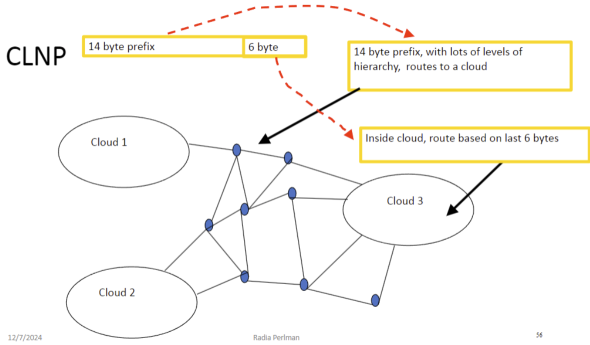
- on failure

Wide Area SDN

- Google B4, Microsoft SWAN, Amazon SDWAN
- within a WAN, just send all packets to the Central SDN controller instead of finding the shortest path
- just forward all packets to SDN and run an algo to find best distributed route, forget protocols

CLNP: IP+Ether Alternative

- redundancy and issues with Ethernet (Layer 2) + IP (Layer 3)
  - Ethernet was meant to be configuration free → 6 byte addresses even just to communicate to fewer nodes on the same link
  - Ethernet never meant to be forwarded, but now it is
  - Ethernet requires MST, no loops
  - IP is HEAVY on configuration and nodes moving readdresses and susceptible to changes
- CLNP - 20 byte addresses per host, shared 14 byte prefix per "Cloud" (the organizational structure, analogous to AS (Autonomous Systems)), node-specific 6 byte suffix

- 

  CLNP

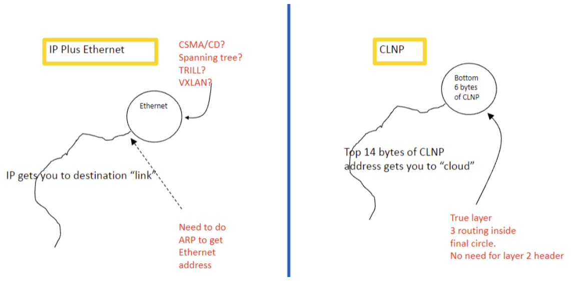
  14 byte prefix

  6 byte

  14 byte prefix, with lots of levels of hierarchy, routes to a cloud

  Inside cloud, route based on last 6 bytes

  12/7/2024 Radia Perlman 56

- "ES-IS" protocol where nodes announce themselves to the routers
- CLNP vs current:
- 

  IP Plus Ethernet

  CSMA/CD?
  Spanning tree?
  TRILL?
  VXLAN?

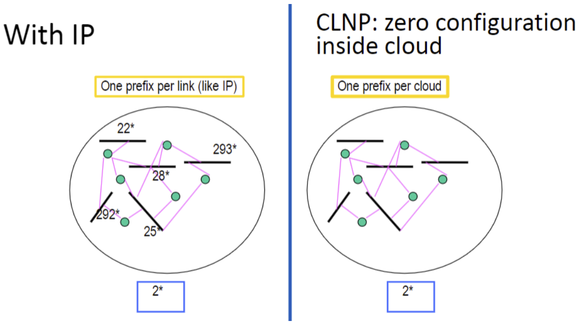
  IP gets you to destination "link"

  Need to do ARP to get Ethernet address

  CLNP

  Top 14 bytes of CLNP address gets you to "cloud"

  True layer 3 routing inside final circle.
  No need for layer 2 header

- 

  With IP

  CLNP: zero configuration inside cloud

  One prefix per link (like IP)

  One prefix per cloud

- advantages
  - no need for dhcp or extra configs per link
  - allows for loops between links and routes
  - no need for NAT (address space is large enough for each node to have its own address, no way there are millions of nodes in a single cloud)
- why it wasnt adopted
  - people didnt like that it didnt follow OSI
  - we still had some IPv4 addresses left, so switch to IPv6 instead of CLNP
  - There was a TCP already built on top of CLNP