Network Layer

Lecture 15 Layer 3 Switching

How do we go from LAN to a much larger network?

Why doesn't ethernet switching scale?

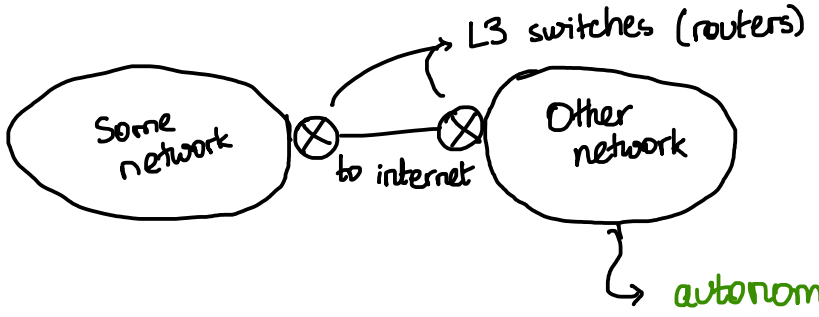
→ In the spanning tree, the path between two nodes could be long. (potentially very unoptimal because we are not using all links)

→ The forwarding table, whose size can be as large as the number of hosts, can be very cumbersome to use.
This is a result of flat addressing.
To fix this, we use hierarchical addressing in IP.

→ If a switch in the tree goes down, we reconstruct the spanning tree.
There are periodic "Hello" messages to ensure that the tree is intact. (if not received by someone, we reconstruct)
In a large network this could happen often, thus wasting resources frequently.

→ Earlier, there were no common addressing scheme or communication protocols across the globe.

L3 switches forward based on the IP address.



autonomous system (AS)

Each AS can choose its own internal routing protocol.
(the distance heuristic mentioned at the end of the prev. section)

There is intra-domain routing (within AS) and inter-domain routing (between AS)

In the internet, inter-domain routing is done using BGP — the Border Gateway Protocol.

Let us start with intra-domain routing. It is broadly of two types: distance vector and link-state routing

- distance vector → RIP (Routing Information Protocol)
- link-state → OSPF (Open Shortest Path First), IS-IS (Intermediate System to Intermediate System)

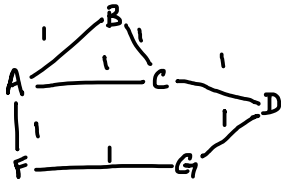
They are essentially just algorithms to:

- → find shortest path (each hop is assigned a weight by the admin)
- → avoid cycles.

A router does not need to know the entire route, only the next hop in a shortest path.

Distance vector routing uses a distributed version of the Bellman Ford algorithm.

Distance vector routing



A (and each node) first sends out (A, 0)

its IP ↵ ↳ distance to itself

It then updates its forwarding table after hearing each message.

Destination	Next hop	Cost
A	-	0
B	B	1
C	C	1
F	F	1

Next, A sends its table to its own neighbours.

(A,0), (B,1), (C,1), (F,1)

From C, it hears (C,0), (A,1), (B,1), (D,1)

It then updates its table as

Destination	Next hop	Cost
A	-	0
B	B	1
C	C	1
F	F	1
D	C	2
G	F	2

Proceeding, it builds up a forwarding table, choosing the neighbour closest to a destination at each step

What happens if a link fails?

A node X recognizes that the link has failed and sends this information to its neighbours, saying that its distance to that node is now ∞.

Similarly, if a neighbour's next hop for that destination is X, it updates its own cost as ∞ as well

This spreads until we reach a node with a different next hop.

If we receive a packet for that node in the intermittent period, it is discarded.

How often does this occur?

→ Triggered update: An event triggers a routing update.
(We try to send on a link and we fail)

→ Periodic update: Periodically, give neighbours information about routing table.

No particular node knows the topology of the entire network.

Lecture 16 Count-to-Infinity and Link State Routing

The Distance Vector protocol essentially shares the dest and next columns of the routing table.

Let us look at the count-to-infinity problem. Count-to-infinity

Say

X —1— A —1— B

Dest	Next	Cost
A	A	1
B	A	2

Dest	Next	Cost
X	X	1
B	B	1

Dest	Next	Cost
A	A	1
X	A	2

Suppose the X–A link fails. Then A sends (X, ∞) (and (B,1)) to B. Further, at nearly the same time, suppose B sends (X,2) (to A) (and (A,1))

A's table was

Dest	Next	Cost
X	-	∞
B	B	1

and on hearing B, becomes

Dest	Next	Cost
X	B	3
B	B	1

and simultaneously, B's table becomes

Dest	Next	Cost
A	A	1
X	-	∞

Now, A tells B (X,3), (B,1) so B will update to (A, A, 1), (X, A, 4).
This repeats ad infinitum, with the cost to X "counting to infinity".

One solution: Keep a maximum distance considered as ∞ and stop if we reach it.
This value is 16 in RIP.

Split-Horizon

One other solution is split-horizon.

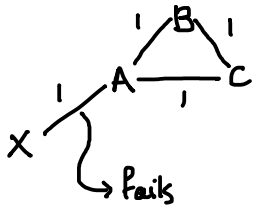
→ Do not advertise information about a destination to a neighbour if that neighbour is the next hop to the destination.
In our example, B would not tell A anything about X.
As a result, both nodes would end up with a (X, ∞) entry. ↳ (the (X,A,2) entry in particular)

Split-Horizon with Poison Reverse

Another is split-horizon with poison reverse

→ A node tells its next hop to the destination that its distance to the destination is ∞.
B sends advertisements (X,∞) to A.

However, the above do not fix the problem in general.



(there are also more general counterexamples that do not depend on this)

A sends (X,∞) to B and C Suppose the message to C is lost
Then, B's table is updated with (X, C, 3). Now, B tells this to A (A is not its next hop anymore) and A's table is updated with (X, B, 4).
A will relay this to C. As before, this is a loop and the count-to-infinity problem arises once more.

In RIP (Routing Information Protocol),
The cost of all links is 1.
The routing problem is partially fixed ($16 \equiv \infty$ now) but as a result, we cannot have larger networks.
↳ more than 16 hops

Distance Vector:
+ simple and easy to implement
- count-to-infinity and routing loops
- convergence of routing tables may take a long time.

Link-State Routing

The alternative is link-state routing.
Each node broadcasts information about costs to immediate neighbours.
(sort of a flipped version of D.V. — globally tell local information instead of locally tell global information.)
Each node can reconstruct the entire topology and find the shortest distance using any standard algorithm.
LSR uses Dijkstra's Algorithm, wherein each node finds a shortest path tree to all the other nodes in the network.
A's routing table is then built from the tree.
Routing loops are not a problem because on link failure, the failure is broadcast to everyone (from both sides).
All nodes rerun Dijkstra's Algorithm.
+ No routing loops or count-to-infinity.
+ Convergence of routing table is fast.
- Algorithm is more complex (than Distance Vector)

The remaining question is: what do we choose for the costs?

We have studied how to use the weights to find the shortest paths. What should these be in practice?
A lower weight corresponds to being used more often

In ARPANET, there were 56 kbps and 9.6 kbps links. There were also terrestrial and satellite links.

Let us zoom in on a single link. What weight do we assign?

Idea 1. Use latency. The latency of a single packet on this link is equal to (queuing delay + speed of light delay + transmission delay)

$$\text{transmission delay} = \frac{\text{packet size}}{\text{no. of bps}}$$

This latency keeps changing from packet to packet however. Take some time window instead and set the link weight as the average of all packets in the window.

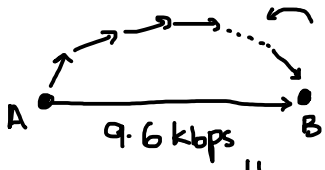
The weight is low if
- → the queue is relatively empty
- → the link is fast

However, this encountered several problems.

→ Under heavy load, there were several routing oscillations. If we decide to use a link, the queue fills up and we switch back to another link.
(Using a link increases its weight over time)

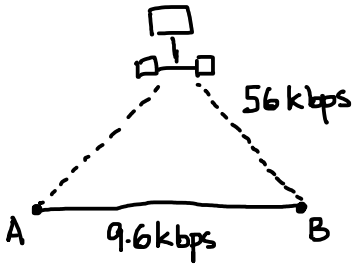
- The end-to-end latency keeps changing, which might affect the application layer.
- The order in which packets are received might be wrong, since we could change link weights halfway through. Again, this could affect application layer performance.
- Routing loops are possible because weights may change often. (the algorithm ensures acyclicity for fixed weights)

→ The range of link weights is large.
As a result, some links are penalized too much.



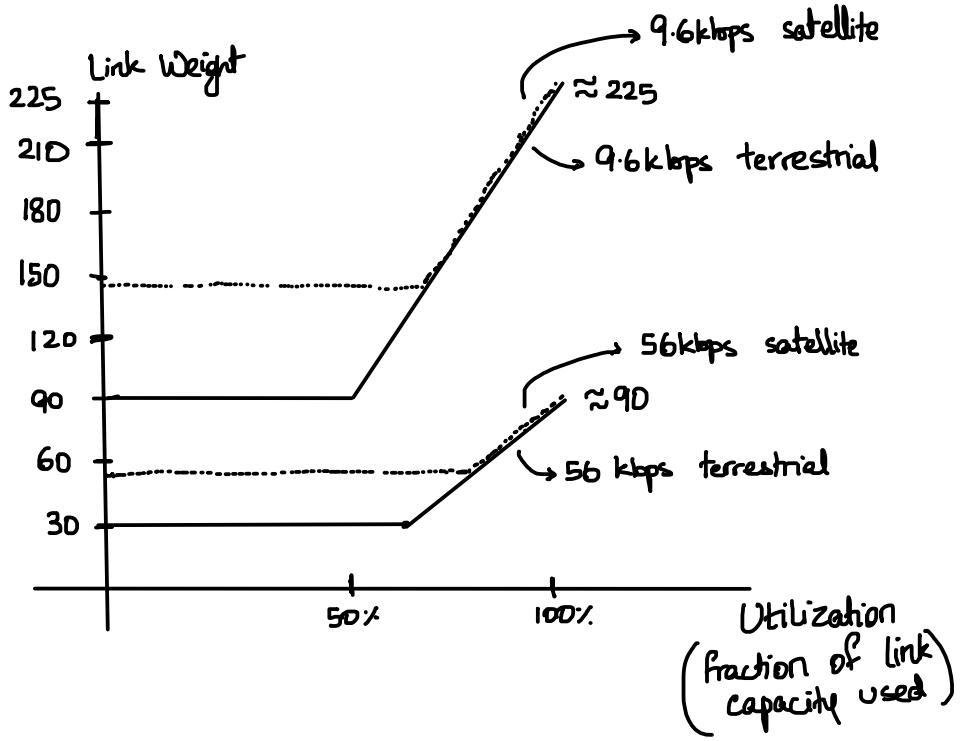
The two paths have equal weight.
It makes more sense to just use the 9.6 kbps link.

→ Satellite links are penalized too much.



→ Due to speed of light delay, this could have too much weight.

What they finally used is:



→ Ratio of max wt. to min is ~7.
→ 56 kbps satellite preferred to 9.6 kbps terrestrial
→ Weights changed infrequently.

What do we use today?

→ In OSPF,

$$\text{Link wt.} = \max\left\{1, \frac{10^8}{\text{Link speed (bps)}}\right\}.$$

→ In Network Operations Centers (NOCs), network engineers can manually change and set weights.

Lecture 18 IP Addressing

Recall that we have hardcoded MAC addresses in Layer 2.

The IP address (layer 3) is configurable.

IPv4 had 32 bits (not enough) — IPv6 has 128 bits.

NAT (Network Address Translation) is used to reuse IP addresses.

The IP address is written as

___ . ___ . ___ . ___

each 8 bits, written as decimal.

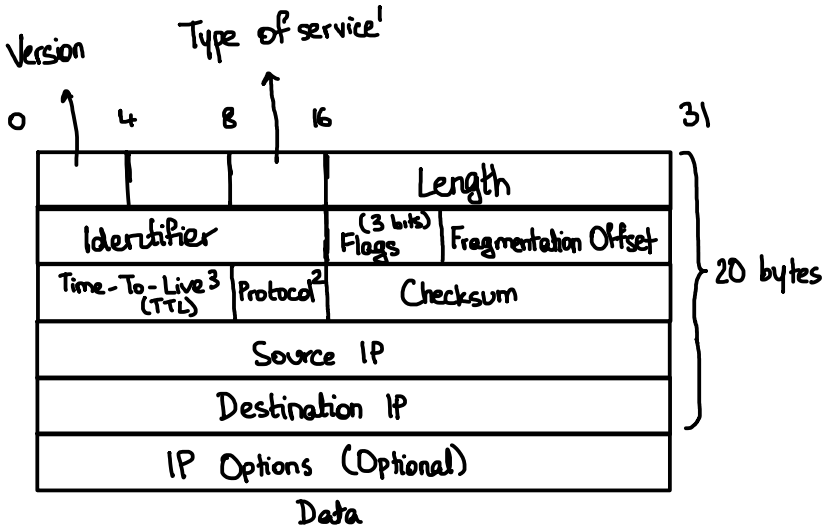
For example, an all 1s address is written as 255.255.255.255.

Special addresses:

- The all 1s address is reserved for broadcast (to be read by all host machines).
- 10.*.*.* or 192.*.*.* is a private IP address (anything) (They can be reused)

  Public IPs are usually unique on the internet.

The IPv4 header looks like



1. demarcate priority of packet
2. Protocol at next layer
   6: TCP
   17: UDP
   1: ICMP
3. Packet survives iff TTL>0. Decremented at each router. Discard if TTL=0. (Fixes routing loops)

Let us now look at the IP addresses.

We want the routing table to be small. (not size 2^{32})

Flat addressing like in Ethernet would be problematic.

Say IP is a.b.c.d.

Assign a slice of addresses instead of arbitrary values. Keep some prefix, so now everyone's address in that region has a particular prefix, making it easier to store in the table.

Class A: 8 bits (Network (Common)) 24 bits (Host) Up to 2^{24} hosts

Class B: 16 bits Network 16 bits Host

Class C: 24 bits Network 8 bits Host

Subnetting: Given a slice, how do we divide addresses among LANs and configure the internal router?

Subnetting

Say class C.

24 bits 8 bits

73.52.30. ?

First, we should divide between the various LANs

A subnet mask denotes which bits in the IP address to use when deciding which LAN to route to.

Say the subnet address for LAN 1 is S_1, and LAN 2 is S_2.

If the destination address is D, we check if

$(D \text{ AND } M_1) = S_1$ or $(D \text{ AND } M_2) = S_2$ (or neither)

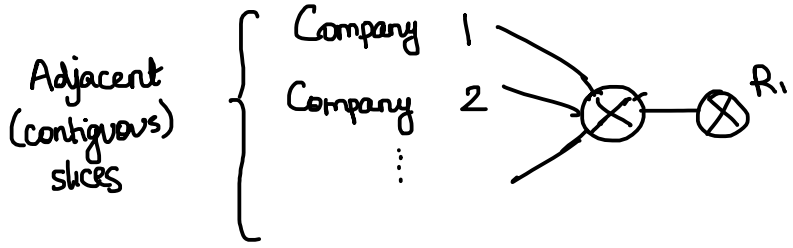
↓ mask for LAN 1

so if 73.52.30.*, it would be 1111.1111.1111.0000.

↓ not meant for either LAN.

(M could be equal to M_2)

We may sometimes want to do the opposite.
That is, combine multiple slices.



Can the entries at R_1 be combined?

Supernetting

This process is called supernetting.

(Maybe the slices combine to form a single prefix)

An IP prefix is usually denoted as

$a.b.c.d/N$

↳ consider N leading bits

(Check if first N bits of destination correspond to that of a.b.c.d)

For example, combine

128.112.128.* ⋮ 128.112.135.*

} to 128.112.128.0/21

≡ 128.112.128.0/24

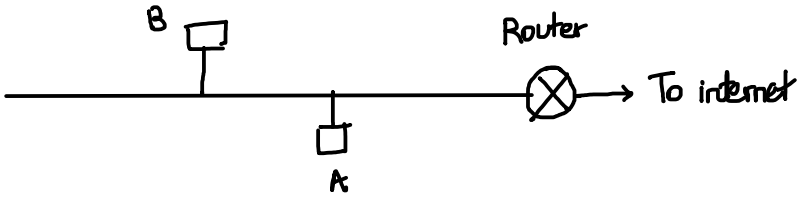
We should ensure that no addresses are missing in the middle.

This method using arbitrary prefix lengths is called CIDR - Classless Inter Domain Routing.

Lecture 19 ARP and DHCP

We now look at ARP - Address Resolution Protocol.

ARP



Suppose A wants to send a packet to B (and knows B's IP).
What MAC layer frame does it send if it does not know B's MAC?
(The ethernet frame has destination MAC after preamble)

ARP helps us determine MAC given the IP.
↳ above DLL, PHY.

A sends an ARP packet. (Preamble | Dest. MAC | Source MAC | Type/Length | ARP packet | CRC)

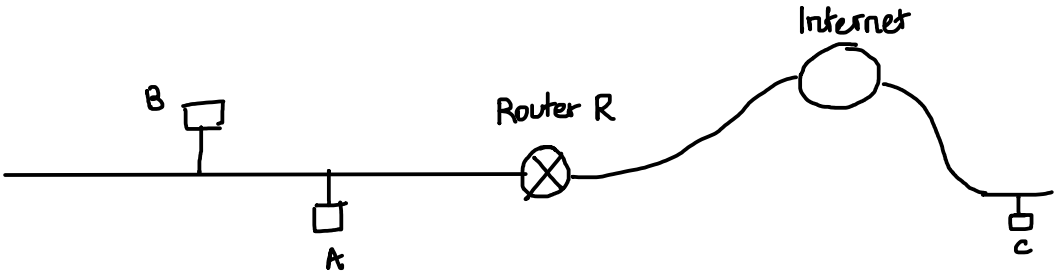
→ the destination MAC is all 1s (broadcast)

→ the ARP packet contains the sender (A) MAC, sender IP, target (B) MAC, and target IP
↳ all 0s because it is unknown.

→ Each receiver checks if their IP matches, and sends an ARP reply if it does.

→ The ARP reply has the sender (B) MAC, sender IP, target (A) MAC, and target IP. This is a unicast frame.

→ This information about B's MAC is stored in A's ARP cache. This has a timeout (of the order of minutes) because IP-MAC binding is not permanent — we can switch out our WiFi card.



What do we do if C (instead of B) is problematic?
It would be fixed if A manages to intelligently set the destination MAC as that of R if this is the case.

(i) How does A know if the destination IP belongs to their own network?

(ii) If not, how does it get R's MAC address?

- For (i), we can AND the destination IP with the subnet mask and check if the resulting IP is equal to our own IP ANDed with the mask. (Dest. IP AND MASK == IP_A AND MASK)
  If it is, then B belongs to our own network.
- For (ii), if we know R's IP address, we can perform usual ARP to get R's MAC.
- How do we get R's IP short of manual configuration?

If the 2 bytes assigned for the type / length are >1536, it corresponds to type. (not length)
(Normally, it is <1500 for length)

In particular, 0x806 means that it is an ARP packet.
↳ in hexadecimal

We get the default router's IP, and our own IP using DHCP — Dynamic Host Configuration Protocol.

DHCP

There should be a DHCP server on the network.

DHCP is above UDP, IP, DLL, PHY on the protocol stack.

(alongside application but shares information with IP/DLL)

The data is broadcasted. The IP has a protocol field for UDP and a port number for DHCP.

↳ 68, 67
- ↳ DHCP client
- → DHCP server

A sends out a DHCP Discover packet.

→ The destination MAC is all 1s. (broadcast)

→ The destination IP is all 1s. (broadcast)

→ A receiver discards the packet if port 68 is not open.

→ The DHCP server replies with an offer, which contains a potential IP address for A (the server looks up its tables for a free address).

The destination MAC is A's.

Discover
Offer

The source MAC is the server's.

The source IP is the server's.

The destination IP is all 1s (broadcast) — A need not have configured itself with the correct IP address.

→ This is followed by a request from A for the IP address, and then the server acknowledges that the IP is assigned.

Why are we doing this in two rounds (discovery, offer, request, ack)?

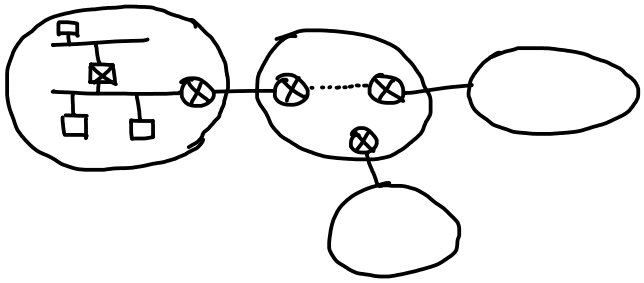
- There may be multiple servers, and we want to allocate a single IP.
- When A receives the offer, it checks if this IP is already in use by sending out an ARP.

When we set the destination MAC as all 1s, shouldn't it reach the entire internet? No, gateway routers do not send out IP broadcasts and keep them local to this network.

What if there are multiple networks with a single DHCP server? There is a relay agent that sends a unicast DHCP packet to the DHCP server, and forwards the offer to A after receiving it.

Now, we look at BGP (Border Gateway Protocol), which is used for inter-domain routing (routing between autonomous systems).

BGP



The AS together form a network.

An autonomous system connected to multiple other AS and paying them for their service is called a multi-homed customer

Autonomous systems with a wide coverage (across the country / globe) are called Tier 1 autonomous systems. They are usually not customers

An AS taking service directly from a tier 1 ISP is called a tier 2 AS.

tier n tier (n+1)

A customer is an AS that does not provide service to anyone

A peering link is a link between two AS, neither of which take service from the other (they co-operate and it is convenient to send through them). Their bandwidth is usually low to avoid misuse — they are under-provisioned since no-one gets paid for traffic on that link.

Peering Link

A service-level agreement gives a guarantee about the throughput / latency within a certain AS (NOT end-to-end). This is within the network, so depends on the link weights etc.

What does BGP do? It helps to find a path across the AS-level graph.

abstract the AS to single nodes

There are BGP speakers, which communicate using the e-BGP protocol.

routers running BGP external

Speakers

They advertise the prefix ($x.y.z.w/N$) / AS path / attributes as a BGP advertisement

To communicate between speakers of a single network, we use i-BGP.

interior

The known prefix / AS path / attributes are sent.

The path gets longer as we proceed.

(note that path is stored, not next hop)

So if an announcement with a certain address is made, we are guaranteeing that we can reach that address.

On peering links though, we might want to keep certain paths secret.

Corporate greed is a feature of BGP!

Even customers might not want to tell what it is connected to, for fear of being used as an intermediate service provider by the connected providers. BGP allows this too.

It does not compel *anyone* to give out all their information.

Since there may be multiple paths to the same prefix, we must make a choice.

The communicating e-BGP routers are usually directly connected.

(e-BGP uses port 179) so for HTTP

In i-BGP, the learnt information is shared within the AS, to all the BGP speakers.

using IGP

IGP is Interior Gateway Protocol, which is just normal intra-domain routing.

- → e-BGP speakers learn AS paths from neighbouring e-BGP routers in other AS.
- → e-BGP nodes share learned information via i-BGP to other speakers in their AS.

IGP

- → BGP speakers select routes to various IP prefixes.
- → Insert chosen routes into IGP, since all routers must be able to forward packets to a destination IP.
- → (Optional) Announce newly created routes to the neighbouring AS.

BGP Attributes

What are BGP attributes?

(LOCAL_PREF) 1. Local preference: The admin of the AS can say if they prefer a particular path.
(higher local preference ≡ attribute value is greater ≡ more preferable path)

(MED) 2. Multi-exit discriminator: If there are multiple connections between two certain AS, this shows the preference of using the current router.
(lower value ≡ would prefer to receive messages on this router)
Advertised by the person sending the route. Could set the MED as distance to the particular router.

(AS_PATH) 3. AS-path: This is just the list of AS numbers to the destination with that IP prefix.
↓
first AS which advertised

The destination AS need not contain the destination IP due to say, supernetting.

4. Next hop: This is the IP address of the next router in the path (in the adjacent AS). This can be shared within a AS (using i-BGP).

Each BGP speaker chooses which AS route to use among the many available for the same prefix. The rules to do so (in order of preference) are:

1. Use a route with the largest LOCAL_PREF.
2. Use a path with the shortest AS_PATH. (number of AS on path)
3. Use a path with the lowest MED
   (even across paths with distinct adjacent AS)
4. Use a path learnt over e-BGP instead of i-BGP.
5. Use a path with the least IGP metric to the next hop. ← all tied must be i-BGP
   ↳ intra-domain

Hot Potato Routing

This is called hot potato routing.

6. Use the path with the lowest ROUTER_ID among the BGP speakers which have sent the advertisement.

   ROUTER_ID = highest IP address of all the router interfaces.

Lecture 21 BGP-IGP interaction

How does BGP interact with IGP?

1. Encapsulation: Suppose we want some BGP speaker R_1 to use hot potato routing to another speaker R_2 in the same AS but the intermediate routers do not know about BGP. The prefix we want to send is not in the IGP routing table.

Encapsulation

R_1 encapsulates the normal IP packet in another packet (with source R_1 and destination R_2) and sends it. The old IP packet becomes the payload of the new packet.
On receiving, R_2 strips off the outer layer and forwards the internal packet onward through BGP.

2. Pervasive BGP: This is the case when every router in the AS is a BGP speaker. There is a BGP table and an IGP table at each node. The former is used to determine the gateway/exit and then the latter is used to determine the next hop to send to the gateway.

Pervasive BGP

3. Tagged IGP: This is the only method where there is actual interaction. Internal routers may not be BGP speakers, but IGP allows addition of tags. The destination (IP prefix) is tagged with the gateway router and then broadcast. The IGP routers search for the (dest., tag) pairs in their IGP table and choose the path with the least cost.

Tagged IGP

Recall supernetting.

Suppose we have both 142.31.135.0/24 and 142.31.0.0/16 as entries and we get, say, a packet for 142.31.135.20. Which route do we choose?

The rule is that we choose the longest prefix that matches.

How do we quickly perform longest prefix matching?

A hardware solution is CAM: Content Addressable Memory.
(we do not discuss it here)