- About the Author ?
  - Abhishek Sagar, MTech from IIT Bombay, 2012 Batch in CSE
  - Expert in Networking Domain, System Programming
  - Passionate Software Developer and loves knowledge sharing
  - Past Employers : Brocade Communications (2012 – 2017)
    Juniper Networks (2017 – 2021)
  - Current Employer : Cisco Systems (2021 – Present as on Jan 2022)
  - Contact
    - Email : csepracticals@gmail.com
    - www.csepracticals.com
    - Telegram grp : telecsepracticals (For instant Tech Query resolution)
  - Social Presence
    - LinkedIn : https://in.linkedin.com/in/abhishekiit

Lecture VDO 0

Introduction

- **Who should do this course ?**
  - Computer Science Students – at-least have completed 3rd year of their engineering
  - Working professionals in stucked in testing profile networking domain and want to switch to Development jobs
  - Students looking to build up the career in High paying product based MNCs – (Not limited to Networking based MNCs)
  - This course is based towards covering Networking concepts
  - Its guaranteed, you would learn things which you never heard of in this course
  - The content of the course starts from very basic level and progress to advance concepts – at each stage the concept is demonstrated through mini assignments
  - **Helping resources –**
    - We enlist the helping links/Documents/ebooks to cover the required topics as we progress into the course
    - There are many thick books available in the market - please don't overwhelm yourself
    - Follow the course, and you will be at right place at right time
    - Not everything mentioned in books is a topic of interview or usable in the networking
  - **What Next ?**
    - To be a complete package to join the Networking field as a DEVELOPER , You must do Networking certifications In addition to this course

- **Table of Contents**

Setting up Linux Development Environment on your machine

- **Table of Contents**

0. Setting up Linux Development Environment on your machine (optional)

Installation of Virtual box and Linux OS(Ubuntu 14.04) And Use GIT for all your projects and assignments
https://www.youtube.com/watch?v=LjTjFieUNBM

Please Google to seek help if you stuck somewhere !

Check List :

1. Have working Laptop with host OS (Host OS could be any OS – Windows Or Linux)
2. Have Virtualization software installed – **Virtual box** in our case
3. Have Ubuntu 14.0.4 installed in Virtualization software
4. Test internet connection from inside your Virtual Machine (Ubuntu) by pinging to www.google.com from Terminal
5. Have VIM, GIT, CSCOPE, SCREEN installed in your VM (Linux OS)
6. Login from Windows using ssh client (MobaXterm) into your VM
7. Created Git hub account and get familiar working with it

Lecture VDO 1

OSI Model

- **OSI Model and TCP IP stack**

Theoretical OSI model

Application layer
Presentation layer
Session layer
Transport Layer
Network Layer
Data link layer
Physical layer

The Open Systems Interconnection model is a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology

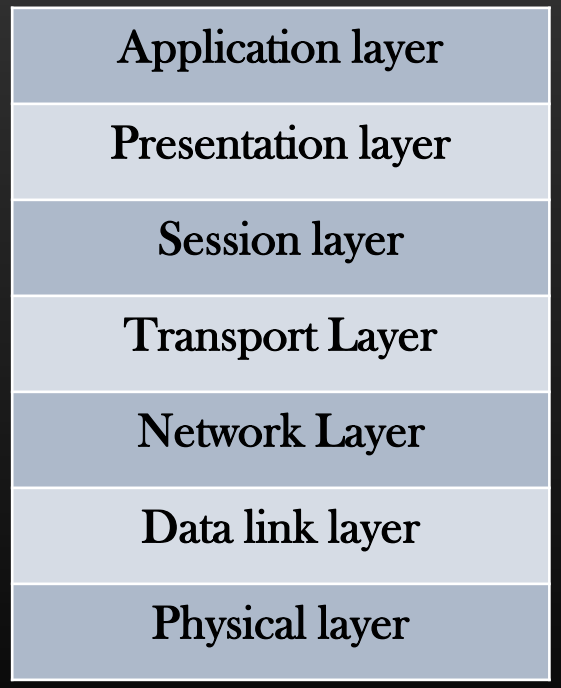
Reference/Standard/Guideline . . .

Lecture VDO 1

OSI Model

- **OSI Model and TCP IP stack**

Theoretical OSI model



Reference/Standard/Guideline . . .

- Description of the Networking subsystem (network stack)
- It is a guideline . . .
- Layer – logically complete functionality of a networking component is referred to as a layer
- Each layer has a specific function
- Functions of layers do not overlap
- Data/packet moves across the layers bi-directionally
- All layers stack together to built a complete networking subsystem
- One most common example of OSI model implementation is TCP/IP network stack which runs inside your OS

Lecture VDO 1

OSI Model

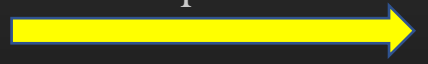
- **OSI Model and TCP IP stack**

Theoretical OSI model

Application layer
Presentation layer
Session layer
Transport Layer
Network Layer
Data link layer
Physical layer

Reference/Standard/Guideline . . .

Actual Implementation



Practical OSI model
TCP/IP Stack

Application layer
Transport Layer
Network Layer
Data link layer
Physical layer

What is actually implemented in OS

Pres. layer and session layer are partially Implemented in layers above it and below

Lecture VDO 1

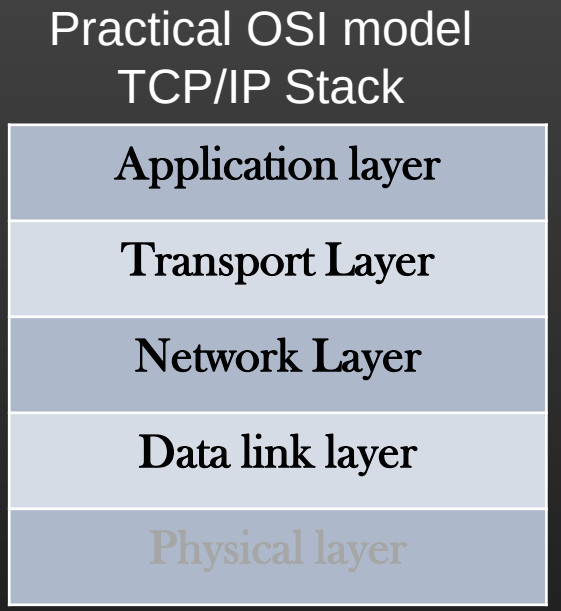
OSI Model

Type of Questions asked from OSI model:

1. Given a protocol X (X – ARP, ICMP, HTTP, DNS etc) , on which layer of OSI model does protocol X operates ?

2. What are the changes happen in the packet content as it traverses through the TCP/IP stack during the course of incoming and outgoing packet journey ?

3. What happens at each layer when a packet is received :
   a. Destined for remote machine
   b. Destined for Local machine



TCP/IP Stack Layer Relation Ships

- Application
- Transport
- Network
- Data Link
- Physical

Transport Layer

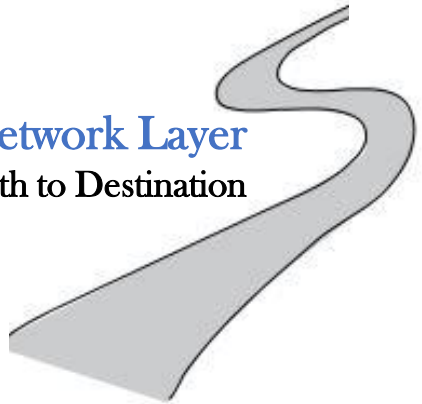
TCP : Responsibility to safely take your Luggage to destination

UDP : You are responsible for your Own luggage loss !

Application

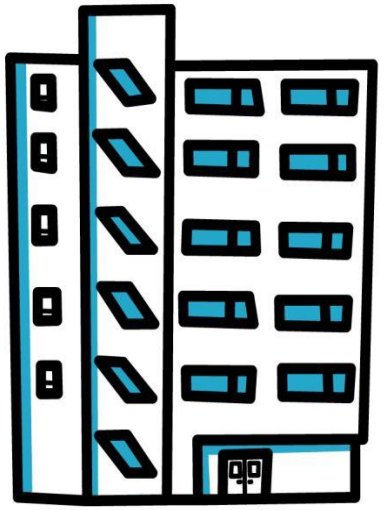
Network Layer

Path to Destination



Data Link Layer

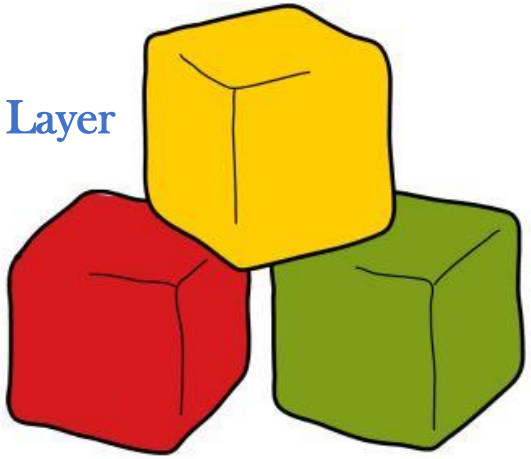
Traffic signals on Road (hop by hop)



New Apartments Destination

Session & Pres Layer

Packing of Luggage (Encryption & Decryption)



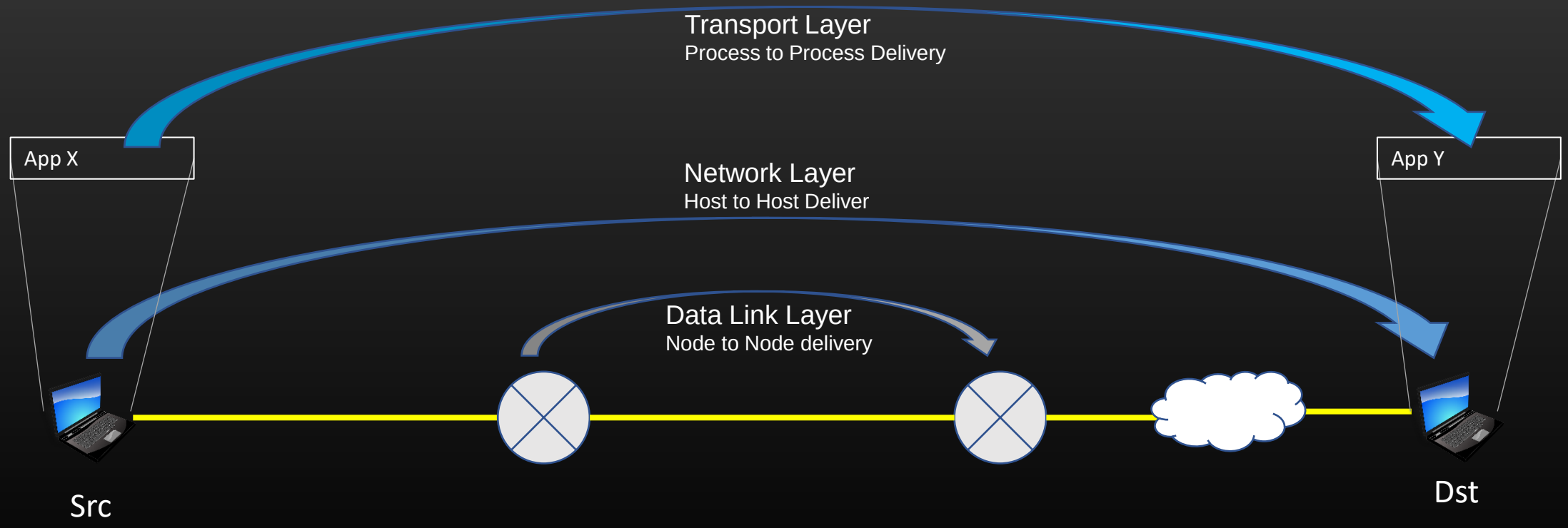
Application Data

Lecture VDO 1

OSI Model

- Isolated, Non-Overlapping Responsibilities

Application layer	Generic, implements networking application
Transport Layer	Process to Process Delivery
Network Layer	Source node to Destination node
Data link layer	From node to its adjacent node
Physical layer	Transmit data as electrical signals

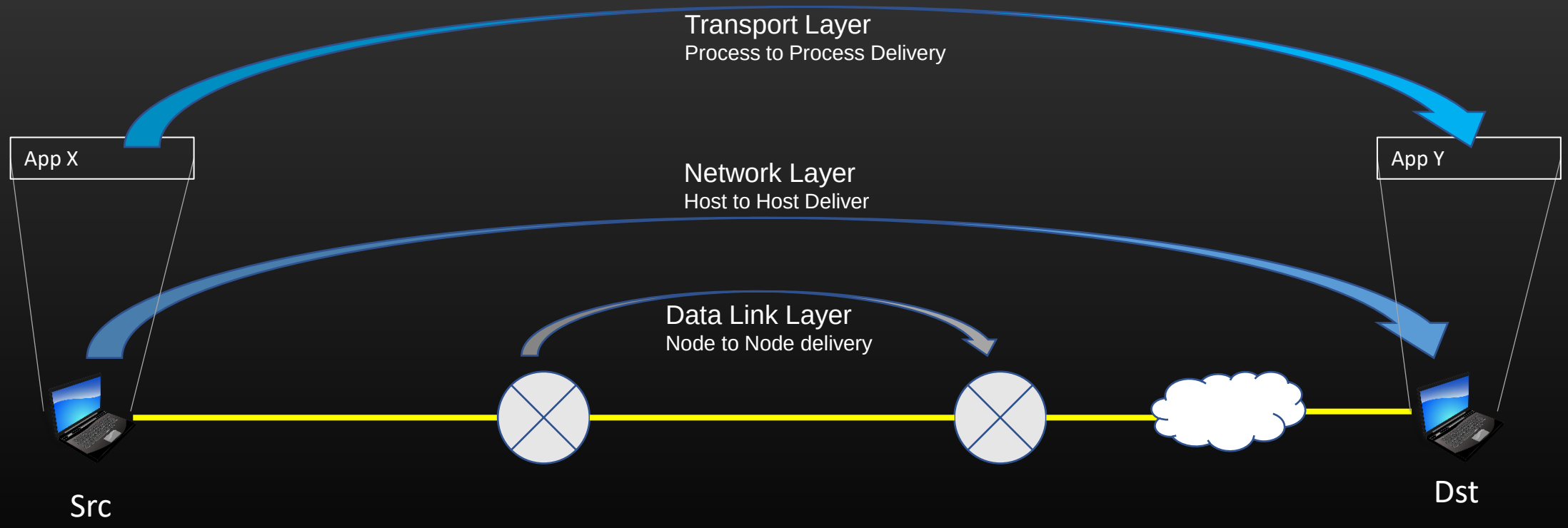


Lecture VDO 1

OSI Model

- No layer is aware of layer above it or below it

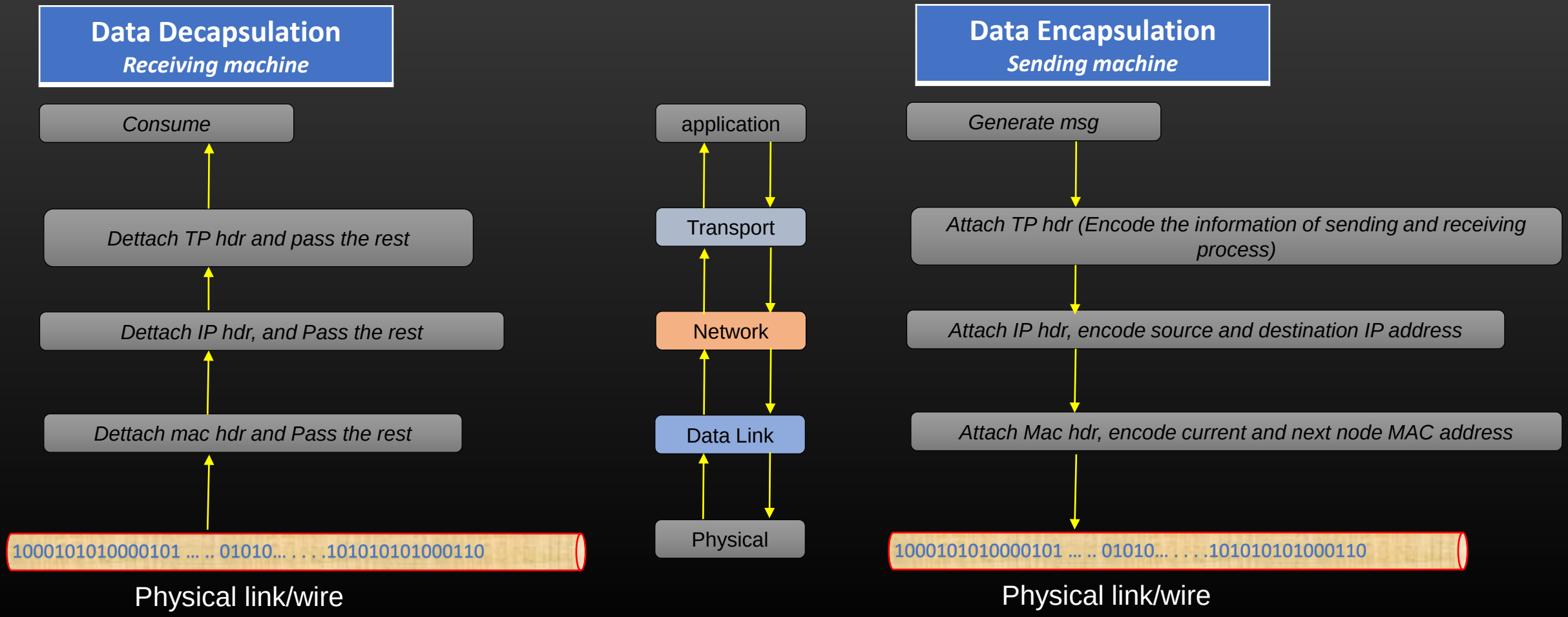
Application layer	Ping (ICMP), HTTP, WhatsApp, All mobile APPs etc
Transport Layer	UDP, TCP
Network Layer	IP, IPv6
Data link layer	Ethernet
Physical layer	



Lecture VDO 1

OSI Model

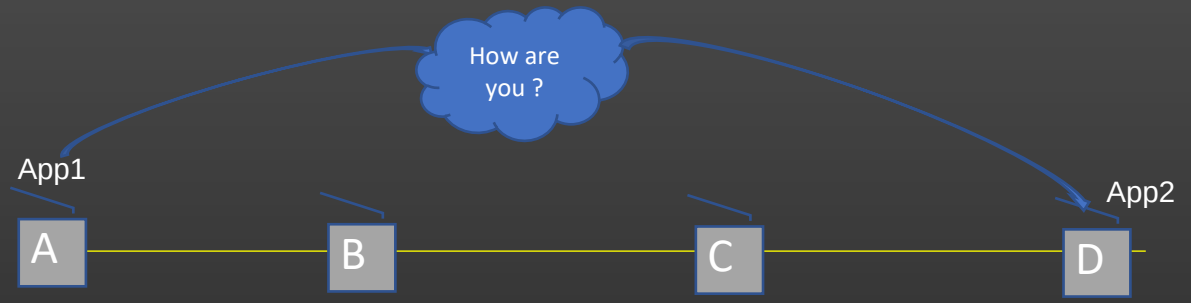
- **Data En-De-capsulation**

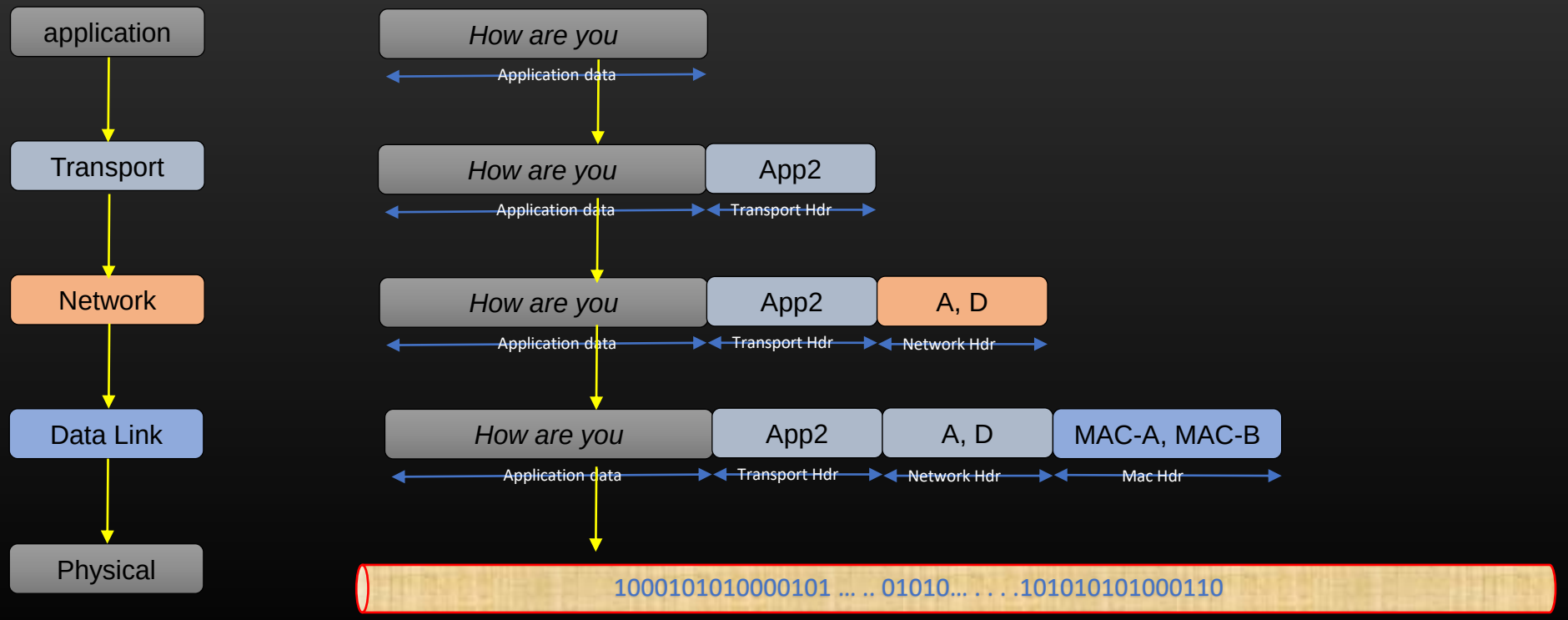


Lecture VDO 1

OSI Model

- **Data Encapsulation on A (Sending machine)**

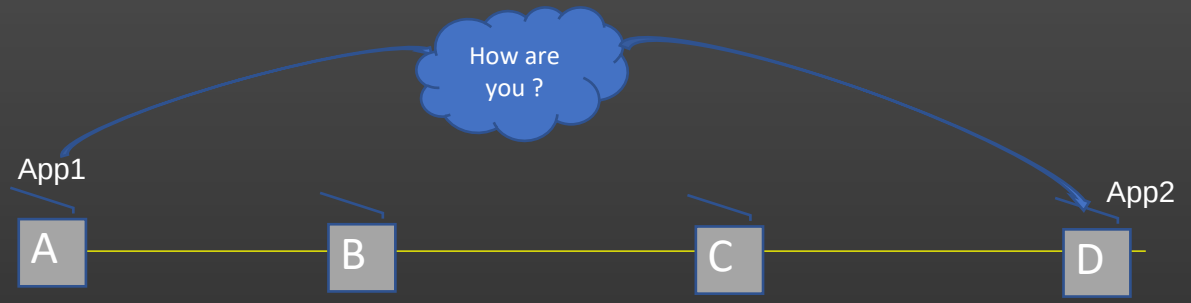


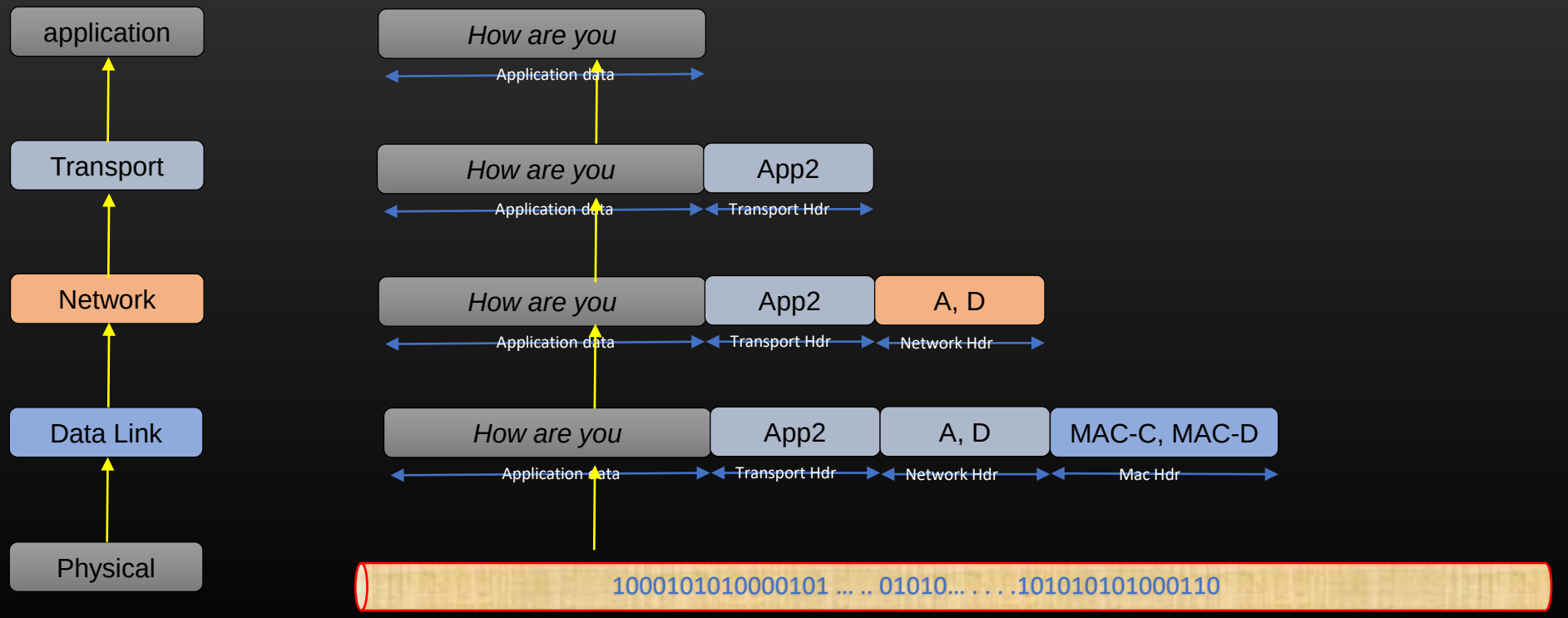


Lecture VDO 1

OSI Model

- **Data De-capsulation on D (Receiving machine)**

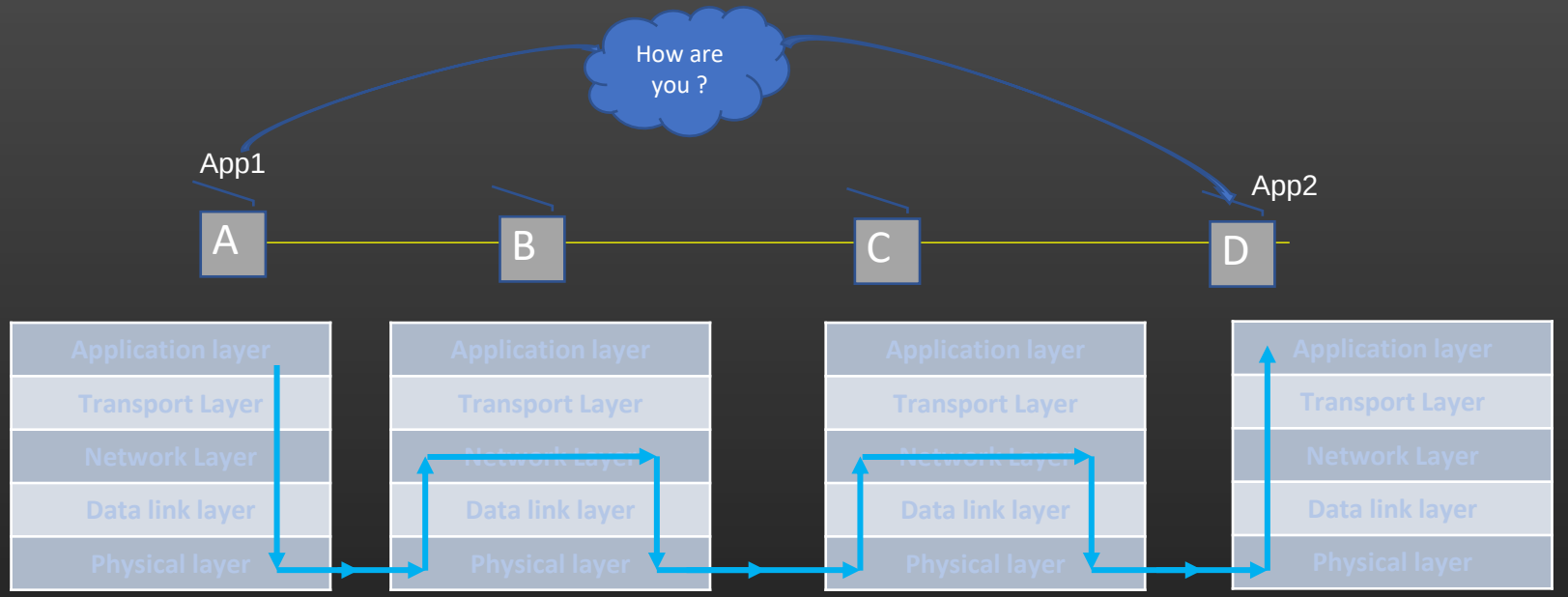


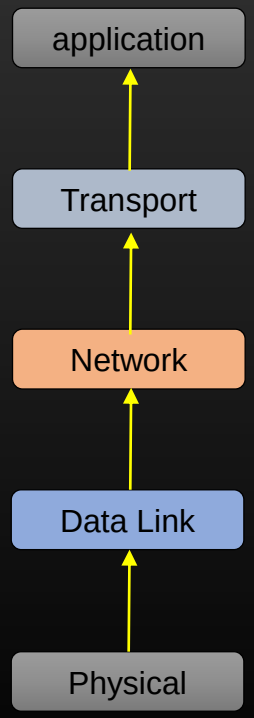


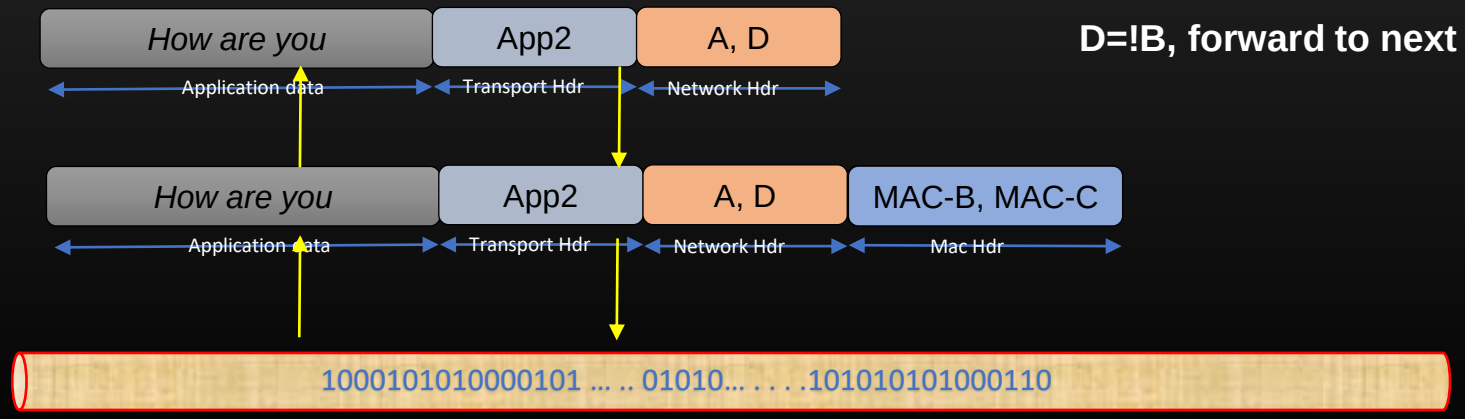
Lecture VDO 1

OSI Model

- **Data En-De-capsulation on B (Intermediate machine)**





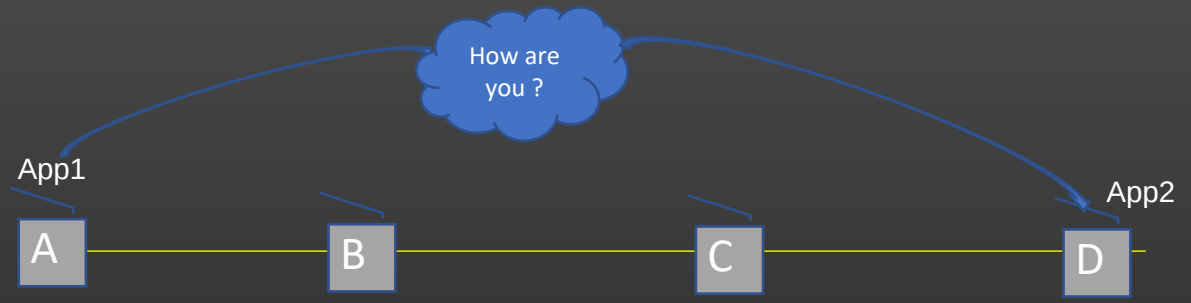


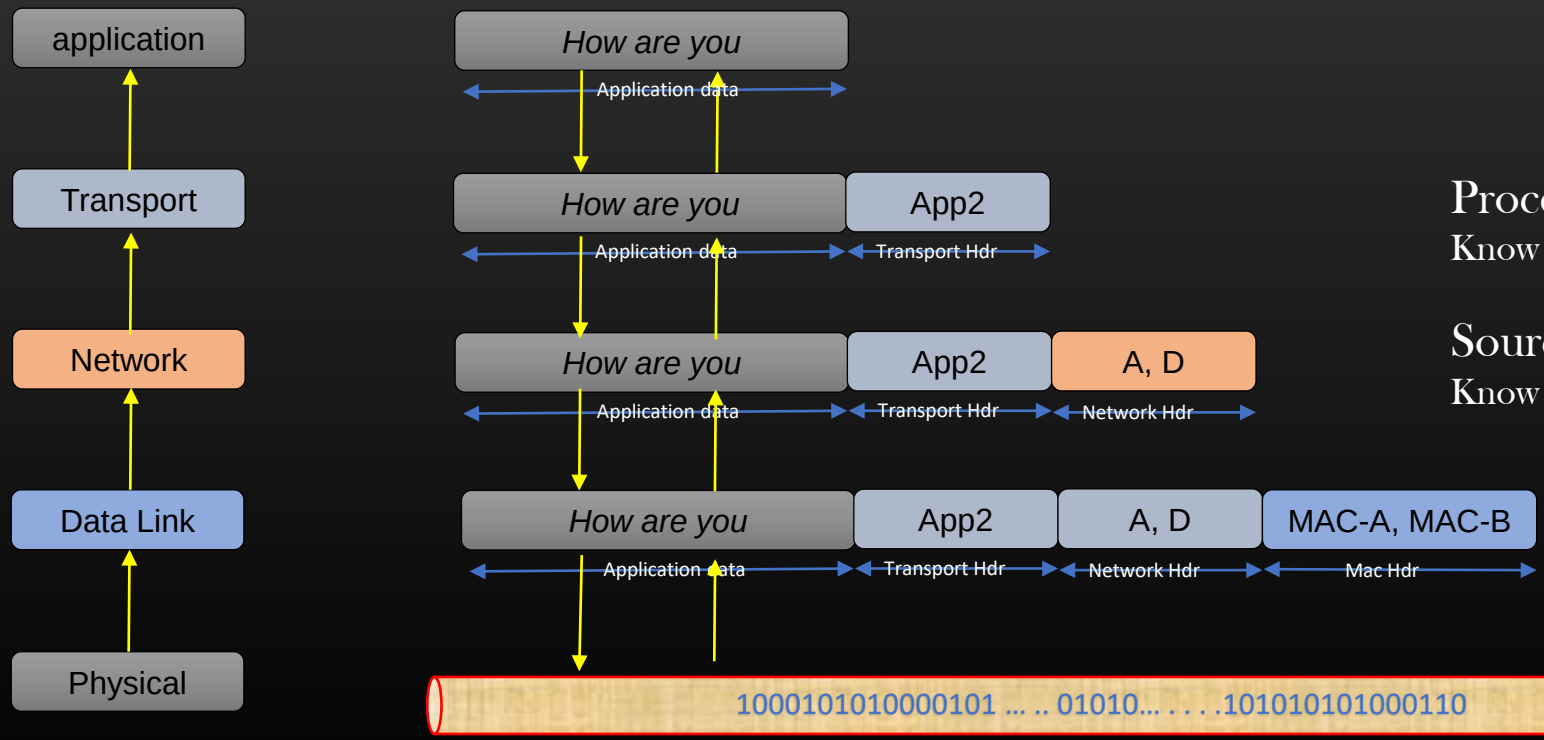
D=!B, forward to next hop towards D

Lecture VDO 1

OSI Model

- **Layer functions**





Process to process delivery
Know about App2, but do not know about D

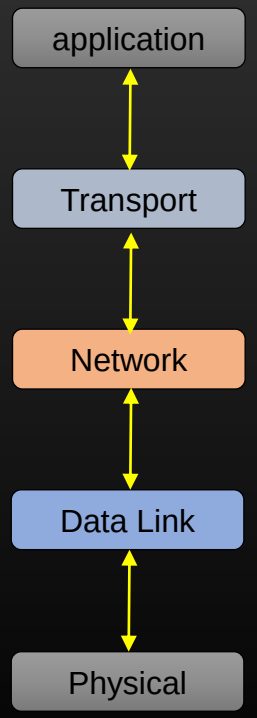
Source to Destination Delivery
Know about D & nexthop, but do not know about App2

Hop by Hop traversal
Know only nexthop,
Do not know about App2 and D

Lecture VDO 1

OSI Model

- **Layer functions**



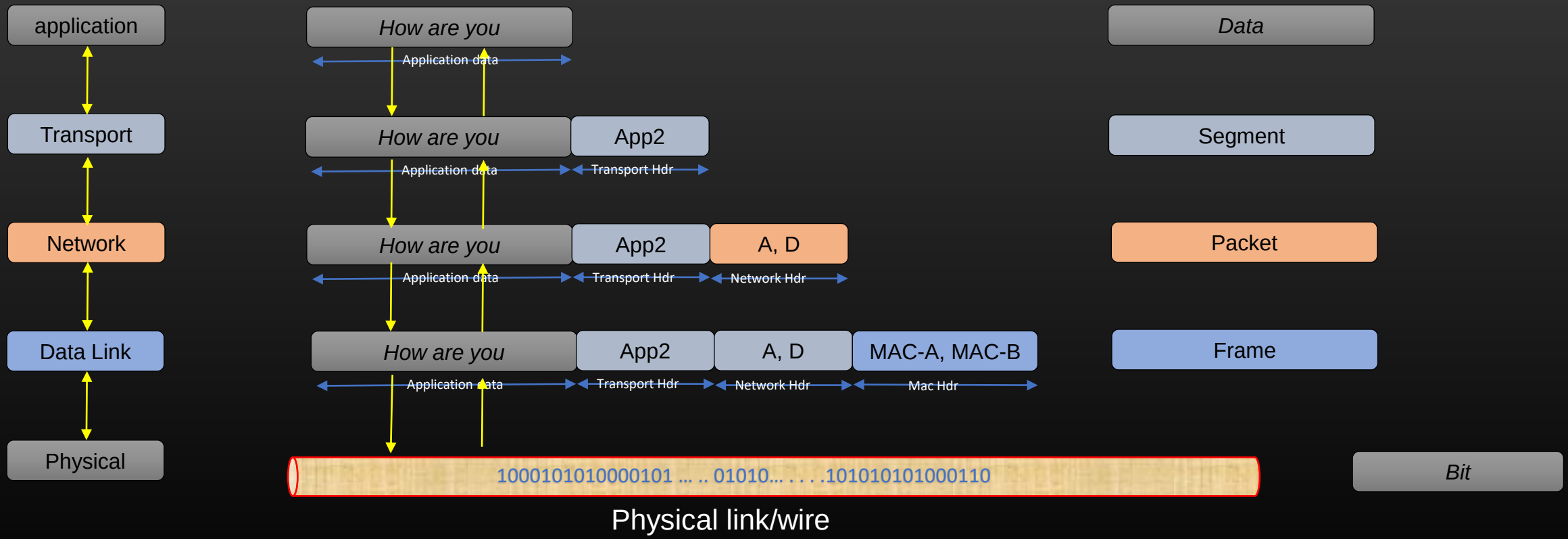
1. Every layer has its own well defined function to perform
2. All Layers work closely to make the packet journey happen from App1 to App2
3. Responsibilities of each layer is well defined, and isolated from each other
4. Each layer hides its own information which is abstracted away from the layer above and below it

   For example, Network layer doesn't need to know anything about App2 (transport layer information) and next hop B (Mac layer info)

Lecture VDO 1

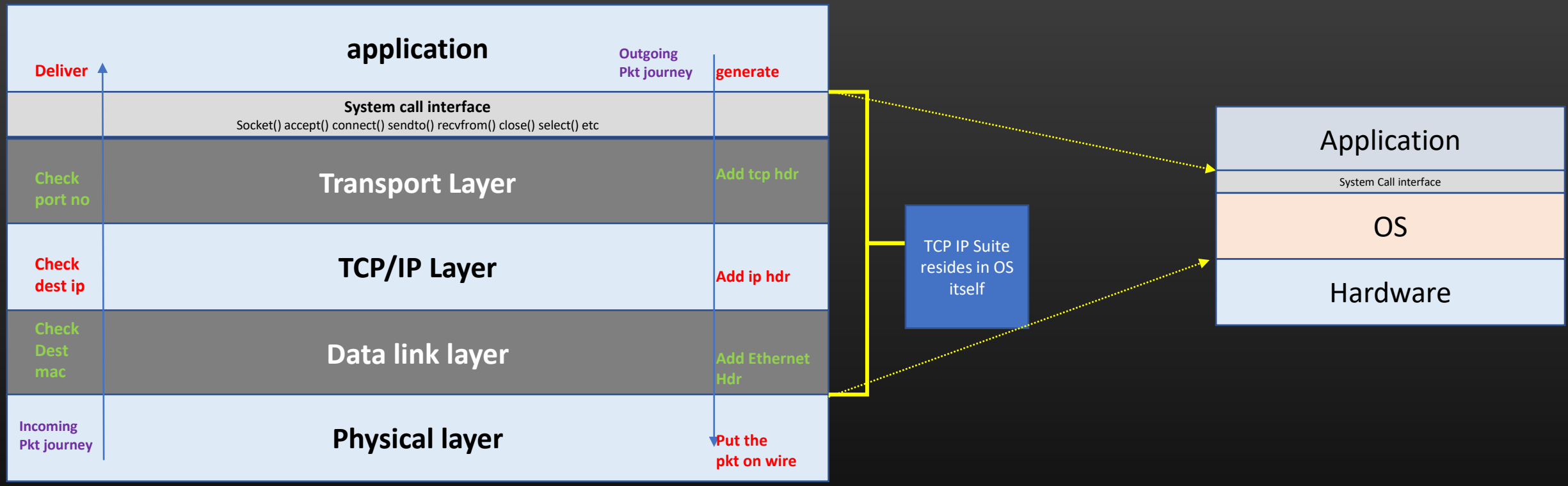
OSI Model

- **Layer specific terminology**



Lecture VDO 1

OSI Model



OSI Model | TCP/IP stack

Lecture VDO 2

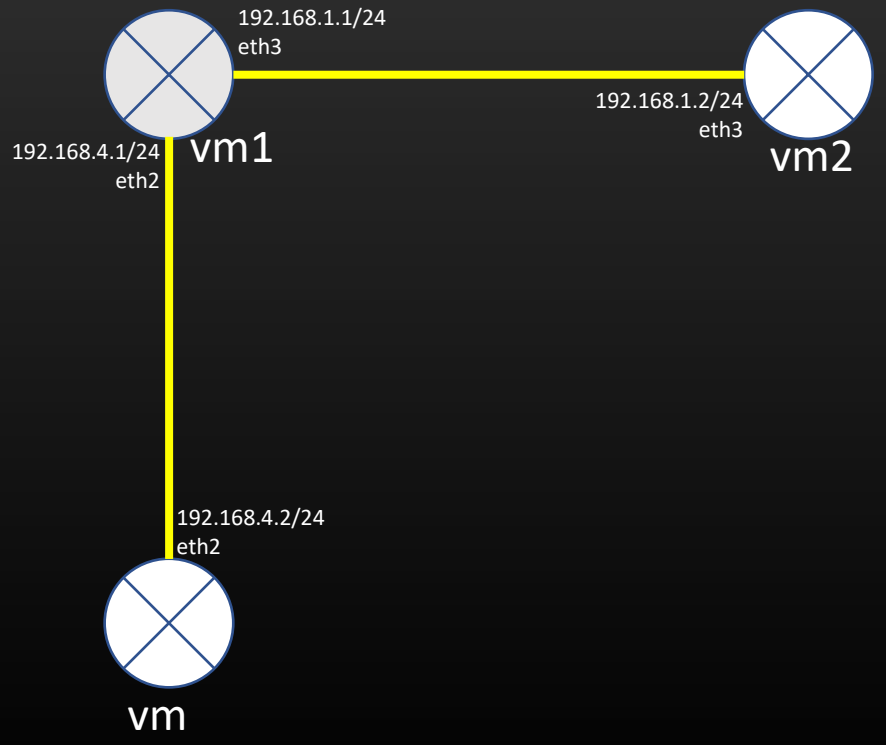
Multi node Setup environment

- **Table of Contents**

Lecture VDO 2

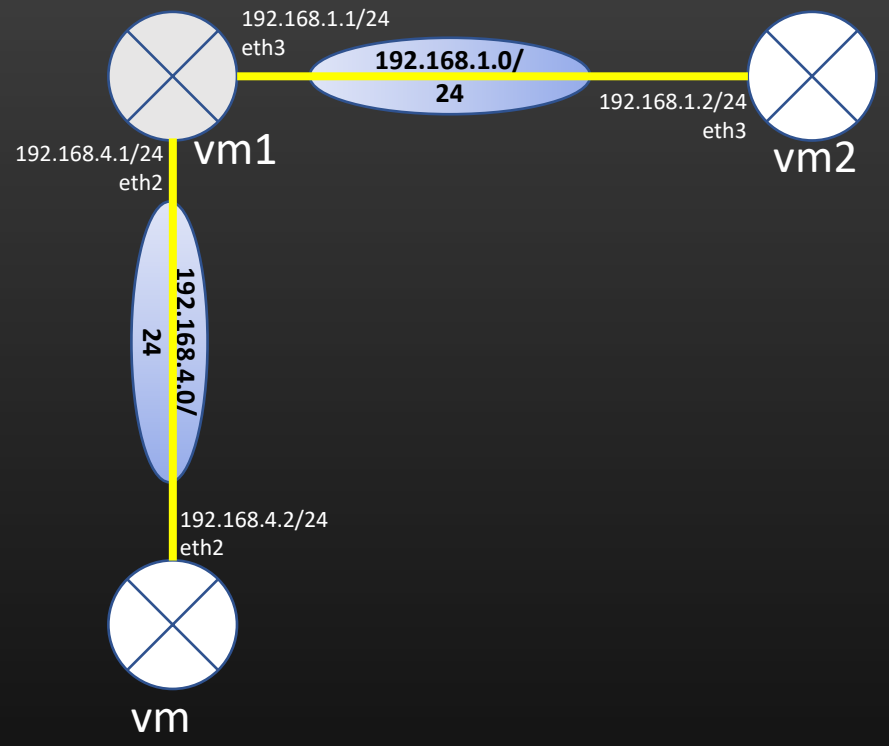
Multi node Setup environment

- In Networking, you often need multiple machines/nodes to test your code, create a network topology
- We are not very rich, neither stupid.
- Let create a multi-node topology of routers on our single machine with the help of Virtualization software – Virtual box
- We will create a 3 node topology



Lecture VDO 2

Multi node Setup environment->local and remote subnets

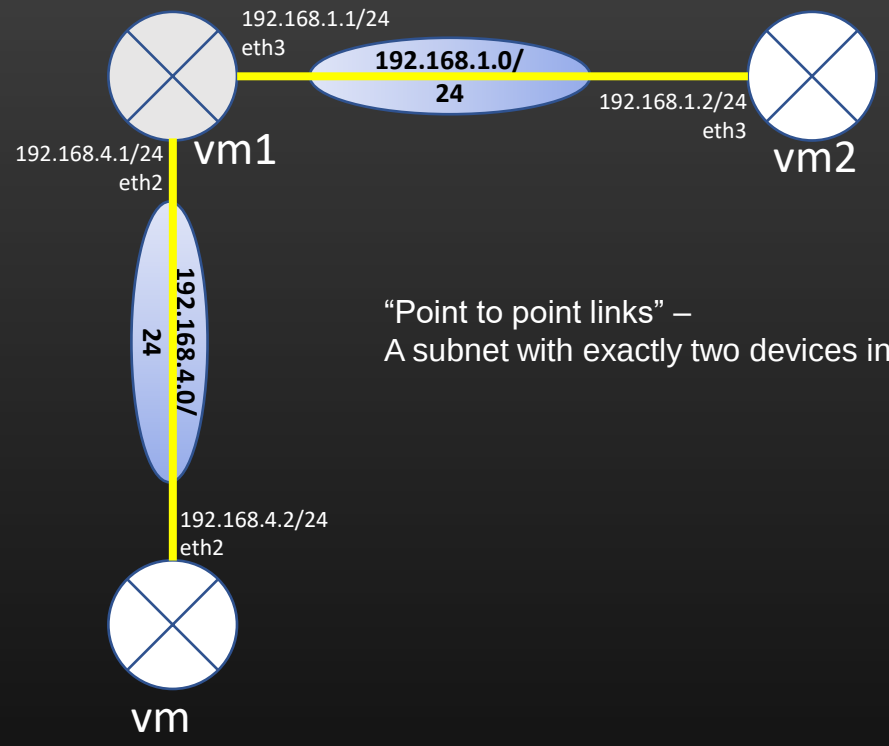


Subnet basics

- Every interface's ip address lies in one subnet
- A subnet has network ID
- Apply mask on an ip address , we get network ID
- Ends of point to point links are always in same subnet
- We will study subnetting and IP address maths in detail in next module, for now just fast introduction so that you understand how to configure and setup multi node environment

Lecture VDO 2

Multi node Setup environment->local and remote subnets



"Point to point links" –
A subnet with exactly two devices in it

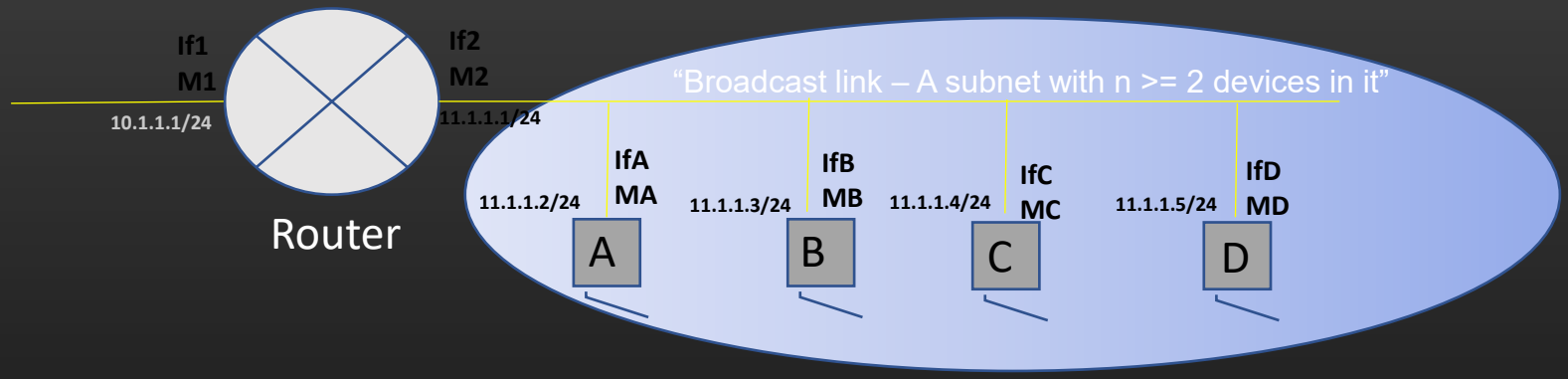
Understand Local and Remote Subnet

- A set of interface of devices interconnected with one another AND has ip addresses configured such that all of them falls have same network id forms one subnet
- A subnet can be local Or remote relative to a given router
- A subnet A.B.C.D/M is said to be remote subnet relative to router X if None of the local interface of a router has ip address configured which falls in A.B.C.D/M subnet
- A subnet A.B.C.D/M is said to be local subnet relative to router X if at-least one of the local interface of a router has ip address configured which falls in A.B.C.D/M subnet
- Loopback subnet (A.B.C.D/32) is always a local subnet for the local router and remote subnet for any other non-local router (Remember this point, we get back to this later when we will study loopbacks in detail)

Router	Local Subnet	Remote subnet
VM	192.168.4.0/24	192.168.1.0/24
VM1	192.168.4.0/24 192.168.1.0/24	-
VM2	192.168.1.0/24	192.168.4.0/24

Lecture VDO 2

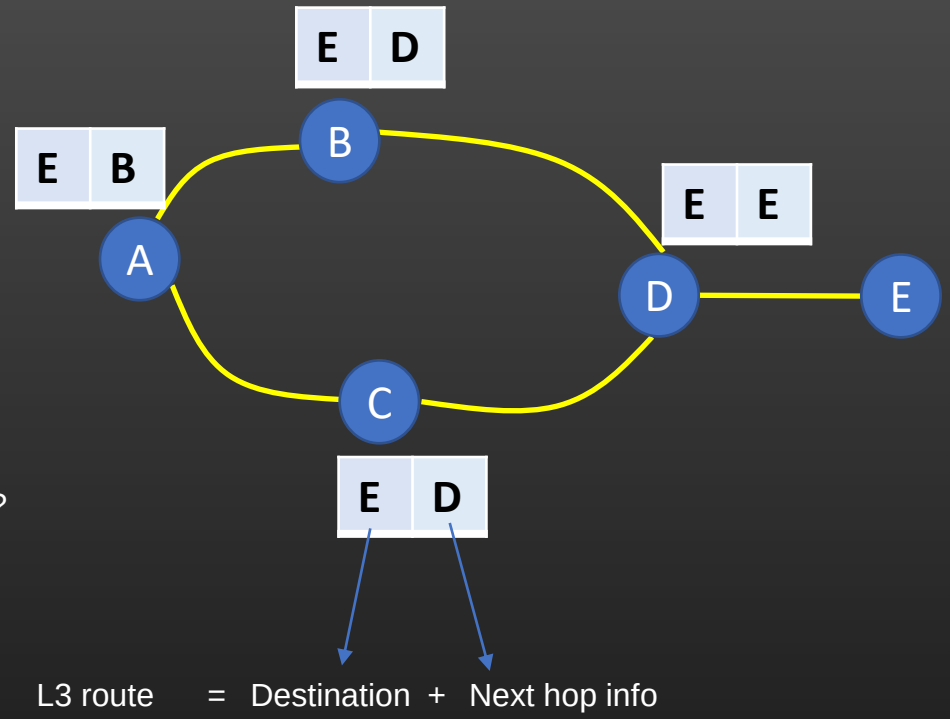
Multi node Setup environment->local and remote subnets



Router/machines	Local Subnet	Remote subnet
Router	10.1.1.0/24 11.1.1.0/24	-
A	11.1.1.0/24	10.1.1.0/24
B	11.1.1.0/24	10.1.1.0/24
C	11.1.1.0/24	10.1.1.0/24
D	11.1.1.0/24	10.1.1.0/24

Lecture VDO 2

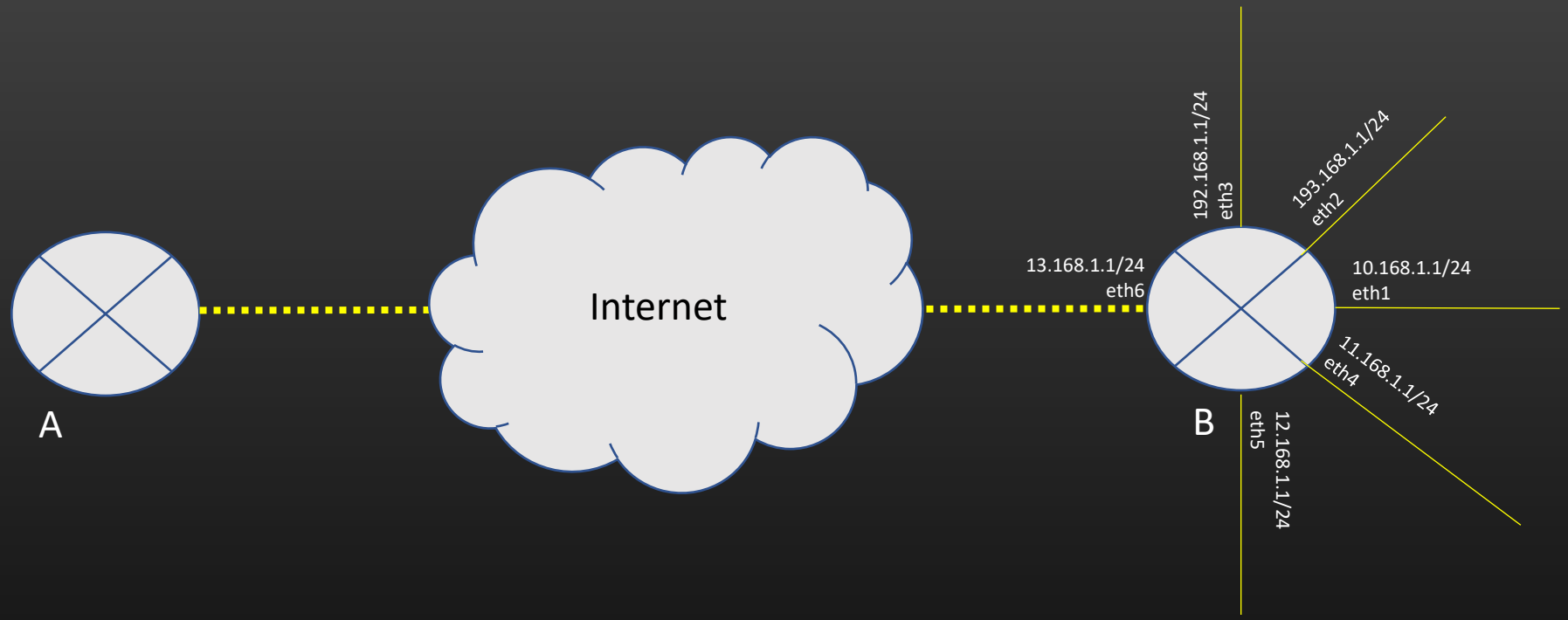
Multi node Setup environment->L3 routing configuration



- What will Router A do if it receives a packet destined for Router E ?
- Will Router A forwards the packet to B or C ?
- How A would know that in order to forward the data to Destination D , it has to forward the packet to B Or C. Which will be better path ?
- We need to feed rules in every L3 router in the network which defines the behavior of a L3 router to how to forward the packet destined to remote Destinations
- These set of rules are called “L3 routing information”
- Without L3 routing information, Routers which are more than one hop away cannot talk to each other

Lecture VDO 2

Multi node Setup environment->L3 routing configuration



B's local subnets
192.168.1.0/24
193.168.1.0/24
10.168.1.0/24
11.168.1.0/24
12.168.1.0/24

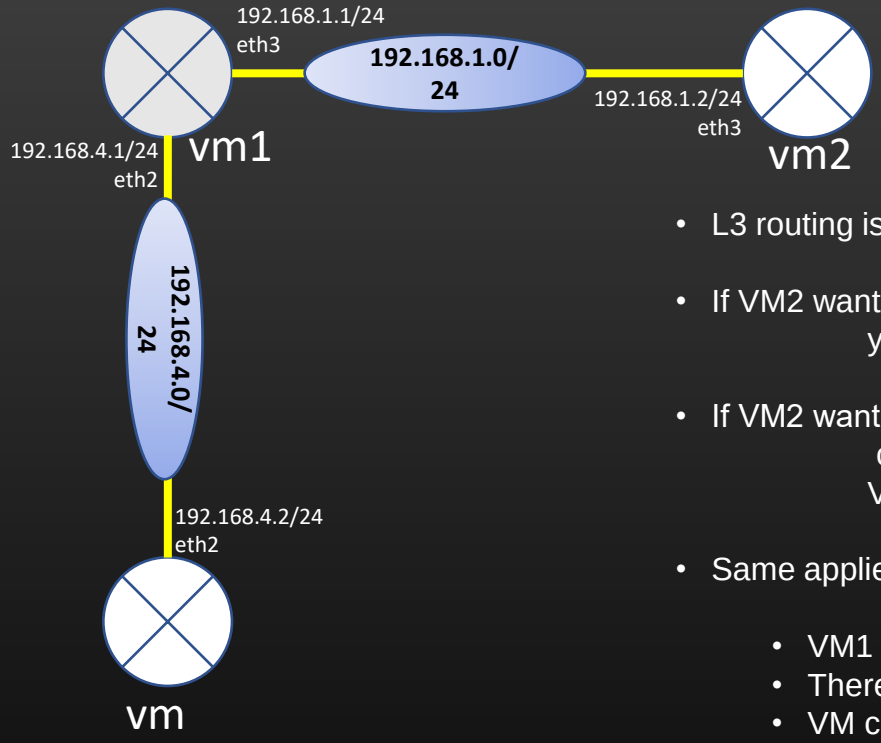
- Whenever we say Router/Machine B is reachable from router/machine A it means, A has a routing information to reach at-least one local subnet of B

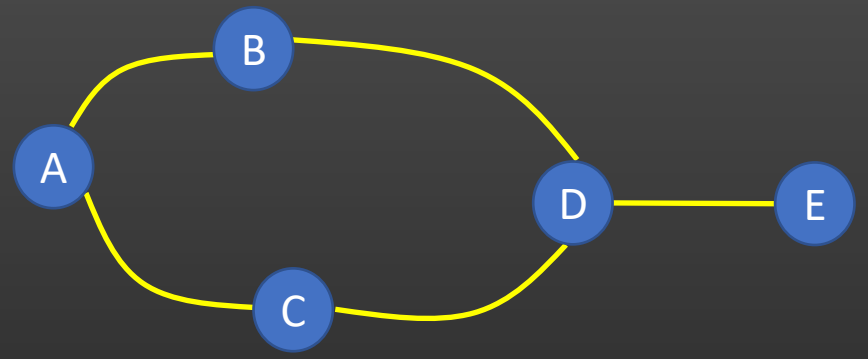
- A can reach B as long as A knows the next-hop information to reach Destination D, where D Is any local subnet of B

- L3 routing is not about how to reach a particular Machine, but its all about how to reach a particular subnet

Lecture VDO 2

Multi node Setup environment-> L3 routing configuration

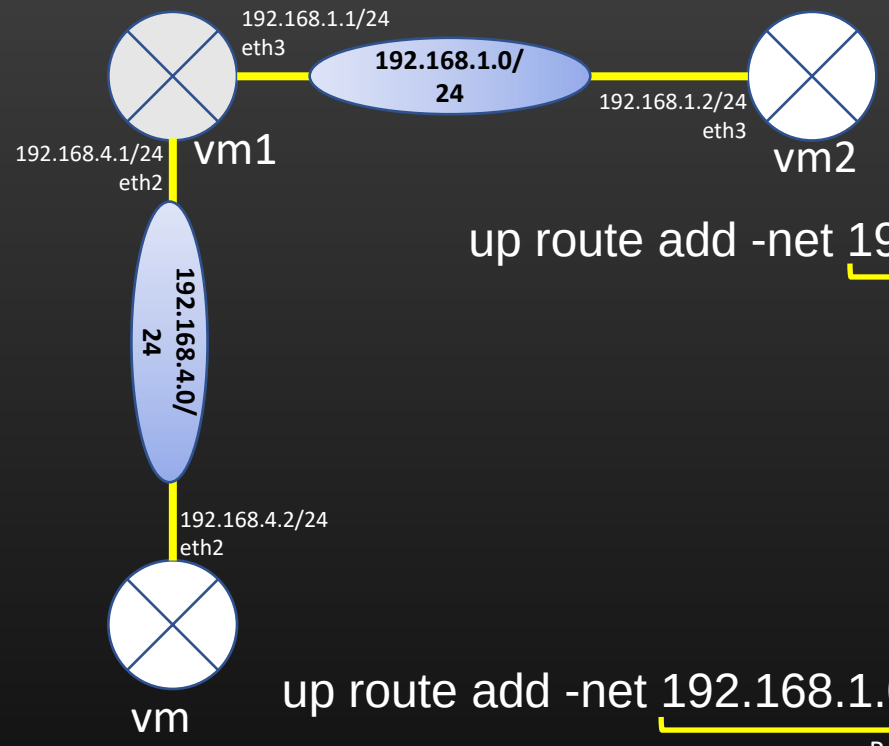




- L3 routing is not about how to reach a particular Machine, but its all about how to reach a particular subnet
- If VM2 wants to talk to VM, then VM2 should know how to reach at-least one VM's local subnet. This is same as if your uncle wants to visit to your house, he should know atleast one path to your house.
- If VM2 wants to talk to VM (read it as "if VM2 wants to reach one of local subnets of VM"), VM2 has to be configured with L3 routing info which tells how to reach VM (its one of local subnet) because VM and VM2 are in different subnets
- Same applies if VM wants to talk to VM2
    - VM1 shares a local subnet with VM2 and VM1.
    - Therefore, VM1 can talk to VM as well as VM2
    - VM can talk only with VM1 but not with VM2 unless L3 route to VM2 is configured on VM
    - VM2 can talk only with VM1 but not with VM unless L3 route to VM is configured on VM2
    - Thus machines are to be configured with L3 routing information to enable them talk with other machines which are present in remote subnets (more than one hop away)
- L3 route simply says – To reach a remote subnet X, Nexthop is Y.
- Note that : L3 routes dictates how to reach REMOTE subnets
- L3 routing configuration is not required to reach LOCAL subnets

Lecture VDO 2

Multi node Setup environment-> L3 routing configuration



up route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.1.1 dev eth3

Remote Subnet : 192.168.4.0/24

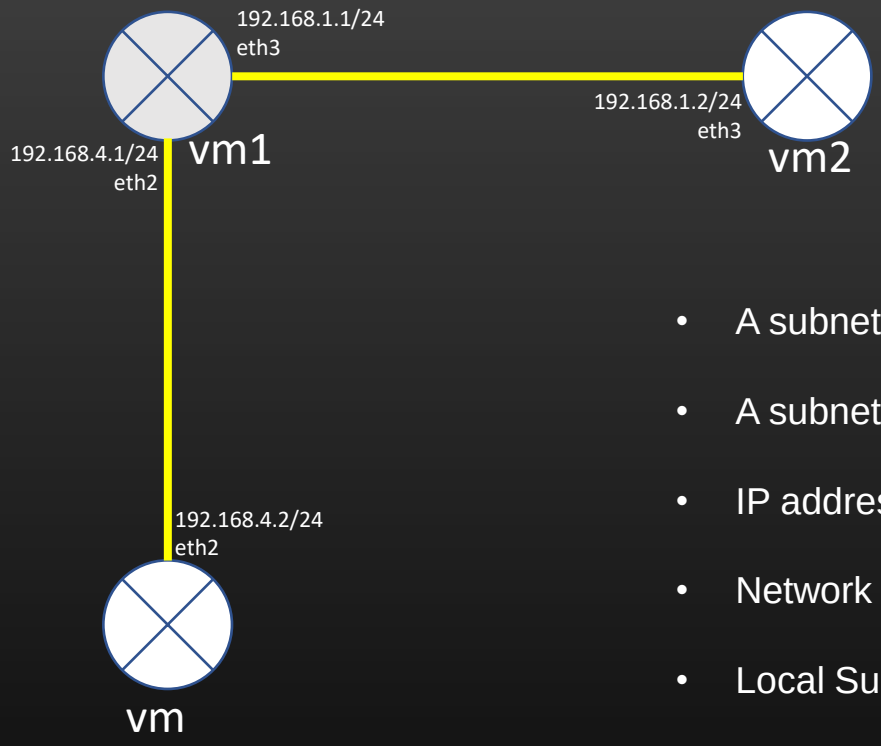
up route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.4.1 dev eth2

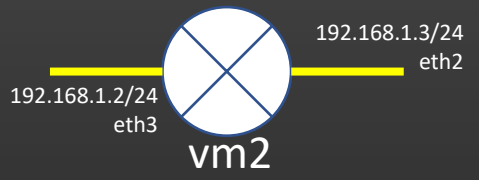
Remote Subnet : 192.168.1.0/24

- We always install L3 route in a router For a remote subnet
- No need to install L3 route to reach local subnet (Why somebody has to tell you the path to your own home ??)
- No need to install any L3 route in VM1 as all known subnets in this topology are local to VM1

Lecture VDO 2

Multi node Setup environment -> L3 table entries



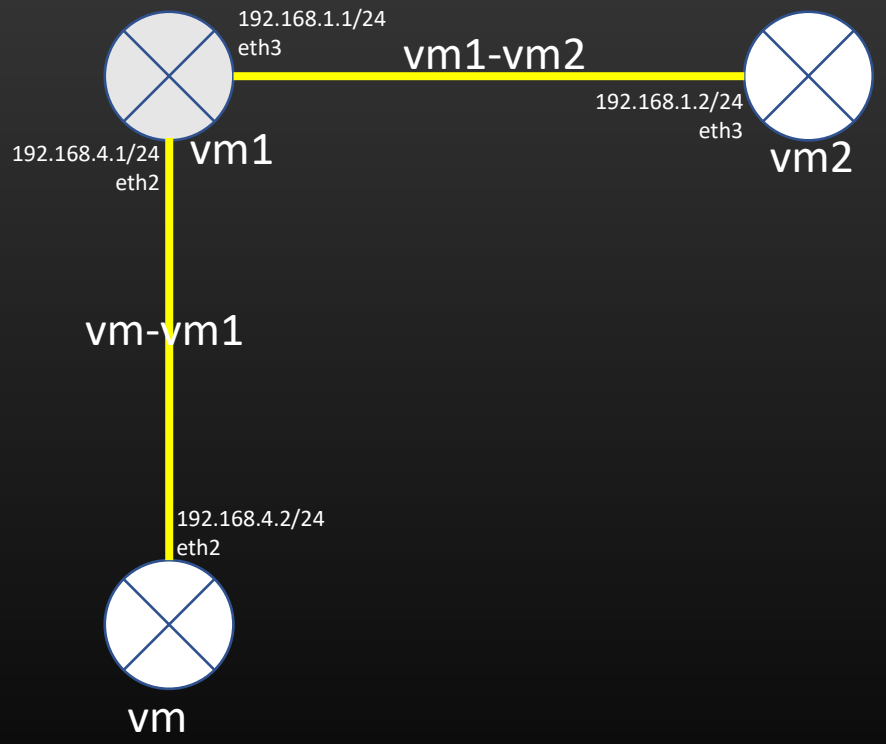


- A subnet is a collection of devices connected with each other in the same network
- A subnet has an ID called network ID
- IP addresses of all devices present in the same subnet have same network ID
- Network ID of a subnet = IP address of any device present in the subnet & mask
- Local Subnet and Remote Subnet
- Interfaces has IP addresses, not machines
- Machine’s multiple Interfaces MUST belong to different subnets
- No two or more interfaces of a single devices can belong to same subnet

Lecture VDO 2

Multi node Setup environment -> practical session

- We will going to create Multi node Topology environment now
- We will create 3 node topology as shown below



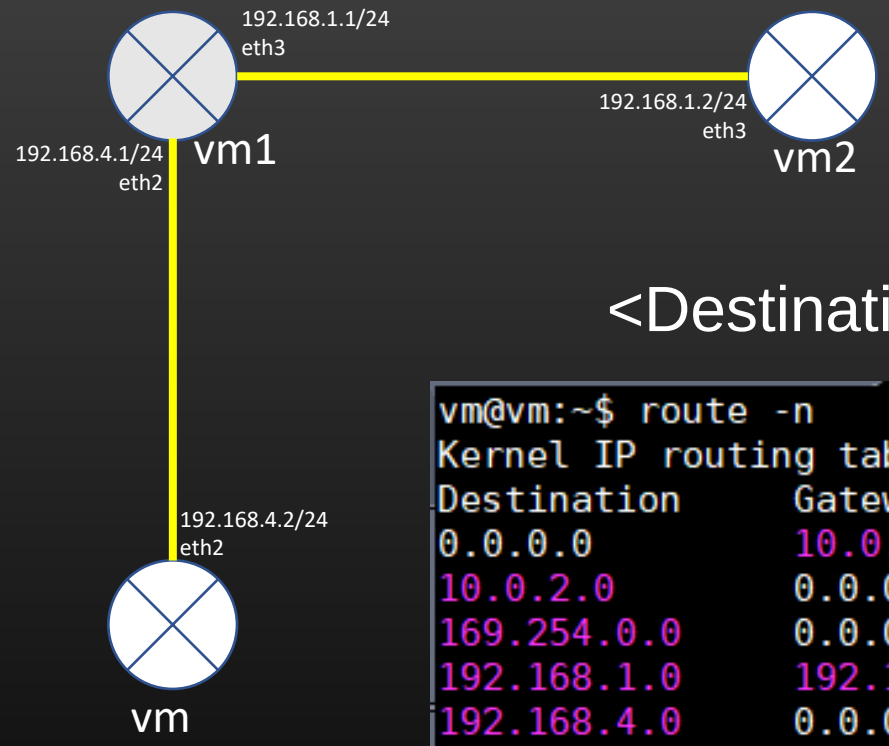
Lecture VDO 2

Multi node Setup environment -> practical session

- Now install L3 routes in VM and VM2 so that they can talk to each other
- Test reachability using ping

Lecture VDO 2

Multi node Setup environment -> L3 table entries



<Destination/mask> <Next hop IP> <OIF>

```
vm@vm:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    0      0        0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U     1      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth2
192.168.1.0     192.168.4.1     255.255.255.0   UG    0      0        0 eth2
192.168.4.0     0.0.0.0         255.255.255.0   U     0      0        0 eth2
192.168.56.0    0.0.0.0         255.255.255.0   U     0      0        0 eth1
vm@vm:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.56.1             ether   0a:00:27:00:00:03   C                     eth1
192.168.4.1              ether   08:00:27:46:dd:8d   C                     eth2
vm@vm:~$
```

Lecture VDO 2

Multi node Setup environment

- In this module we had a quick tour on how to setup multi node environment on a single machine using Virtual box
- We learnt assigning ip addresses and configuring static routes
- We were able to ping machines which are more than one hop away
- We noticed the routing table and arp table entries also.
- We learned the criteria for identifying local and remote subnets relative to a given machine
- This was just a quick tour to make you familiar with networking, I am sure you have lot of un-clarity at this point of time such as :
  - What should be the ip address of local interface and other end interface of the link ?
  - What is subnet mask, how this works ? Why /24 and not /28 or /20 or . . . So on . . .
  - Why not configure routes for local subnets, and configure routes for remote subnets ?
  - What are the semantics of routing table entry ?
  - What does Arp table do ?
  - And so on . . . .

  Let deep dive and learn fundamentals of Networking which often confuses most students in a systematic manner. . . .

Lecture VDO 3

Subnetting & IP address Maths

- **Table of Contents**

Lecture VDO 3

Subnetting & IP address Maths

- We will take a deep dive in L2 and L3 routing
- L3 Routing means – Routing done by Layer 3, the network layer
- L2 routing means – Routing done by Layer 2, the Data Link Layer
- If you are appearing for Networking interview, 100% are the chances that you will be asked to explain Routing
- You must be thorough with the L2 and R3 routing concept – Any doubt – Revise again and again
- It is not a lengthy topic, but deep enough and is the heart of Networking
- Entire Networking revolves around ***Routing***
- There are hundreds of Tutorials, youtube VDOs all over internet on this topic, yet students find it very confusing
- We will pick you from absolute ground zero level, and build up your routing concepts incrementally
- We will do Demonstration of actually routing a packet on the network topology, observe all relevant routing table entries
- Lets begin ..

Lecture VDO 3

Subnetting & IP address Maths

- dividing a network into two or more networks is called subnetting
  like dividing a country into smaller islands – For example a Mauritius islands
  like dividing high school students into separate batches – X-A, X-B etc

- A Subnet is a Layer 2 Network, that is Data Link layer takes care of data transfer from one machine to another machine in the same subnet

- As long as communicating machines are present in the same subnet, Only Data link Layer is responsible for Data delivery

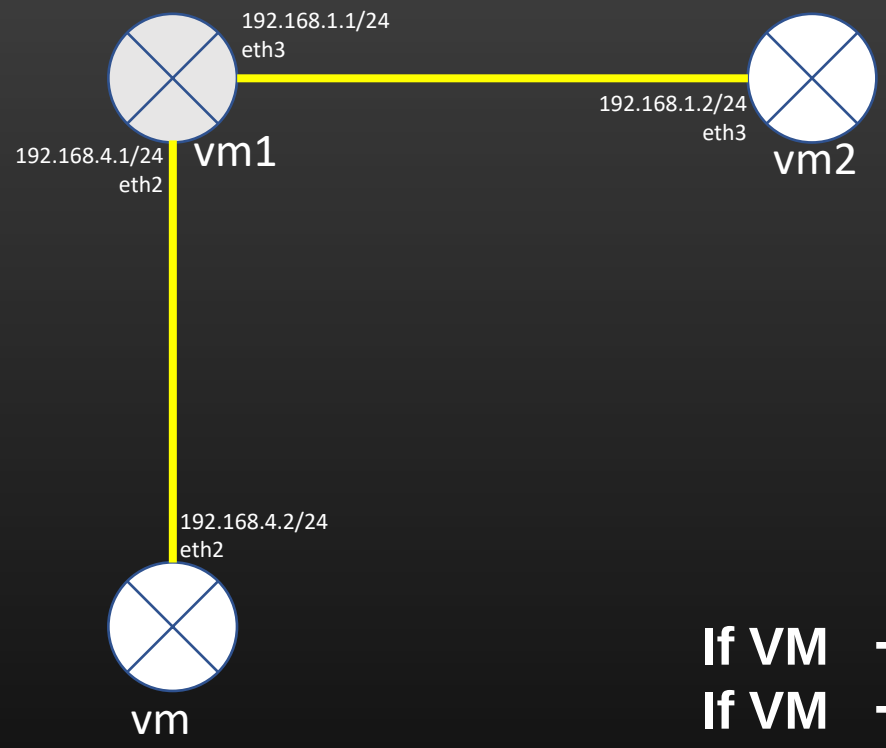
- If communicating machines are present in different subnetwork, Network layer is responsible to take the data from one subnet to another

Same Subnet = Data Link Layer

In different Subnets = Network Layer

Lecture VDO 3

Subnetting & IP address Maths



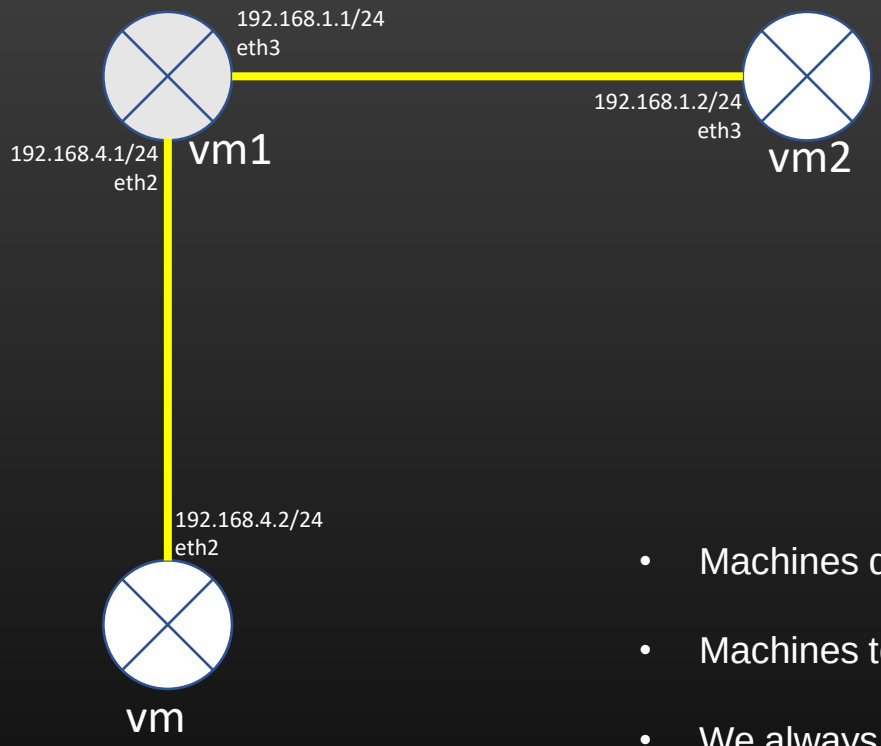
Same Subnet = Data Link Layer

In different Subnets = Network Layer

If VM → Data → 192.168.1.1 Or 192.168.1.2 - L3 routing
If VM → Data → 192.168.4.1 - L2 routing

Lecture VDO 3

Subnetting & IP address Maths -> IP Header



IP Header

Bit Number

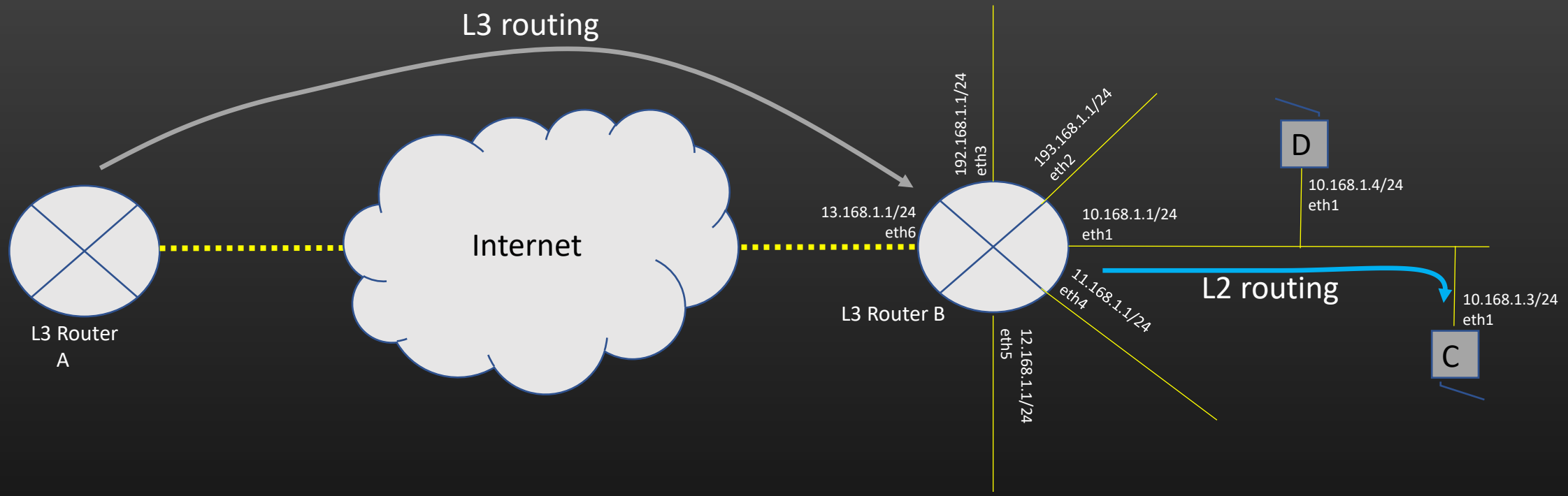
1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Version	IHL	Type of Service	Total Length
Identification		Flags	Fragment Offset
Time to Live	Protocol	Header Checksum	
Source Address			
Destination Address			
Options (optional)			

- Machines do not have IP addresses, Only interfaces have
- Machines to not have Mac addresses, Only Interfaces Have
- We always send Data destined to a particular interface of a remote machine, and not a subnet

Lecture VDO 3

Subnetting & IP address Maths -> Destination IP Address



- If A sends Data with dst ip address = X, where X is 192.168.1.1, 193.168.1.1, 10.168.1.1, 11.168.1.1, 12.168.1.1, then B will consume the data
- If A sends the data with dst ip address = X, where X = 10.168.1.3, then B receives the data and then use L2 routing to send the data to machine C
- L3 routers talk with each other via L3 routing only, L3 routers talk with machines present in its local subnet with L2 routing
- Each interface of L3 router is a subnet

Lecture VDO 3

Subnetting & IP address Maths -> Destination IP Address

Summarization

- If receiving machine receives a packet in which dst ip address exactly matches with the ip address of one of its local interface, receiving machine consumes the packet

- If receiving machine receives a packet in which dst ip address DO NOT matches exactly with the ip address of one of its local interface, but falls in one of its local subnet, machine forwards the packet onto that subnet using L2 routing (More on this when we will be using L2 routing in detail)

- If none of the above Conditions are true, Machine forwards the packet using L3 routing to next router Or Drops the packet (More on this When we will study L3 routing in detail)

Lecture VDO 3

Subnetting & IP address Maths -> Destination IP Address

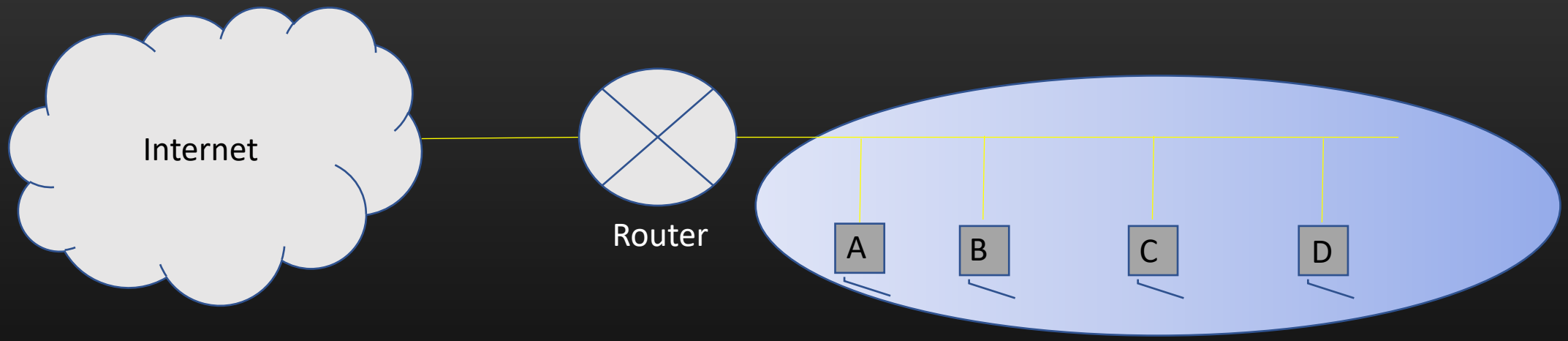
What does Layer 3 router do when it receives a packet ?

Layer 3 Criteria	Action
IP address = exact ip address of one of local interface	Consume the packet
IP address falls in one of the local subnet	Forwards the packet using L2 routing
IP Address falls in remote subnet and L3 routing entry for that remote subnet exists in routing table	Forward the packet using L3 routing
IP Address falls in remote subnet and L3 routing entry for that remote subnet do not exists in routing table	Drop the packet

Lecture VDO 3

Subnetting & IP address Maths -> Broadcast Domain

- How does the Subnet of devices looks like ??
- A subnet is always present behind the L3 Router



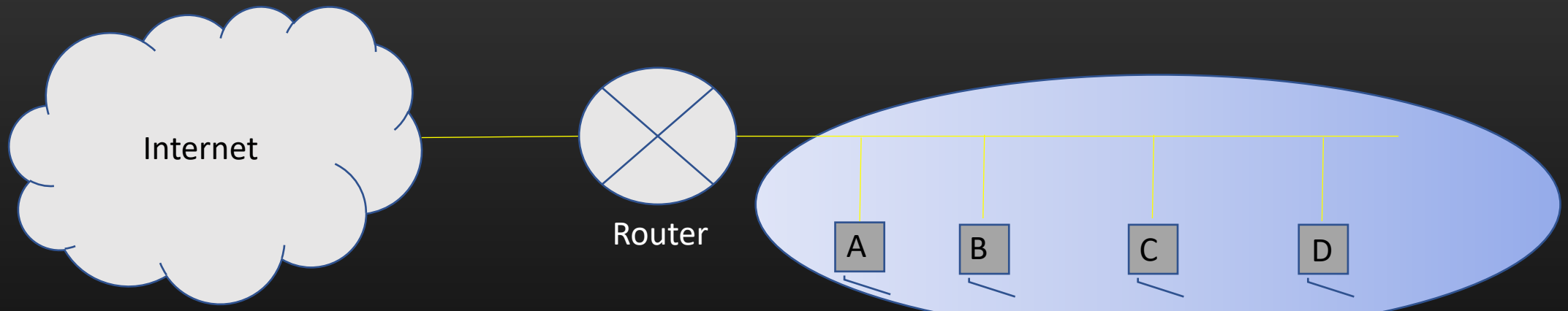
- All devices connected with the same wire (broadcast domain) forms one subnet
- A packet sent by one machine is heard by all other machines in the same subnet

- Example – All machines in your college campus may form one subnet, sitting behind the one main router

Lecture VDO 3

Subnetting & IP address Maths -> Mac address

- Within a Subnet, data transmission across machines is done using only MAC addresses
- Thus, Data link layer operates with Mac addresses only, and nothing to do with IP addresses
- Mac addresses are also called Layer 2 Addresses, IP addresses are also called Layer 3 addresses

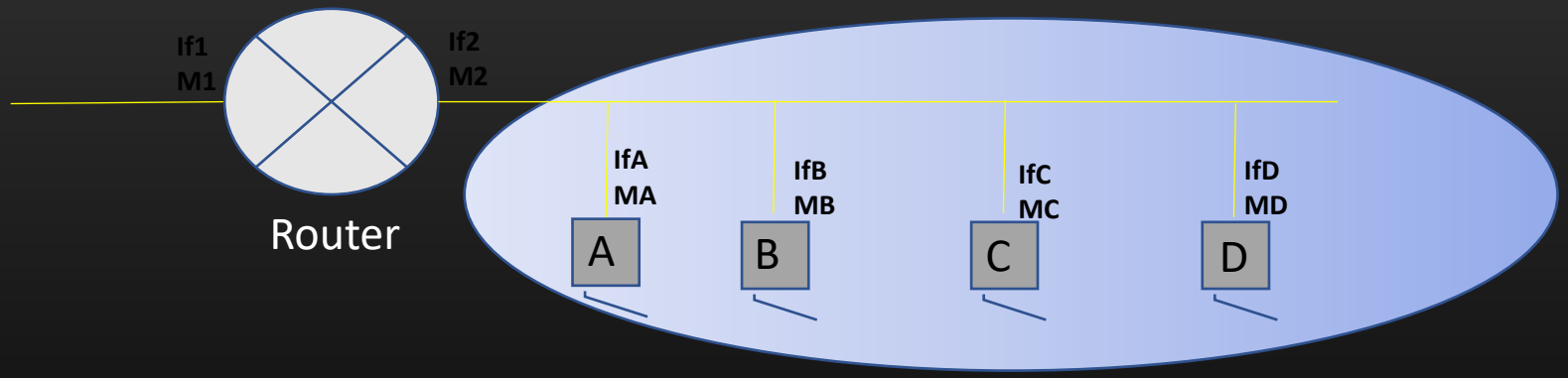


- Who has Mac addresses ? Who assigns ? How can we configure ?
- How does mac addresses looks like ?

Lecture VDO 3

Subnetting & IP address Maths -> Mac Addresses

- Every interface through which a device is connected Has a Mac addresses
- It is assigned by Device manufacturer, and is globally unique. It is burnt in the Hardware, and you cannot change it
- As many interfaces, as many mac addresses
- You can check the mac/ip addresses of your VM Router using ***ifconfig*** command



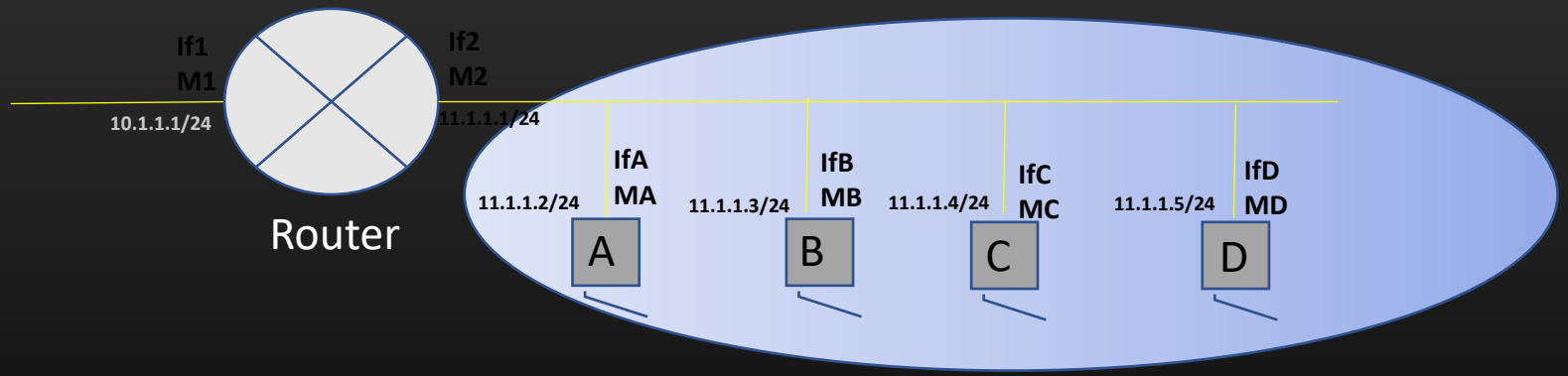
60:A6:56:E3:5C:59

Binary	01100000	10100110	01010110	11100011	01011100	01011001
Hex	60	A6	56	E3	5C	59

Lecture VDO 3

Subnetting & IP address Maths -> IP Address and Subnet Mask

- Every interface through which a device is connected Has an IP address and a subnet mask also
- Unlike Mac addresses, IP/Mask on an interface is configurable and you can change it
- As many interfaces, as many IP addresses
- You can check the mac/ip addresses of your VM Router using ***ifconfig*** command



Binary	01100000	10100110	01010110	11100011
Hex	60	A6	56	E3
Dec	60	166	56	227

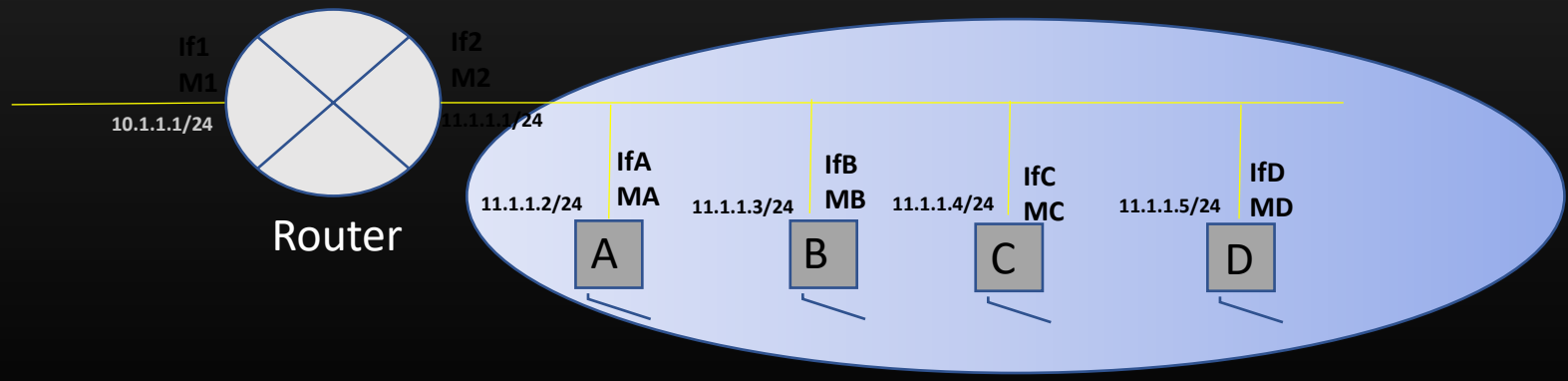
Lecture VDO 3

Subnetting & IP address Maths -> Understanding IP address Maths

- Every IP address is associated with a Mask , a value [0,32]
- Mask is used to form Subnetworks
- Every Subnet has following :
  - Network ID
  - Broadcast Address
  - Max no of Machines which could be present in the subnet
- We can calculate all the above subnet values from IP/MASK of any machine present in the subnet
- For Ex : Machine B –

  IP = 11.1.1.3 , and Mask = 24

  - Subnet/Network ID = 11.1.1.0/24
  - Broadcast address = 11.1.1.255
  - Max N = (2 ^ (32-24)) – 2 = 254



Lecture VDO 3

Subnetting & IP address Maths -> Understanding IP address Maths

- For Ex : Machine B –

  IP = 11.1.1.3 , and Mask = 24

  - Subnet/Network ID = 11.1.1.0/24
  - Broadcast address = 11.1.1.255
  - Max N = (2 ^ (32-24)) – 2 = 254

Steps to calculate **Subnet id** from IP/MASK of a machine B :

1. Represent 11.1.1.3 in Binary format

00001011	00000001	00000001	00000011

2. Create a binary no B from mask such that Ist 24 bits are all 1s, remining bits are 0s
   B is – 11111111 11111111 11111111 00000000

3. Performing AND of Binary No obtained in step 2 with IP address in step 1, we get

00001011	00000001	00000001	00000000

4. Convert the IP address obtained in step 3 into Decimal notation, we get
   11.1.1.0/24 which is our network ID

Lecture VDO 3

Subnetting & IP address Maths -> Understanding IP address Maths

- For Ex : Machine B –

  IP = 11.1.1.3 , and Mask = 24

  - Subnet/Network ID = 11.1.1.0/24
  - Broadcast address = 11.1.1.255
  - Max N = (2 ^ (32-24)) – 2 = 254

Steps to calculate **Broadcast address** of a subnet from IP/MASK of a machine B :

1. Obtain the Network ID first

2. Find M' = Complement of Mask M = 24
   M' is – 00000000 00000000 00000000 11111111

3. Performing OR of Binary No obtained in step 2 with Network ID address obtained in step 1, we get

00001011	00000001	00000001	11111111

4. Convert the IP address obtained in step 3 into Decimal notation, we get
   11.1.1.255/24 which is our network Broadcast address

Lecture VDO 3

Subnetting & IP address Maths -> Understanding IP address Maths

- For Ex : Machine B –

  IP = 11.1.1.3 , and Mask = 24

  - Subnet/Network ID = 11.1.1.0/24
  - Broadcast address = 11.1.1.255
  - Max N = (2 ^ (32-24)) – 2 = 254

Mask 24 = 11111111 11111111 11111111 00000000

Control bits

Steps to calculate **Max** no of machines in a subnet from IP/MASK of a machine B :

1. Given Mask = 24
2. Max = (2 ^ (32-24)) – 2 = 254

Precisely saying – Its not “max no of machines”, but “maximum no interfaces” connected to our subnet.

Lecture VDO 3

Subnetting & IP address Maths -> Understanding IP address Maths

- For Ex : Machine B –

  IP = 11.1.1.3 , and Mask = 24

  - Subnet/Network ID = 11.1.1.0/24
  - Broadcast address = 11.1.1.255
  - Max N = (2 ^ (32-24)) – 2 = 254

Mask 24 = 11111111 11111111 11111111 00000000

Control bits

Generating Assignable IP addresses for subnet devices

1. Take control bits only and Perform LOGICAL OR operation with Network ID. It gives you the assignable IP address for a subnet

2. Repeat step 1 for different permutation of control bits

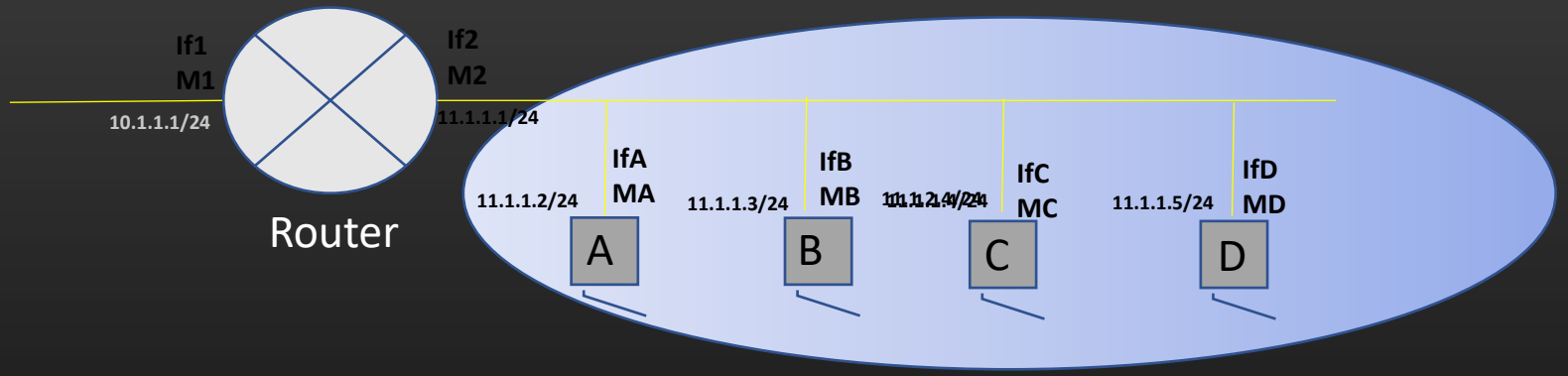
Note : We have subtracted two because – one address is Network ID itself, and other is broadcast address which we already calculated
We Cannot assign network Address and Broadcast address to any machine in a subnet

Thus, in our example – we can assign IP to machines in our subnet as follows :
11.1.1.1/24, 11.1.1.2/24, 11.1.1.3/24 . . . . . , 11.1.1.254/24 = Total 254 maximum machines can be present in the subnet with mask 24

Precisely saying – Its not “max no of machines”, but “maximum no interfaces” connected to our subnet.

Lecture VDO 3

Subnetting & IP address Maths -> Understanding IP address Maths -> Valid & Invalid Configuration



- What if you change the IP address of C to 11.1.2.4/24 ? Is it allowed ?
  - C's Network ID will ne 11.1.2.0/24 whereas rest of the machines network id is 11.1.1.0/24
  - Thus, C is now present in different Subnet altogether
  - Two subnets must always be separated by a L3 router, remember L3 router is required to route the traffic from one subnet to another
  - In this example, there is no L3 router present between C and rest of the machines
  - Hence, C's data cannot be delivered to any of A,B and D Or router itself
  - You can visualize this scenario as two islands (subnetworks) not connected through sea route
  - Hence, this configuration is illegal

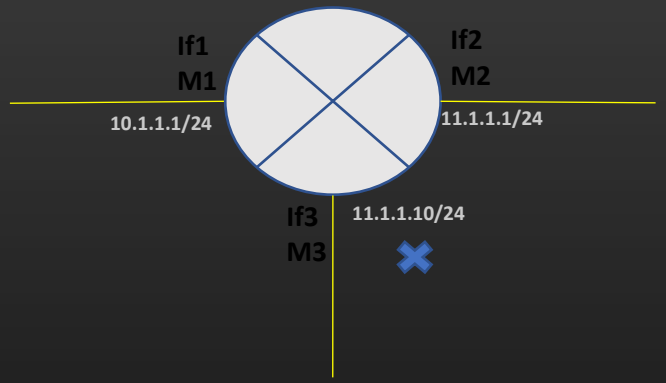
- Conclusion >> Each Router's interface is connected to one subnet (Or another L3 router). Hence, each router interface IP/MASK value must yield unique network id which identifies the subnet on that interface
  >> Any two machines whose IP/MASK yields the same subnetwork ID are said to be present in same subnet

Lecture VDO 3

Subnetting & IP address Maths -> Understanding IP address Maths -> Valid & Invalid Configuration

- The configuration below is illegal. Why ?

Router



- If2 and if3 of a router have the same subnetwork ID – 11.1.1.0/24.
- Hence, the same subnet is actually separated by a L3 router in between !! Illegal !!

Conclusion – We cannot assign IP addresses/Mask which evaluates to same network id to multiple interfaces of an L3 router

View the L3 Router as : Junction of different Subnets

Lecture VDO 3

Subnetting & IP address Maths -> Recommended Mask Value to use for P2P links

- Point to Point Links
  - It is a very common scenario in networking that two devices, say L3 routers, are connected using point to point links

  - The two interfaces of the two routers have the same network ID – hence, they are in same subnet
  - Point to Point ends are always in same subnet !!
  - This is a straight wire which connects the two ends, No room to connect the 3rd device to the subnet
  - So, maximum no of devices could be present in subnet with mask 24 – theoretically = 254, but practically = 2 in this case
  - So, why use /24 as a mask, use mask value which strictly allows two devices in the subnet
  - Such a mask value is = 30. Thus, point to point links, mask value used is usually 30

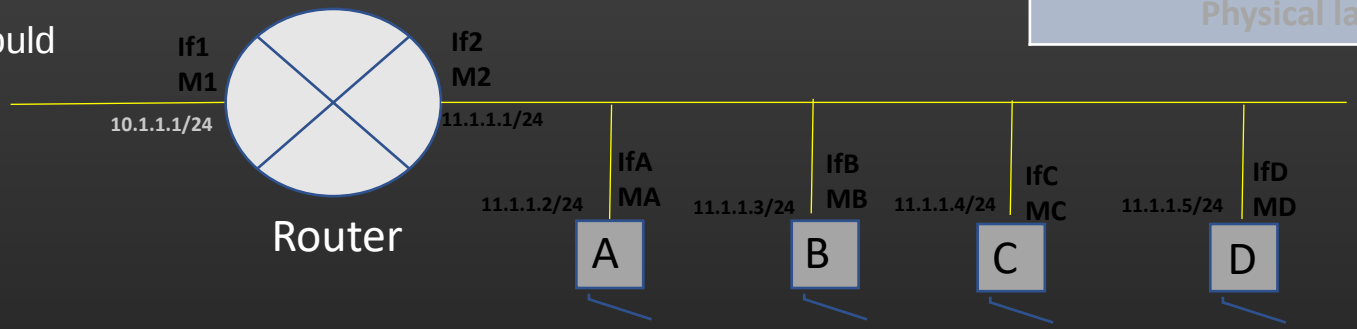
  - With mask = 30, possible IP are :
    11.1.1.1/30, 11.1.1.2/30 – only two, hence only two interfaces can be present in a subnet
    11.1.1.0/30 – Network ID
    11.1.1.3/30 – subnetwork broadcast address

Lecture VDO 3

Subnetting & IP address Maths -> Broadcast Address

Application layer
Transport Layer
Network Layer
Data link layer
Physical layer

- Broadcast Address
  - All receivers in the domain should receive the packet through broadcast address



- Two types of Broadcast Addresses
  - Mac Layer Broadcast Address : ff:ff:ff:ff:ff:ff
  - Network layer Broadcast Address : Already discussed how to calculate

- So, if a Machine A in the subnet wants to send an IP packet such that all nodes in the subnet receives and process it, Node A should specify:
  - Mac Layer Broadcast Address : ff:ff:ff:ff:ff:ff
  - Network layer Broadcast Address : 11.1.1.255

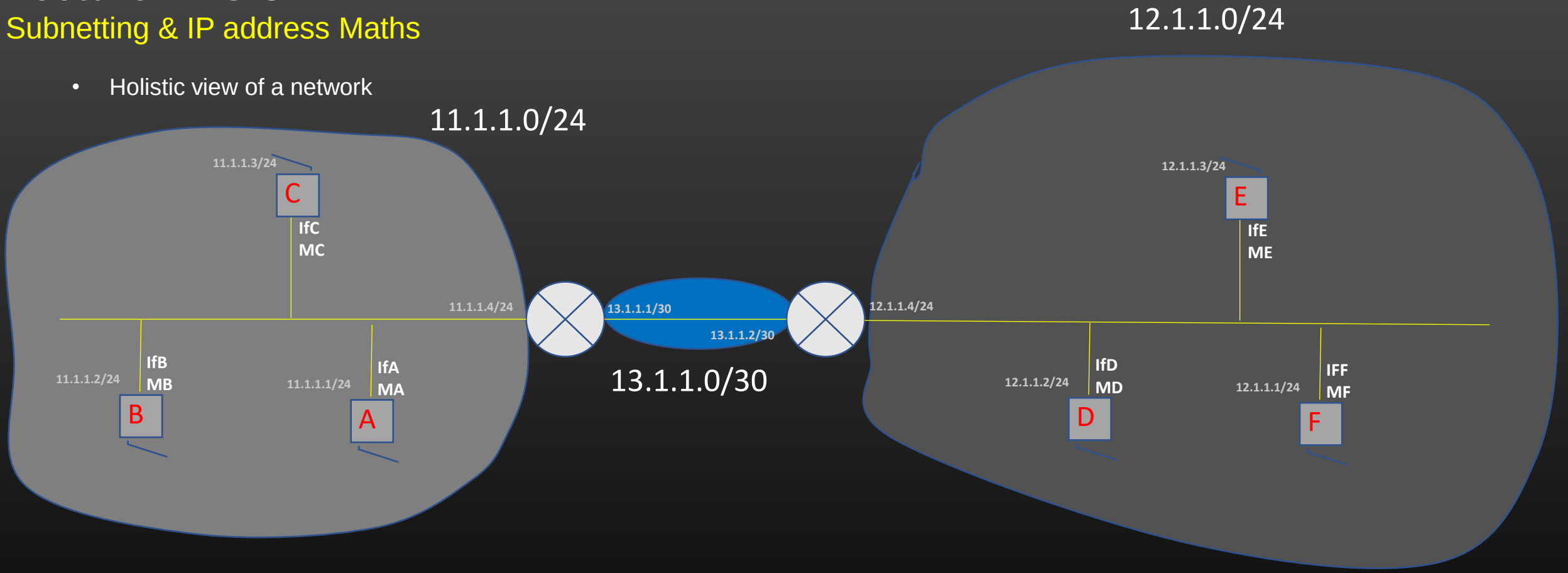
Mac Layer Rule
If Dest_Mac == Mac add of receiving interface Or
Dest Mac is Broadcast Address
Then Accept
Else
Reject

IP Layer Rule
If Dest_IP == IP add of receiving interface Or
Dest_IP is Broadcast Address
Then Accept
Else
Reject

Lecture VDO 3

Subnetting & IP address Maths

- Holistic view of a network



- 3 Subnets, Subnets are separated by L3 routers in between
- Subnets 11.1.1.0/24 and 12.1.1.0/24 can have maximum 254 machines
- No other machine can be added to subnet 13.1.1.0/30
- If a packet with destination IP address = 11.1.1.255, and mac address : ff:ff:ff:ff:ff:ff arrives in subnet 11.1.1.0/24, it will be accepted by all machines (broadcast)
- Next section – L2 routing

Lecture VDO 3

Subnetting & IP address Maths -> Coding Assignment

- Coding Assignment based on what you have studied in this module

- Prerequisites :
  - Bit-Wise AND , OR Operation in C
  - Left and Right shift Operations
  - Thorough understanding of this module

- Upload Assignments
  - Upload all your codes on your github account

- Relevance
  - Prepare you how to use Bit wise manipulation to achieve Networking related Operations
  - Not knowing Bit-Wise operations in C is a great put off for interviewers.

Lecture VDO 3

Subnetting & IP address Maths -> Coding Assignment

- Write a C functions as below in file *ip_maths.c*

Q1. void
get_broadcast_address(char *ip_addr, char mask, char output_buffer);

Usage :

```
char ipadd_buffer[PREFIX_LEN];   // PREFIX_LEN is 16 (constant)
memset(ipadd_buffer, 0, PREFIX_LEN);
char *ip_add = “192.168.2.10”;
char mask = 20;
get_broadcast_address(ip_add, mask, ipadd_buffer);
printf(“Broadcast address = %s\n”, ipadd_buffer);
```

Test case1 :

Input : 192.168.2.10
Mask : 24
Output : 192.168.2.255

Test case2 :

Input : 10.1.23.10
Mask : 20
Output : 10.1.31.255

Use link2 for result verification

Q2. unsigned int
get_ip_integeral_equivalent(char *ip_address);

Usage:

```
char *ip_address = “192.168.0.10”;
unsigned int int_ip = get_ip_integeral_equivalent(ip_address);
printf(“Integer equivalent for %s is %u\n”, ip_address, int_ip);
```

Test case1 :

Input : 192.168.2.10
Output : 3232236042

Test case2 :

Input : 10.1.23.10
Output : 167843594

Use link1 for result verification

Use the below links to verify your result
Link1 : http://www.silisoftware.com/tools/ipconverter.php
Link2 : http://jodies.de/ipcalc

Lecture VDO 3

Subnetting & IP address Maths -> Coding Assignment

- Write a C functions as below :

Q3. void
get_abcd_ip_format(unsigned int ip_address, char *output_buffer);

Usage:

```
unsigned int int_ip = 2058138165; /* = 122.172.178.53*/
char ipadd_buffer[PREFIX_LEN];
memset(ipadd_buffer, 0, PREFIX_LEN);
get_abcd_ip_format(int_ip, ipadd_buffer);
printf("Ip in A.B.C.D format is = %s\n", ipadd_buffer);
```

Use link1 for result verification

Q4. void
get_network_id(char *ip_addr, char mask, char* output_buffer);

Usage :

```
char ipadd_buffer[PREFIX_LEN];   // PREFIX_LEN is 16 (constant)
memset(ipadd_buffer, 0, PREFIX_LEN);
char *ip_add = "192.168.2.10";
char mask = 20;
get_network_id(ip_add, mask, ipadd_buffer);
printf("Network Id = %s\n", ipadd_buffer);
```

Use link2 for result verification

Lecture VDO 3

Subnetting & IP address Maths -> Coding Assignment

- Write a C functions as below :

Q5. unsigned int
get_subnet_cardinality(char mask);

Usage:

```
unsigned char mask = 24;
printf("Subnet cardinality for Mask = %u is %u\n", mask, get_subnet_cardinality(mask));
```

Use link2 for result verification

Q6. int /*Return 0 if true , -1 if false*/
check_ip_subnet_membership(char *network_id, char mask, char *check_ip);

Usage :

```
char *network_id = "192.168.0.0";
char mask = 24;
char *check_ip = "192.168.0.13";
int result = check_ip_subnet_membsership(network_id, mask, check_ip);
if(result == 0)
        printf("Ip address = %s is a member of subnet %s/%u\n", check_ip, network_id, mask);
else
        printf("Ip address = %s is a not a member of subnet %s/%u\n", check_ip, network_id, mask);
```

Lecture VDO 4

Routing – L2 Routing

L2 Routing

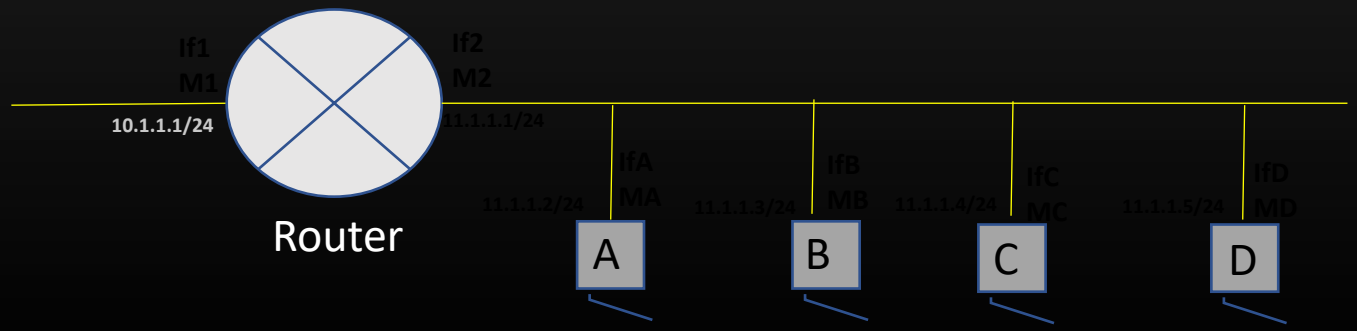
Lecture VDO 4

Routing – L2 Routing

- We will discuss L2 Routing - Routing done at Data Link Layer
- Only Mac addresses is used to deliver the packet to destination machine
- Once the packet has received from outside world to L3 router (Later), thereafter packet is delivered to destination machine in a subnet using mac addresses only
- L2 routing means – Routing within a subnet
  - From one machine to another machine within a subnet

    OR

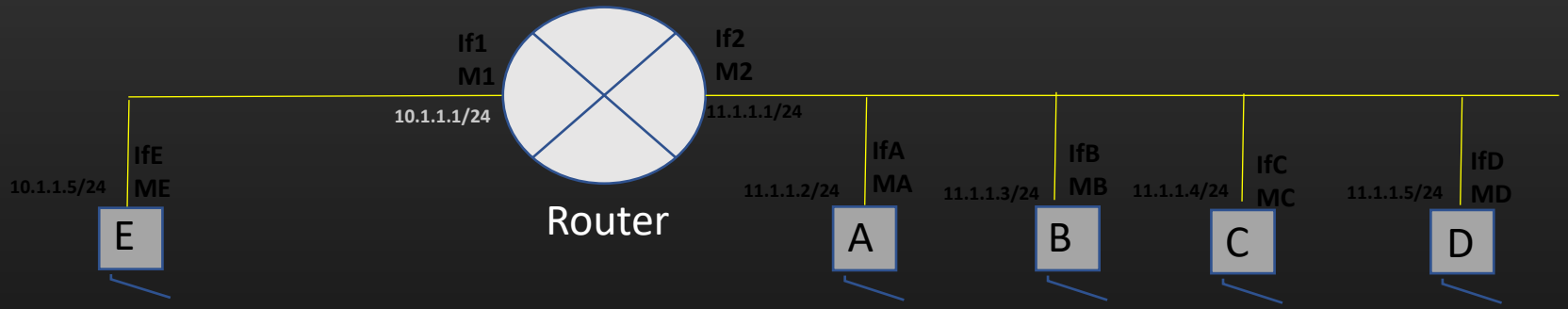
  - From L3 Router to machine present in a directly connected subnet



Lecture VDO 4

Routing – L2 Routing -> General Basics

- Router is a gateway to a subnet. Gateway means – common point of entry and exit to and from the subnet
- Router's each local interface hosts a subnet



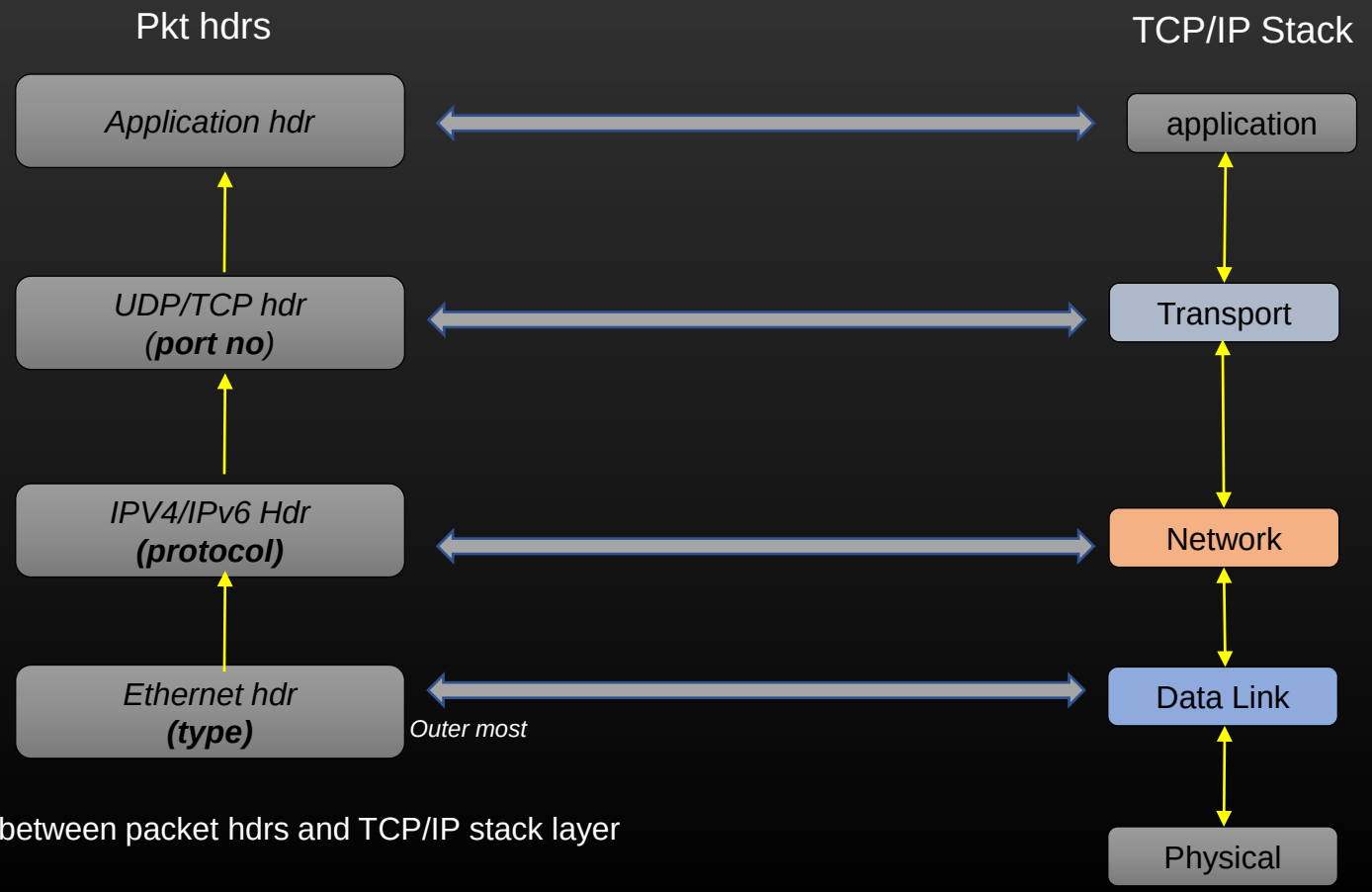
- A machine present in one subnet cannot talk to any machine present in other subnet without L3 router
- L3 router are the boundaries of subnet

Lecture VDO 4

Routing – L2 Routing -> General Basics

- Protocol identifier field

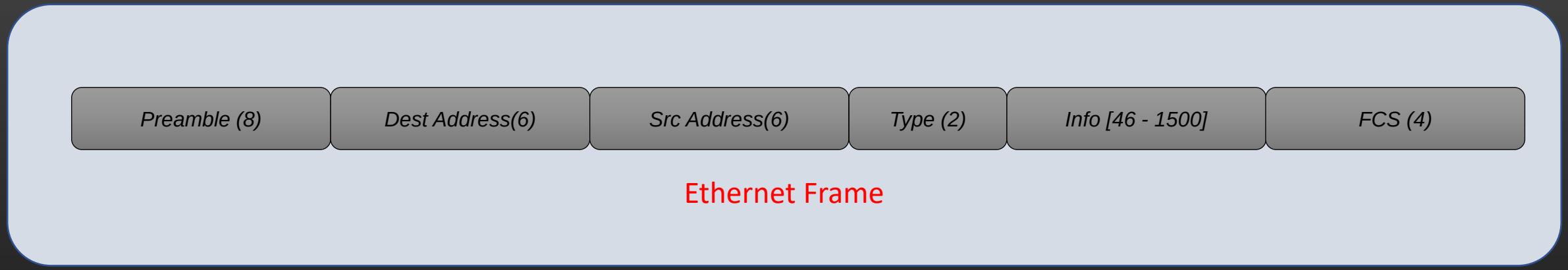
Every header in the packet has to provide the mechanism to convey what is the next hdr following the current header in the packet



One to one correspondence between packet hdrs and TCP/IP stack layer

Lecture VDO 4

Routing – L2 Routing -> Ethernet Header format



Preamble – 8 bytes, mark the start of the new frame.
Dest Addr – 6 bytes Destination Mac address

Source Address - 6 bytes Source Mac address

Type – Identifier what next layer Protocol is
Example, Type = 0800 if next hdr is IP hdr, 0041 for IPv6 hdr
Type = 0806 if next hdr is ARP hdr , etc

Info – Payload data [46 - 1500]
FCS - checksum

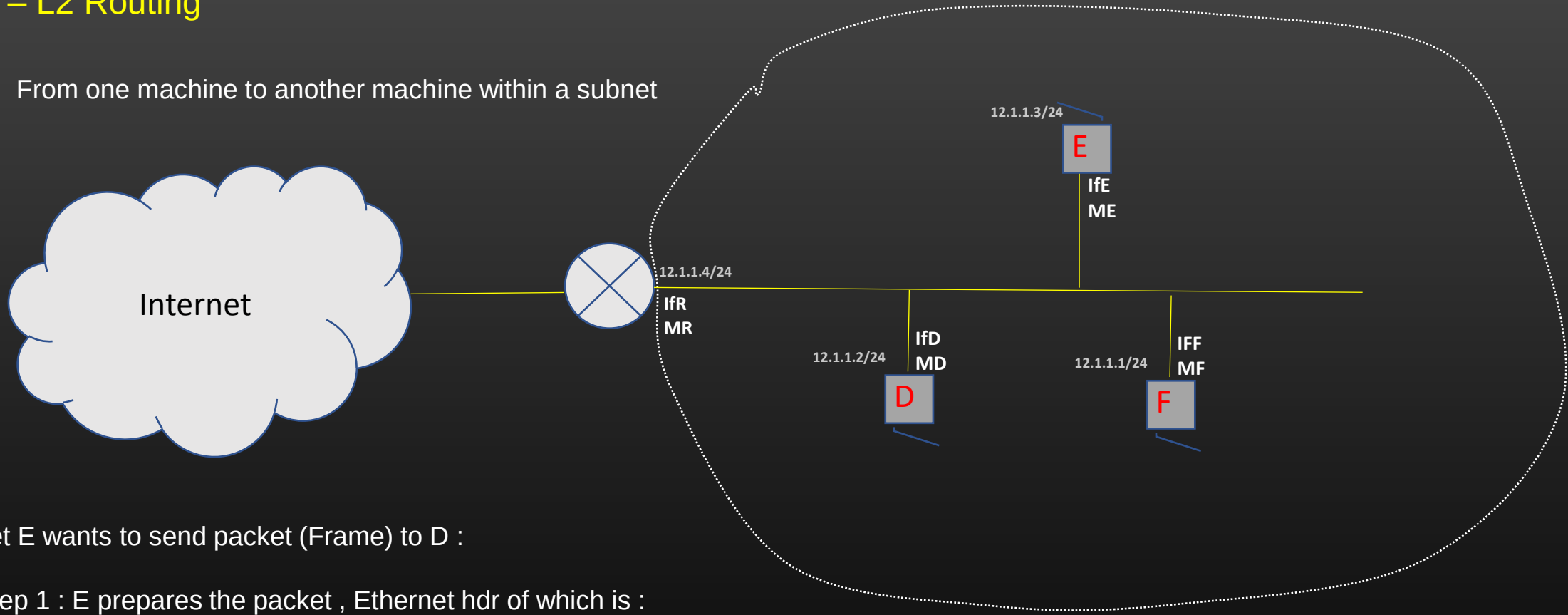
Lecture VDO 4

Routing – L2 Routing -> Ethernet Header format -> MTU size

Lecture VDO 4

Routing – L2 Routing

- From one machine to another machine within a subnet



Let E wants to send packet (Frame) to D :

Step 1 : E prepares the packet , Ethernet hdr of which is :
src mac = ME, dst mac = MD, Type = 0800(IP)

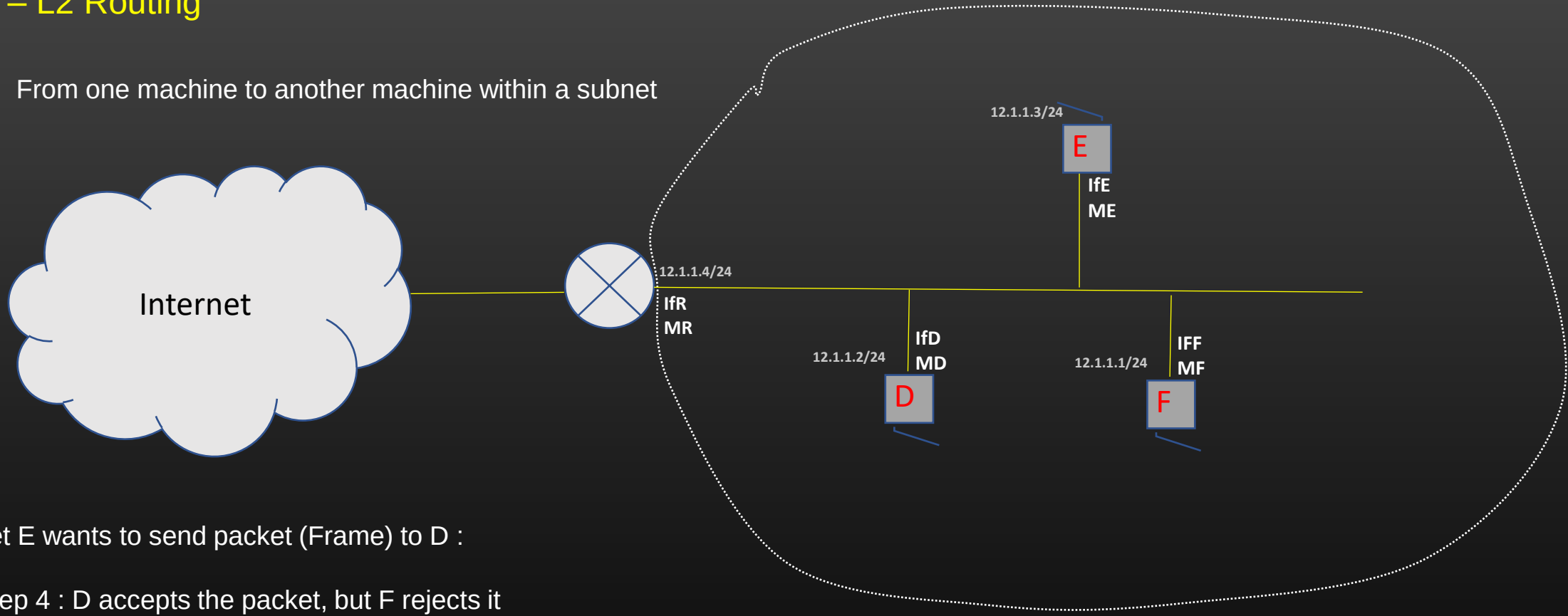
Step 2 : E transmits the packet on wire, the packet is received by all other remaining machines connected to wire

Step 3 : When other machines receives the packet, they check *if Dst mac = Mac of receiving interface, accept the packet*, chop off the Ethernet hdr, and handover the packet to Network Layer. If dst mac != Mac of receiving interface, then receiving machine discards the pkt

Lecture VDO 4

Routing – L2 Routing

- From one machine to another machine within a subnet



Let E wants to send packet (Frame) to D :

Step 4 : D accepts the packet, but F rejects it

Step 5 : Router also receives the packet, but reject it since dst Mac != MR

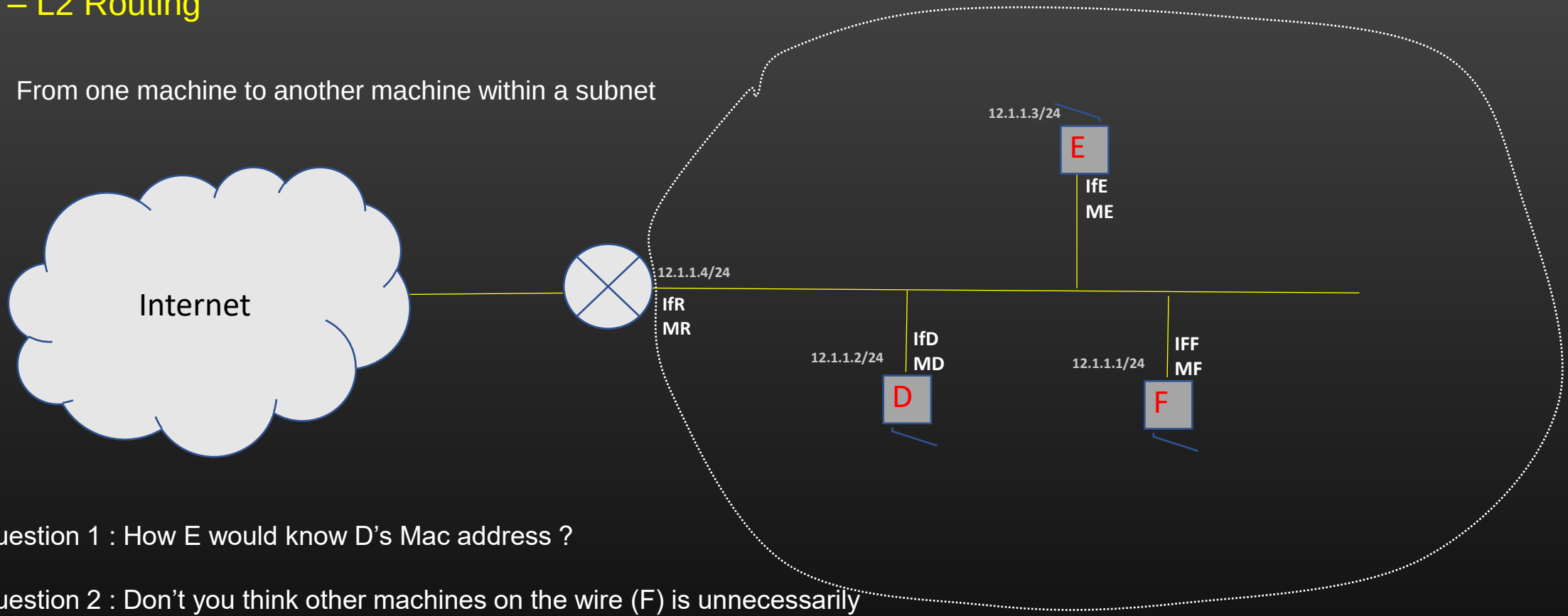
This is how L2 routing is done – Based only on mac addresses.

But don’t you have some Questions ???

Lecture VDO 4

Routing – L2 Routing

- From one machine to another machine within a subnet



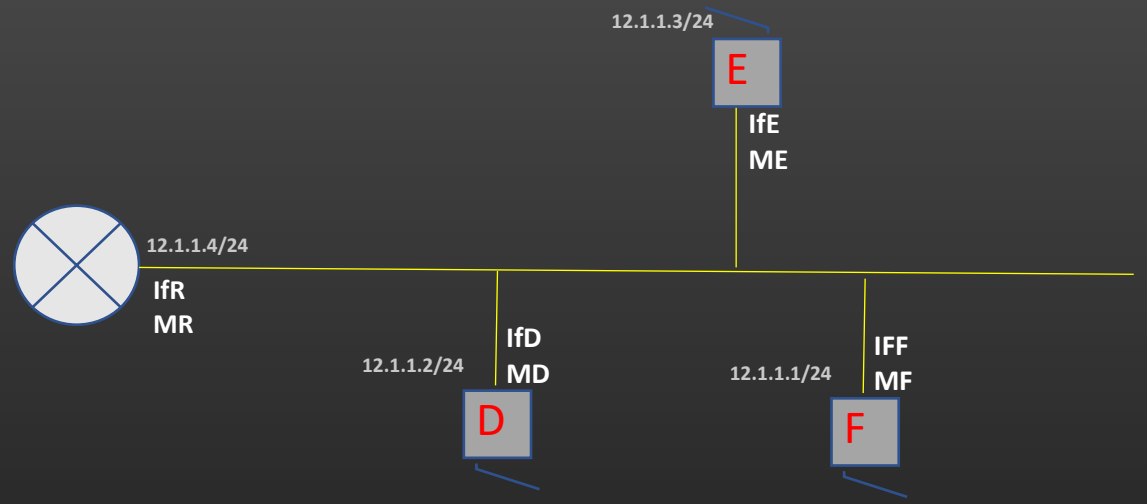
Question 1 : How E would know D's Mac address ?

Question 2 : Don't you think other machines on the wire (F) is unnecessarily disturbed for the packet not destined to it ? (Thrashing)

Answer 1 : ARP (Address Resolution Protocol)
Answer 2 : L2 switch

Lecture VDO 4

Routing – L2 Routing -> ARP Protocol



- ARP GOAL :

  Sender Must know the receiver's MAC address, given that sender knows the Recipient IP address already

  For example :

  Sender : D
  Receiver : E

  D must know E's MAC address which is ME to send a L2 frame to E successfully

  ARP allows D to know E's MAC address so that D can talk to E

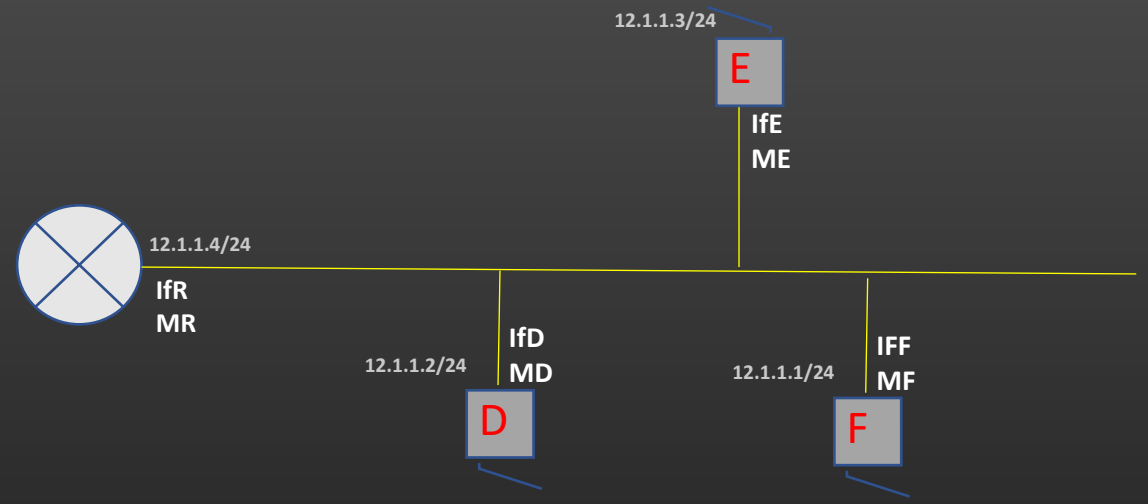
  Similarly, D should know F and L3 router's MAC which is MF and MR respectively in order to exchange frames with them

  Using ARP, D maintains IP-to-MAC Mapping of all other machines it is communication with in a table called ARP table

  Thus, ARP helps Sender to populate its ARP table

Lecture VDO 4

Routing – L2 Routing -> ARP Protocol



- L3 routers and devices (hosts) maintains a table called ARP table
- ARP protocol is used to populate this table
- ARP table contains the mapping of IP to MAC of direct nbrs
- Example : ARP table of machine E and D would be :

IPs	MAC	oif
12.1.1.1	MF	ifE
12.1.1.2	MD	ifE
12.1.1.4	MR	ifE

IPs	MAC	oif
12.1.1.1	MF	ifD
12.1.1.3	ME	ifD
12.1.1.4	MR	ifD

- Now if E wants to send packet to D (12.1.1.2), it checks its ARP table and pick MD as dst mac to be placed in Ethernet hdr out of interface ifE

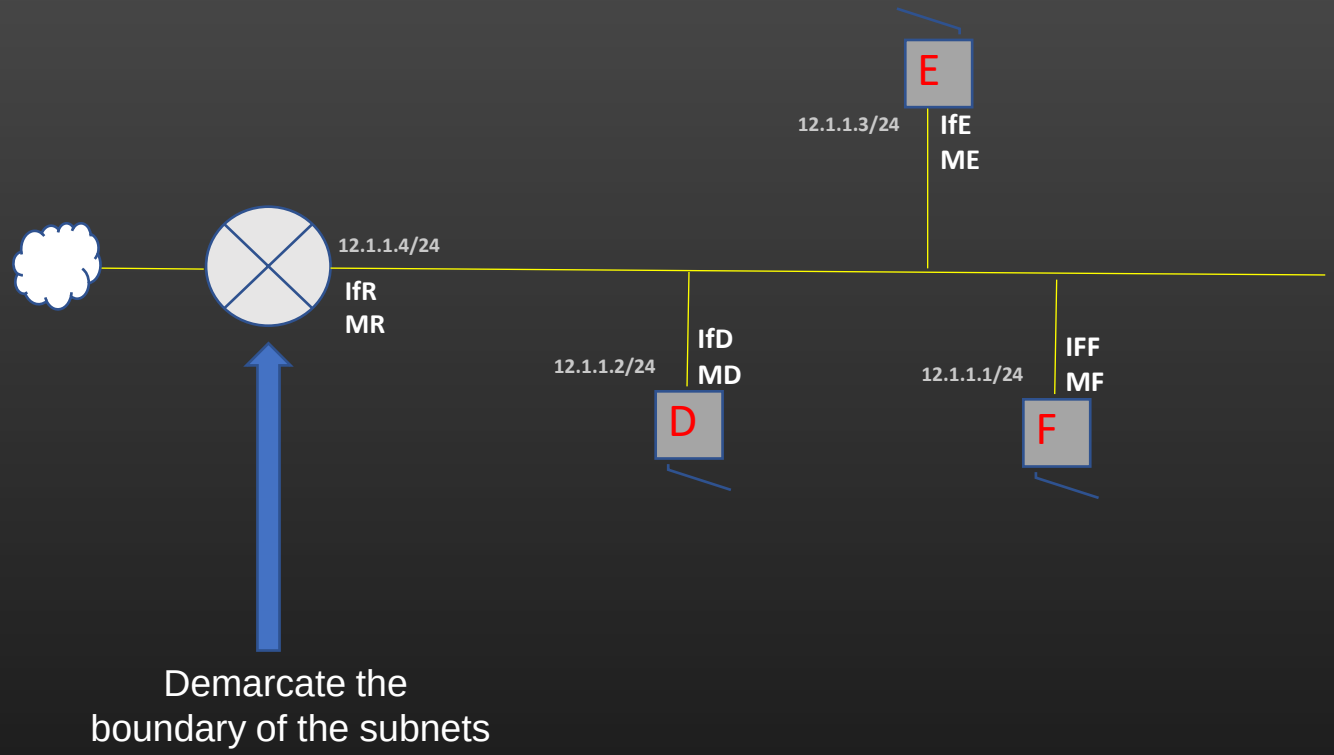
Lecture VDO 4

Routing – L2 Routing -> ARP Protocol

- ARP works using two types of messages :
  - ARP request (broadcast msg)
  - ARP reply (unicast msg)

- ARP messages never crosses the subnet boundary

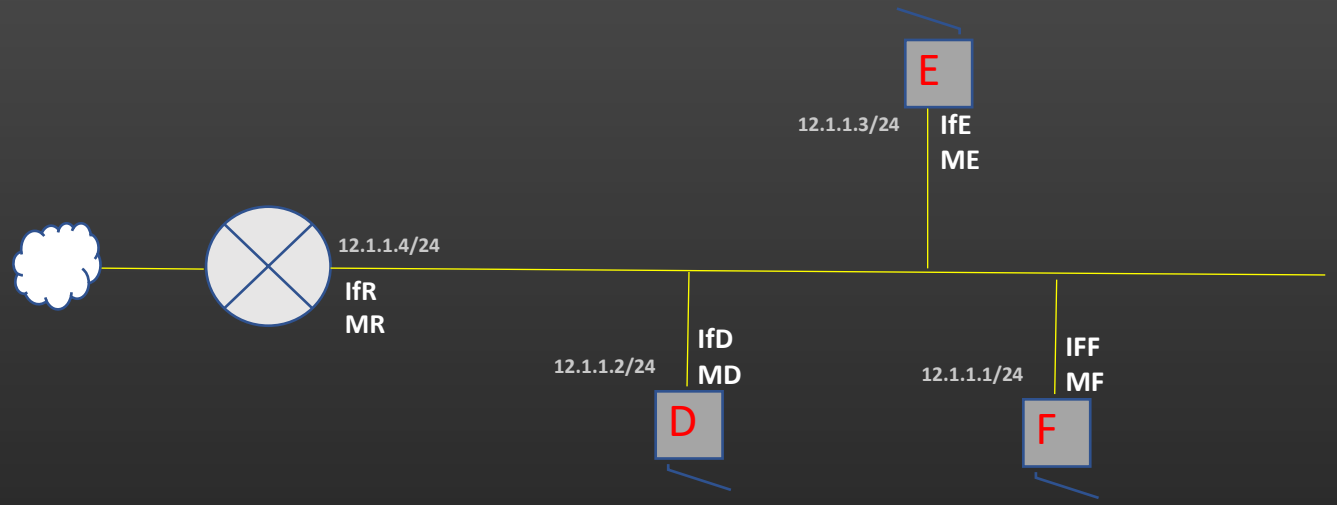
- All machines in same subnets are said to be neighbors



Lecture VDO 4

Routing – L2 Routing -> ARP Protocol

- ARP Header format specification:



Hardware type : 1 for ethernet cable
18 for optical fibre

Protocol type : Is ARP being resolved for IPV4
Or Ipv6 address ?
0x0800 for IPV4

Hw Address Length : 6 if Hw address is MAC

Protocol Address length : 4 for IPV4, 16 for IPV6

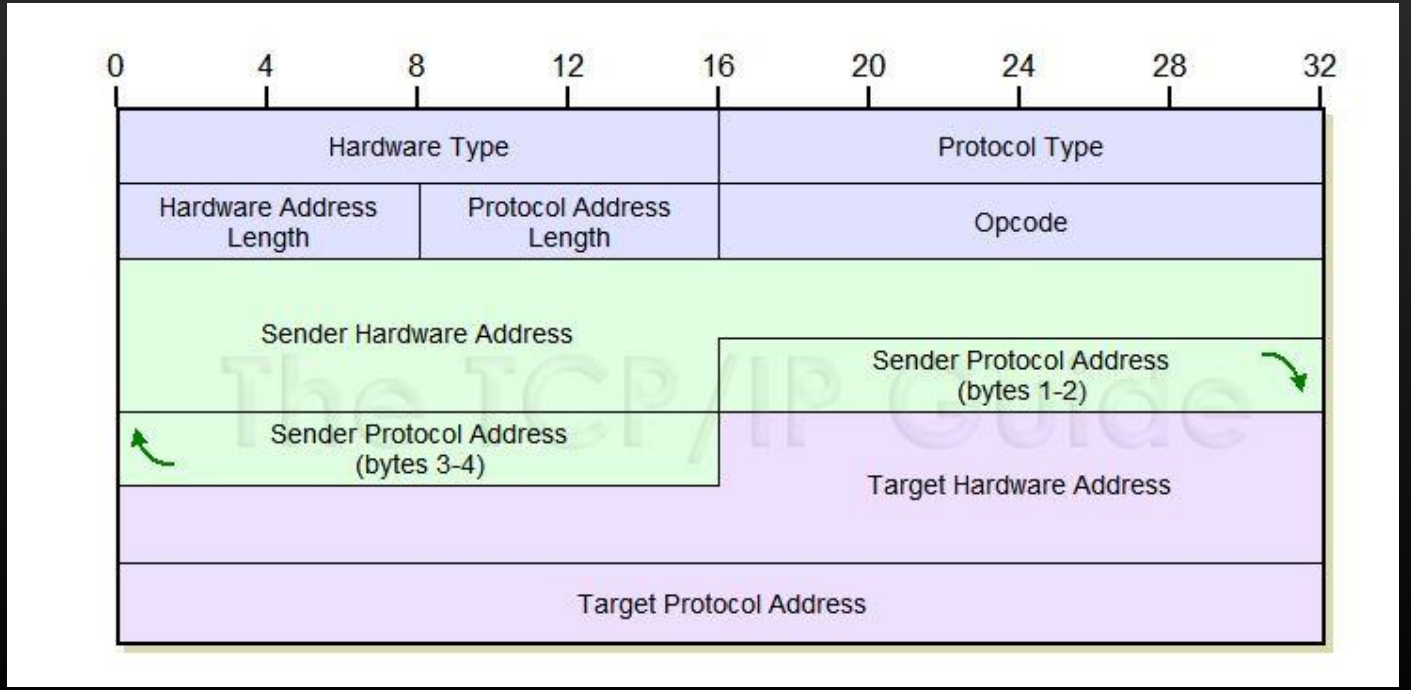
Opcode : 1 for ARP req, 2 for ARP reply

Sender Hw addr : MAC address of sender

Sender Protocol addr : IP address of sender

Target Hw Addr : Recipient MAC Address

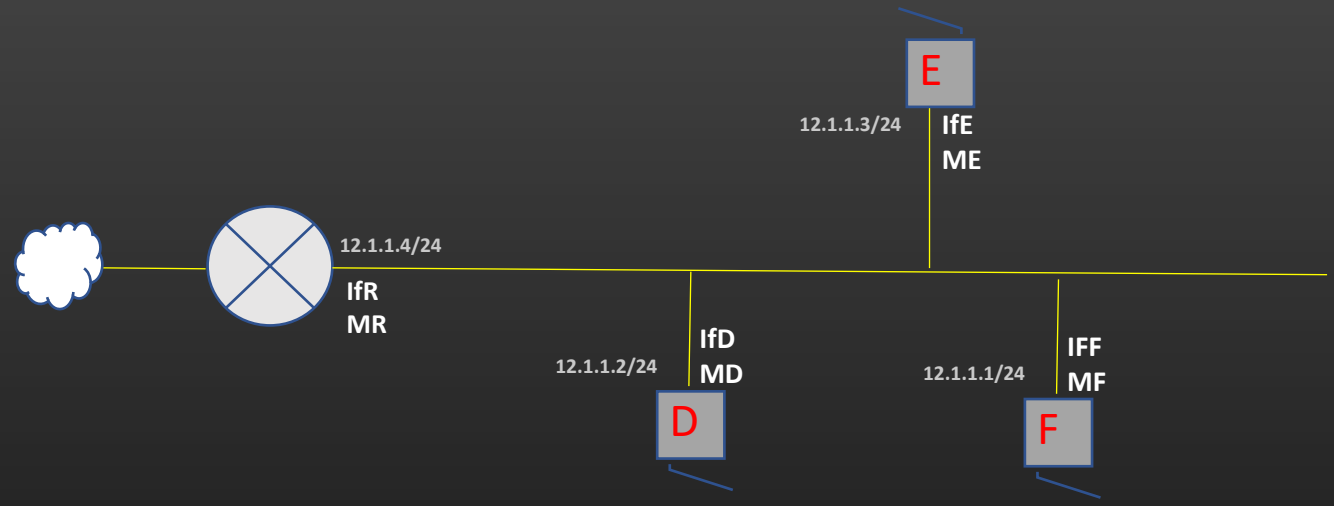
Target Protocol Addr : Recipient IP addr



Lecture VDO 4

Routing – L2 Routing -> ARP Protocol

- ARP works using two types of messages :
  - ARP request (broadcast msg)
  - ARP reply (unicast msg)



Steps regarding How E would know D's Mac Address ?

Step 1 : E wants to send packet to D (12.1.1.2)

Step 2 : E prepares the Ethernet hdr to send packet
dst mac = ?

Step 3 : Now E look up its ARP table and check the entry for IP = 12.1.1.2. E don't find *MD* opposite to 12.1.1.2 entry in its ARP table

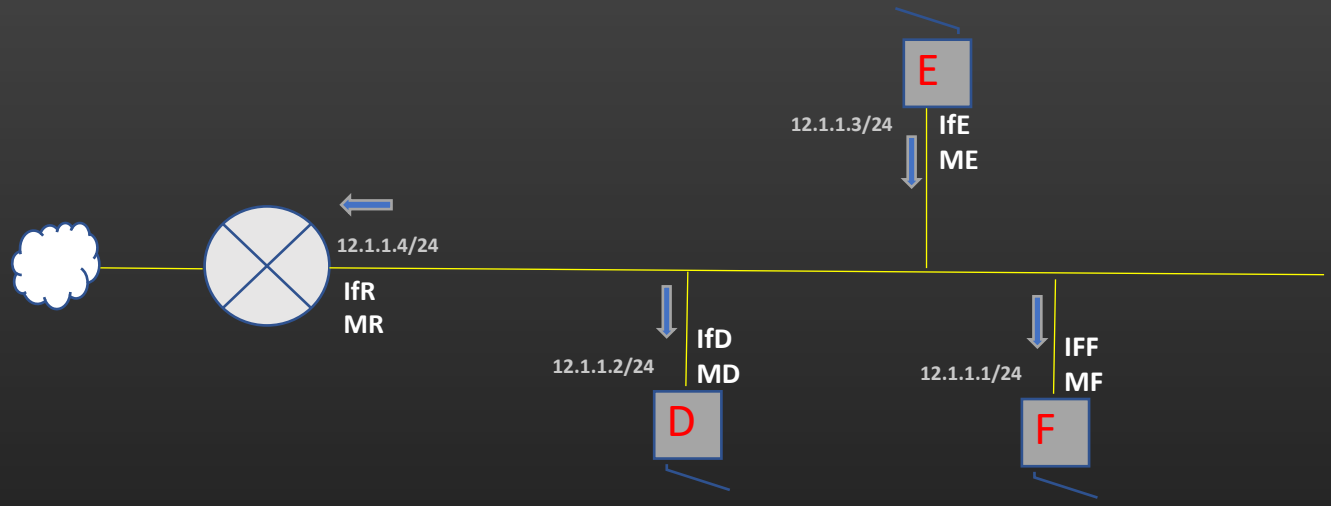
Step 4 : E now checks whether D is within its own subnet or in some distant remote subnet
E computes its own Network ID (12.1.1.0/24)
E apply its own MASK (=24) to D's IP address (12.1.1.2) , it get 12.1.1.0/24 which is same as E's network ID
E concludes D is in its own subnet (What if D is in remote subnet – Later on this)

Step 5 : E now wants to know D's Mac address. It sends out ARP request msg on all local interfaces operating in network 12.1.1.0/24

Step 6 : E places the packet on wire

Lecture VDO 4

Routing – L2 Routing -> ARP Protocol



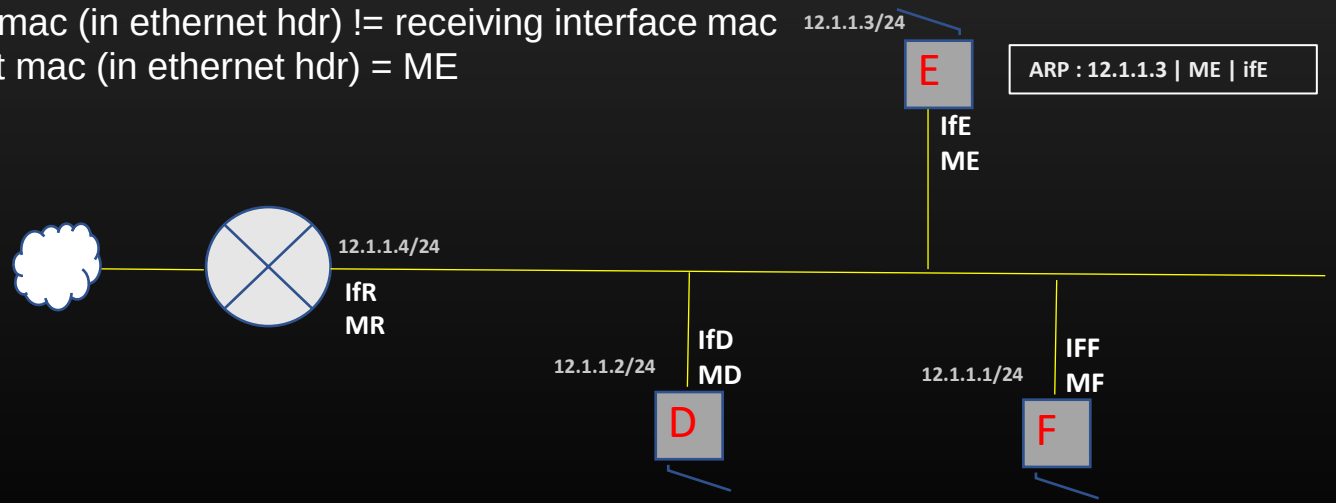
ARP Broadcast Request Message

Lecture VDO 4

Routing – L2 Routing -> ARP Protocol

How E would know D's Mac Address ?

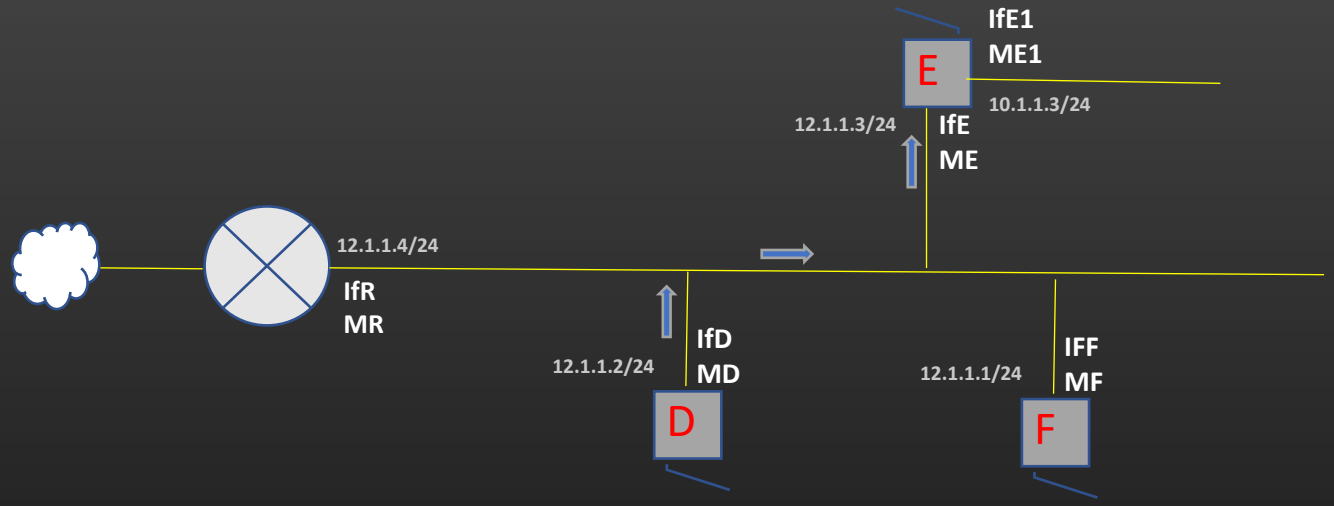
Step 7 : ARP request packet is received by all interfaces connected to the wire
Step 8 : Data link layer of none of the machine discards the ARP req packet because dst mac = broadcast address
Step 9 : All receiving machines chops off the Ethernet hdr and handover the packet to ARP protocol
Step 10 : Data link layer handover the frame to next layer in TCP/IP stack - ARP protocol
Step 11 : ARP protocol, running on top of Data link layer sees dst ip in the ARP packet
Step 11 : if dest ip = ip address of receiving interface, machine sends back ARP reply, else discards the packet
Step 12 : ARP reply contents
src mac = MD, dst mac = ME, src ip = 12.1.1.2, dst ip = 12.1.1.3, TYPE = ARP REPLY
Step 13 : ARP reply is placed on wire by machine D and is received by all interfaces connected to wire
Step 14 : All receiving machines discards the packet if dst mac (in ethernet hdr) != receiving interface mac
Step 15 : ARP replay msg is processed by E only since dst mac (in ethernet hdr) = ME

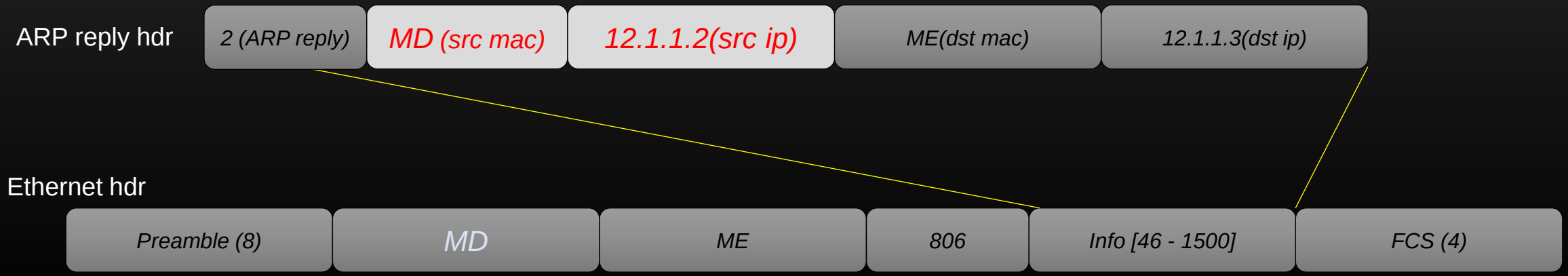


Lecture VDO 4

Routing – L2 Routing -> ARP Protocol

- ARP works using two types of messages :
  - ARP request (broadcast msg)
  - ARP reply (unicast msg)
- ARP message never crosses the subnet boundary





Lecture VDO 4

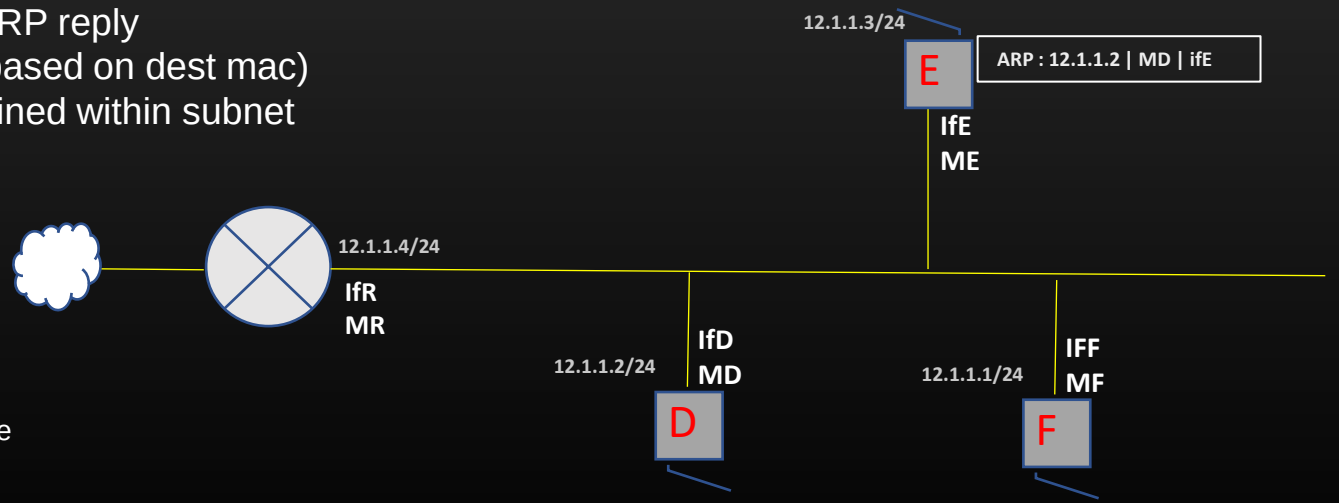
Routing – L2 Routing -> ARP Protocol

How E would know D's Mac Address ?

Step 16 : E updates its ARP table with the following entry :
12.1.1.2 – MD, and oif = ifE (Remember ARP table is always updated using Src Mac and Src ip fields in ARP req/reply msgs)
Step 17 : E now knows what it was looking for – D's Mac address
Step 18 : E now inserts D's mac in ethernet hdr it was preparing and place it on wire when it is ready
Step 19 : Pkt is received by all interfaces connected to wire, all discards but D (Thrashing)
Step 20 : Data from E has been successfully transmitted to D

Points to Note :

- Enquiring machine (E) populates its ARP table using ARP reply
- ARP req msg is broadcast, where as reply is unicast (based on dest mac)
- ARP msgs never crosses L3 router boundary and confined within subnet
- ARP works on Layer 2 only.
- No of ARP entries = No of devices connected to wire



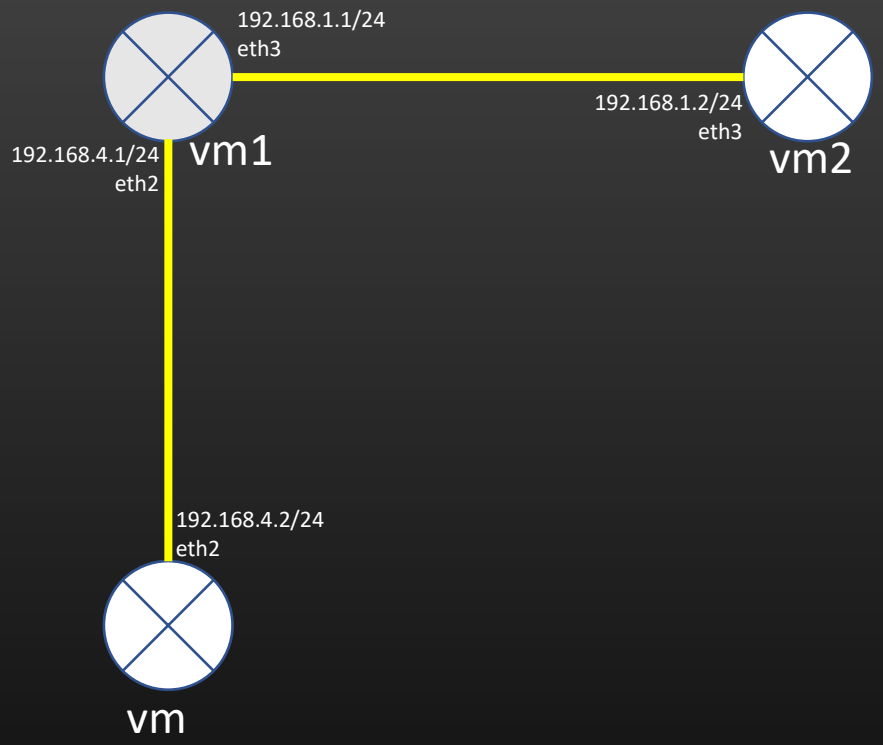
You are advised to read more about ARP online, to know
Exact message format. We have given clear Idea of how ARP works !

Remembering Exact msg format is not required as long as you can explain the
Protocol functioning and main fields in msgs

Lecture VDO 4

Routing – L2 Routing -> ARP Protocol

ARP In Action !!

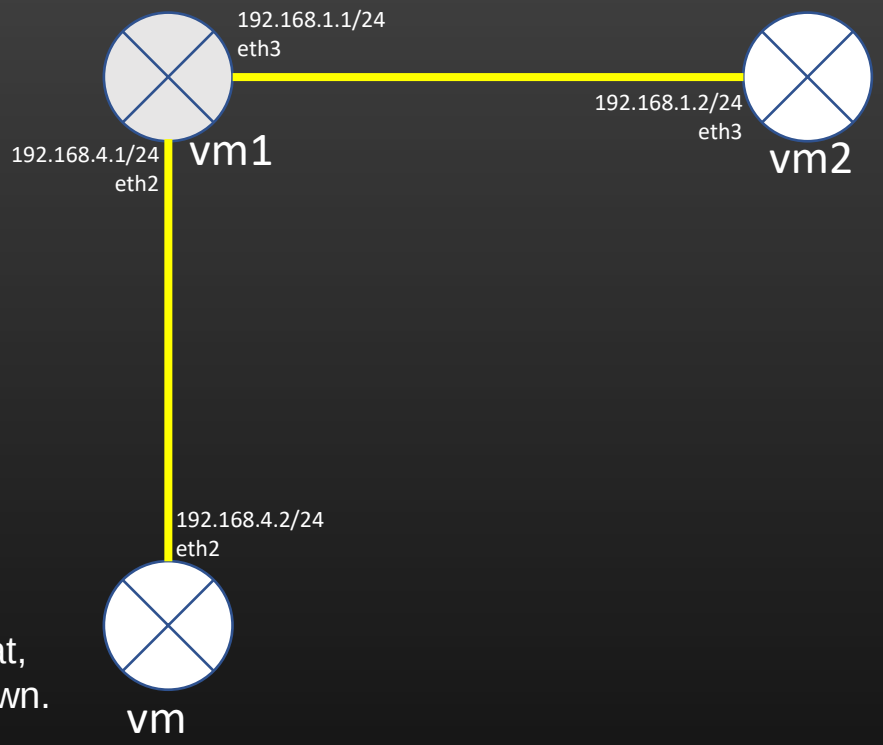


- Remember ARP operates between machines connected to same subnet.
- In this topology, ARP will operate between machines Vm and Vm1 because they have one interfaces each in the Same subnet (192.168.4.0/24)
- In this topology, ARP will also operate between machines Vm1 and Vm2 because they have one interfaces each in the Same subnet (192.168.1.0/24)
- In this topology, ARP will NOT operate between machines Vm and Vm2 because they don't have any interfaces operating in the Same subnet
- ARP do not operates between machines present in different subnets – in this example, between Vm and Vm2

Lecture VDO 4

Routing – L2 Routing -> ARP Protocol

ARP In Action !!



- Start all 3 VMs. See the ARP table in all of these machines using arp –n

- Initially, you will not see ARP entry for the direct nbr machine. For example, arp -n command on VM will NOT should you any IP-MAC mapping entry for IP = 192.168.4.1 (The other machine in same subnet as VM)

- Now ***ping 192.168.4.1*** from VM. Check again arp -n on VM and VM1. You will See IP-MAC mapping on both machines.

Steps :

1. ping command on vm asks vm to send hello (ICMP msg) to 192.168.4.1. For that, Vm need to know src mac, dst mac, src ip, dst ip. Among these dst mac is not known.

2. Vm launches ARP broadcast request out on interface which is same subnet as destination (192.168.4.1)

3. VM1 replies with ARP reply msg. VM on receiving ARP reply populates its ARP table with src mac and src ip specified in ARP reply msg.

4. VM , can now send packet to vm1.

Lecture VDO 4

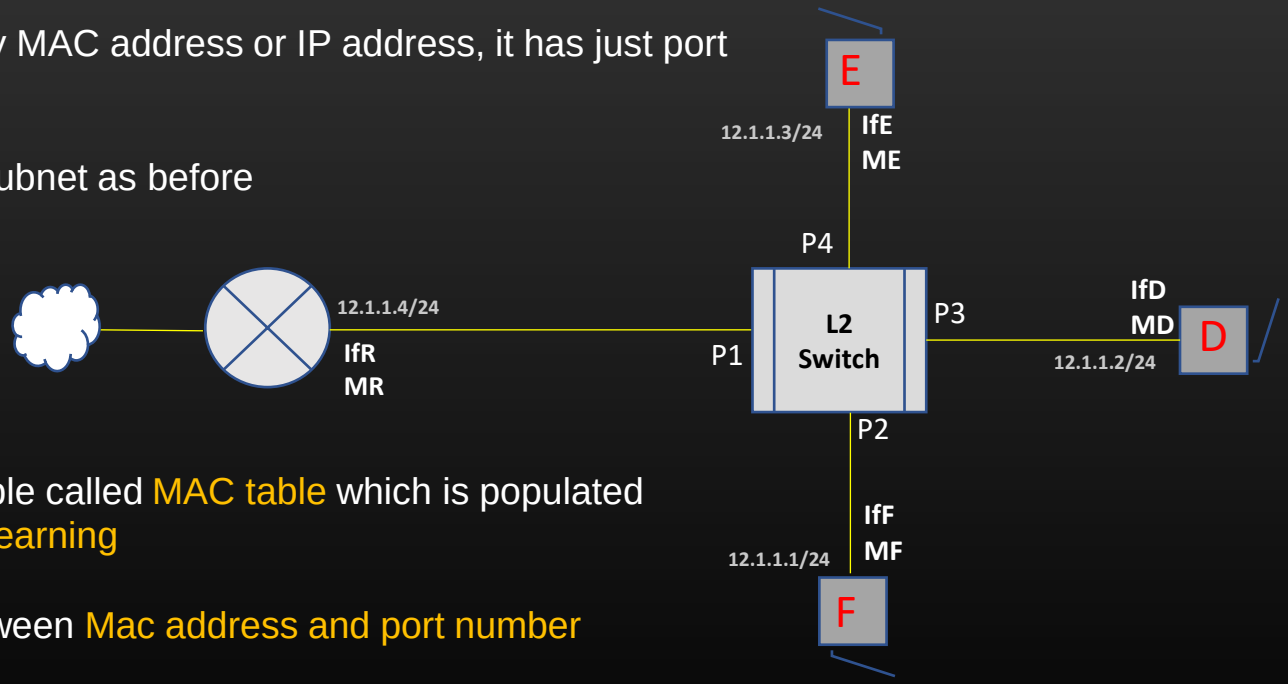
Routing – L2 Routing -> L2 Switch

L2 Switch

Lecture VDO 4

Routing – L2 Routing -> L2 Switch

- Remember the problem of Thrashing of rest of the machines in the subnet When two machines communicate with each other
- Let us deploy L2 switch in the network and understand how it solves thrashing.
- L2 switch is also called a bridge and is a layer 2 device.
- L2 switch interface do not have any MAC address or IP address, it has just port Numbers – P1 . . . P4
- E,D and F machines are in same subnet as before

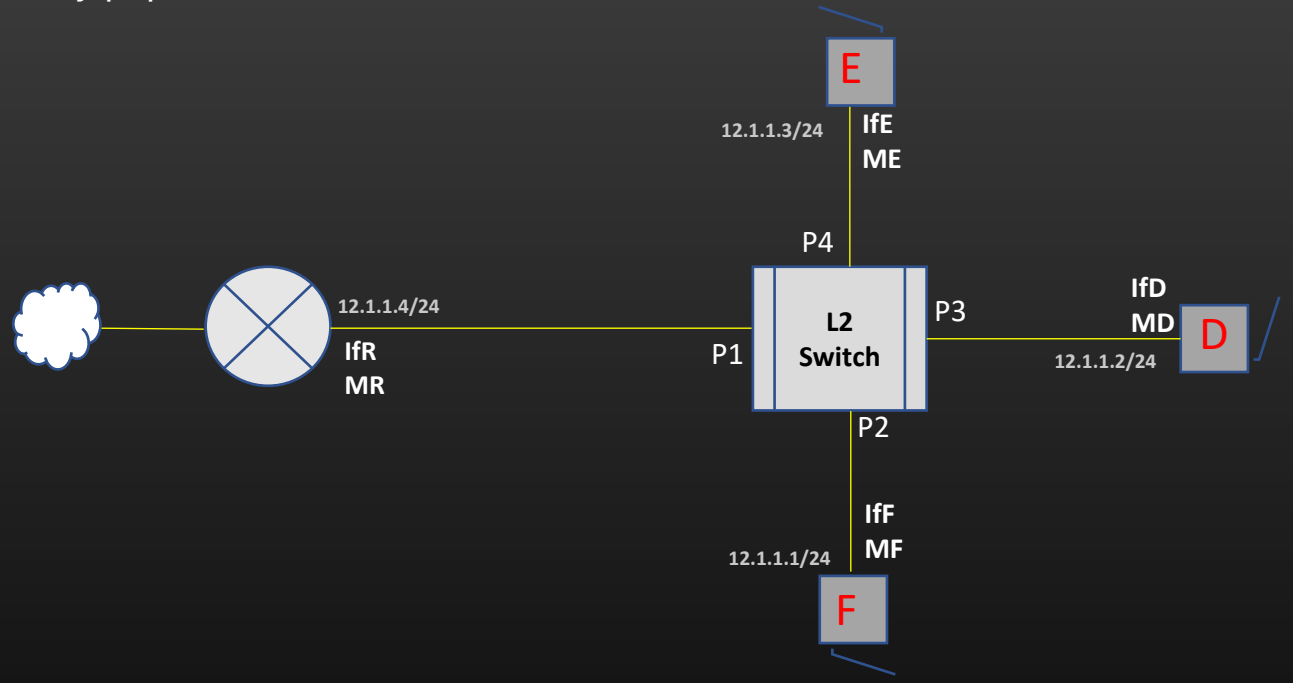


- L2 switch Or bridge maintains a table called MAC table which is populated using a procedure called – Mac learning
- Mac table contains the mapping between Mac address and port number

Lecture VDO 4

Routing – L2 Routing -> L2 Switch -> Mac table

- Fully populated L2 Switch mac table



Mac address	Outgoing port no
ME	P4
MR	P1
MF	P2
MD	P3

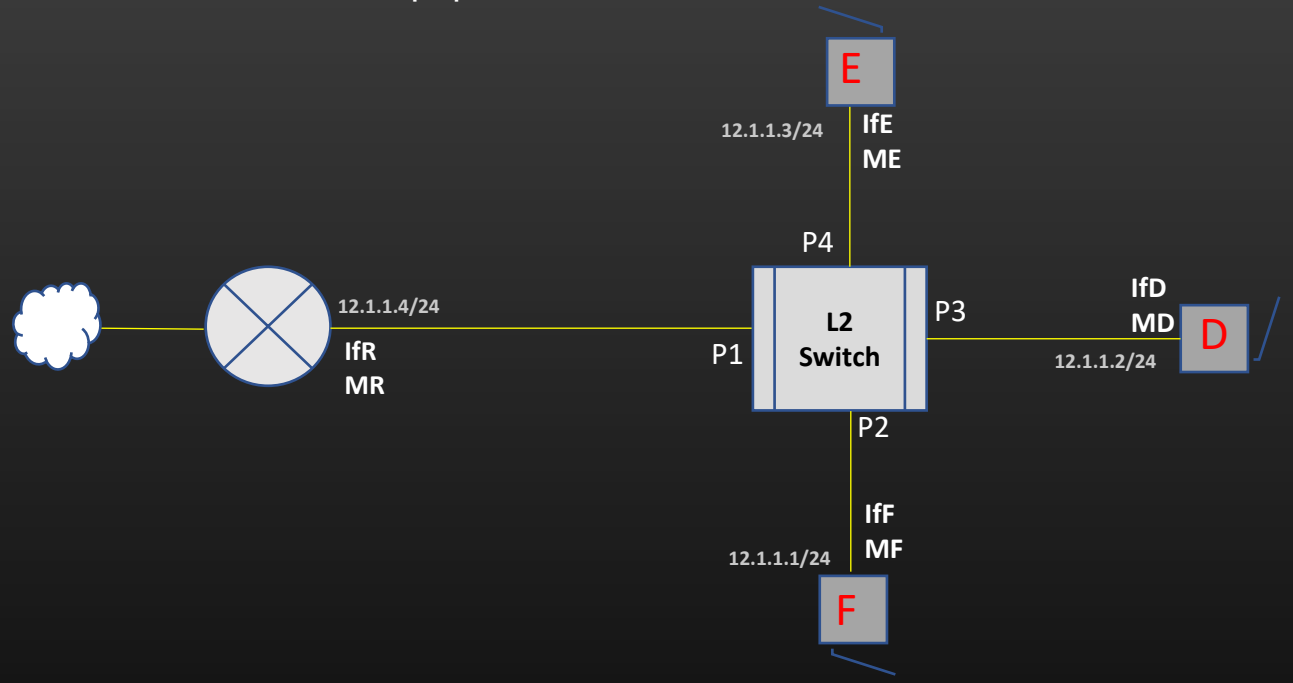
- From the table, if L2 switch receives packet with dest mac = ME from any interface other than P4, it will look up its mac table with ME as key and finds, the packet should get out of port P4.

- Thus, Only E would receive the packet and no other machine in the subnet is un-necessarily thrashed with packet not destined to it.
- I am sorry – When we talk about routing in Data link layer , we should call packet a FRAME !!!!

Lecture VDO 4

Routing – L2 Routing -> L2 Switch -> Mac Learning

- Lets see how L2 switch populate its mac table



Mac address	Outgoing port no

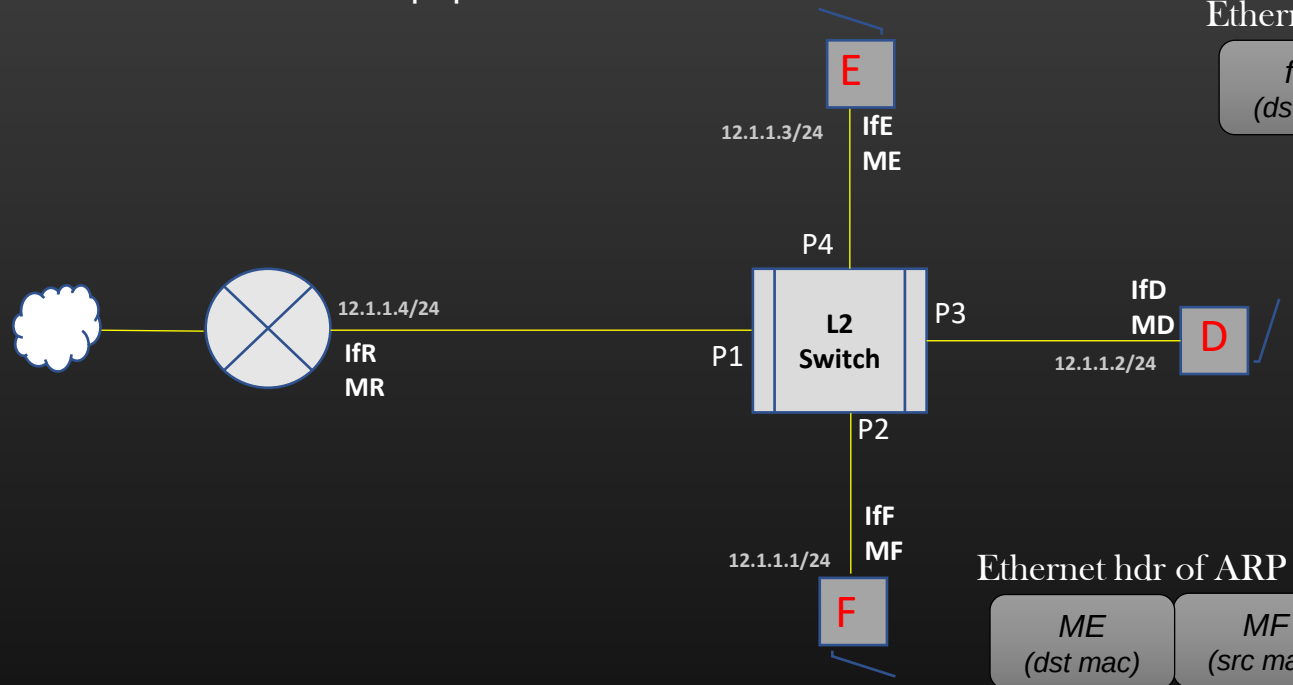
- L2 Switch (Or bridge) make an entry in the mac table by inspecting the contents of Ethernet hdr of the frame
- Whenever L2 switch do not have entry and do not know which port to forward the packet to, it floods the packet out of all ports

- L2 switch populate the Mac table by inspecting the Src Mac address of ethernet frame it receives
- L2 switch forwards the packet to correct port no by inspecting the destination mac address of ethernet frame it receives

Lecture VDO 4

Routing – L2 Routing -> L2 Switch -> Mac Learning

- Lets see how L2 switch populate its mac table



Ethernet hdr of ARP B Msg

ffffff (dst mac)	*ME* (src mac)

Mac address	Outgoing port no
ME	P4

Ethernet hdr of ARP Reply Msg

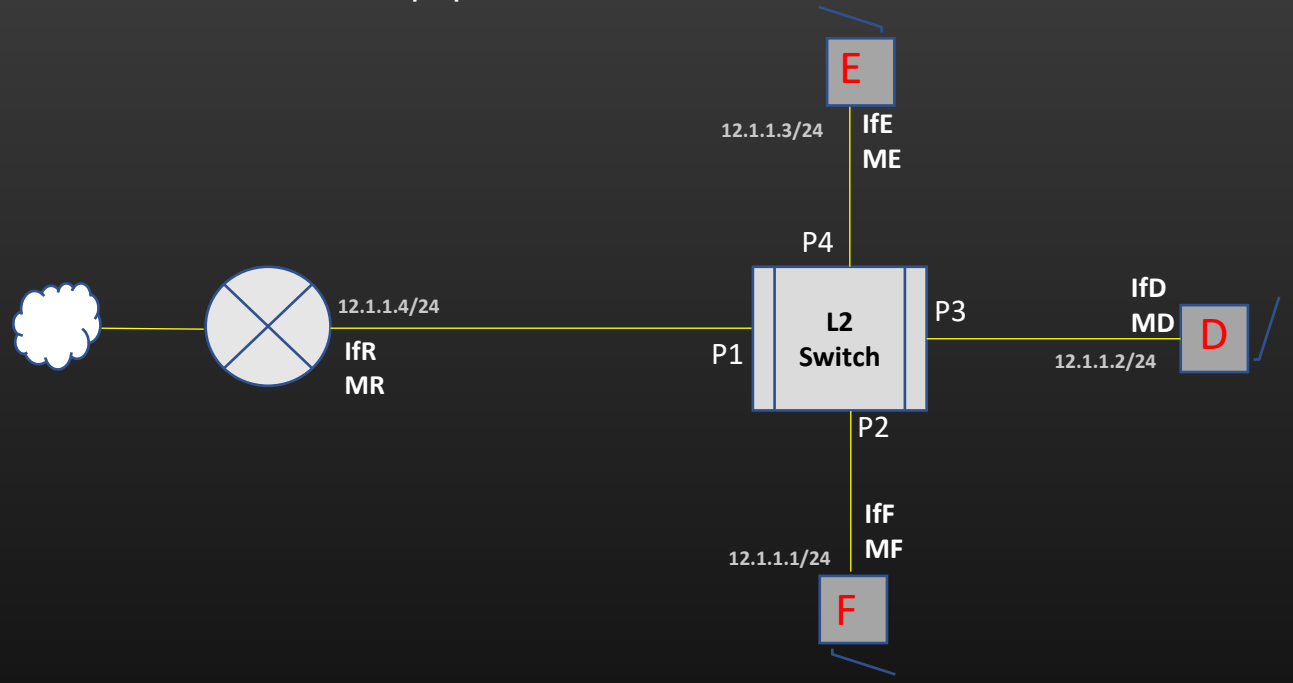
ME (dst mac)	*MF* (src mac)

Lets Us say , E wants to send Data packet to F
Step 1: E sends ARP broadcast to learn F Mac address as before
Step 2 : L2 switch receives ARP broadcast request on port P4 and reads src mac in the packet
Step 3 : L2 switch concludes E sits on port no P4, it makes an entry in Mac table and flood the ARP req on all ports
Step 4 : Like earlier, only F replies to ARP req msg with ARP reply msg destined to E (dst mac = ME in ARP reply)
Step 5 : L2 switch receives ARP unicast reply msg on port P2 and reads src mac in the packet

Lecture VDO 4

Routing – L2 Routing -> L2 Switch -> Mac Learning

- Lets see how L2 switch populate its mac table



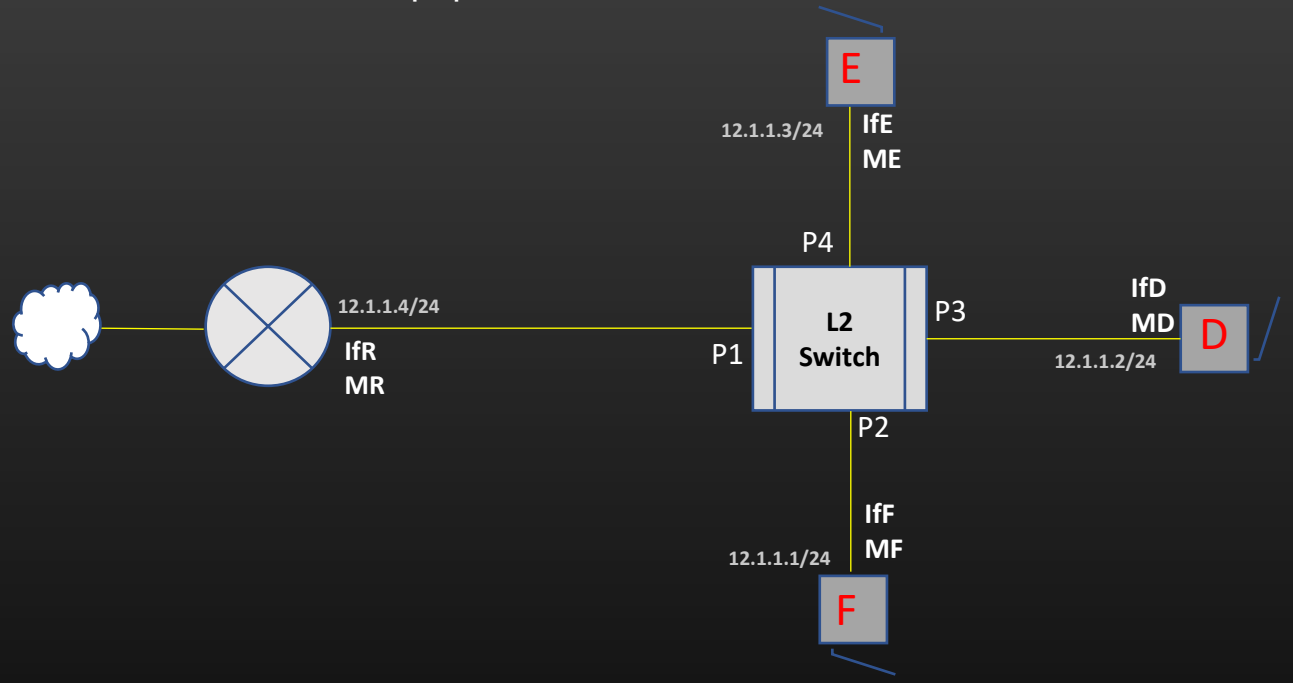
Mac address	Outgoing port no
ME	P4
	P2
MF	

Lets Us say , E wants to send Data packet to F
Step 6: L2 switch concludes F sits on port no P2, it makes an entry in Mac table
Step 7 : L2 switch reads dest Mac in ARP reply msg looks up its Mac table with dest mac as key
Step 8 : Since dst Mac = ME, L2 switch now knows that ARP reply should be forward to port P4
Step 9 : L2 switch forwards ARP reply to port P4
Step 10 : E receives ARP reply and now can prepare the Ethernet frame to send to F

Lecture VDO 4

Routing – L2 Routing -> L2 Switch -> Mac Learning

- Lets see how L2 switch populate its mac table



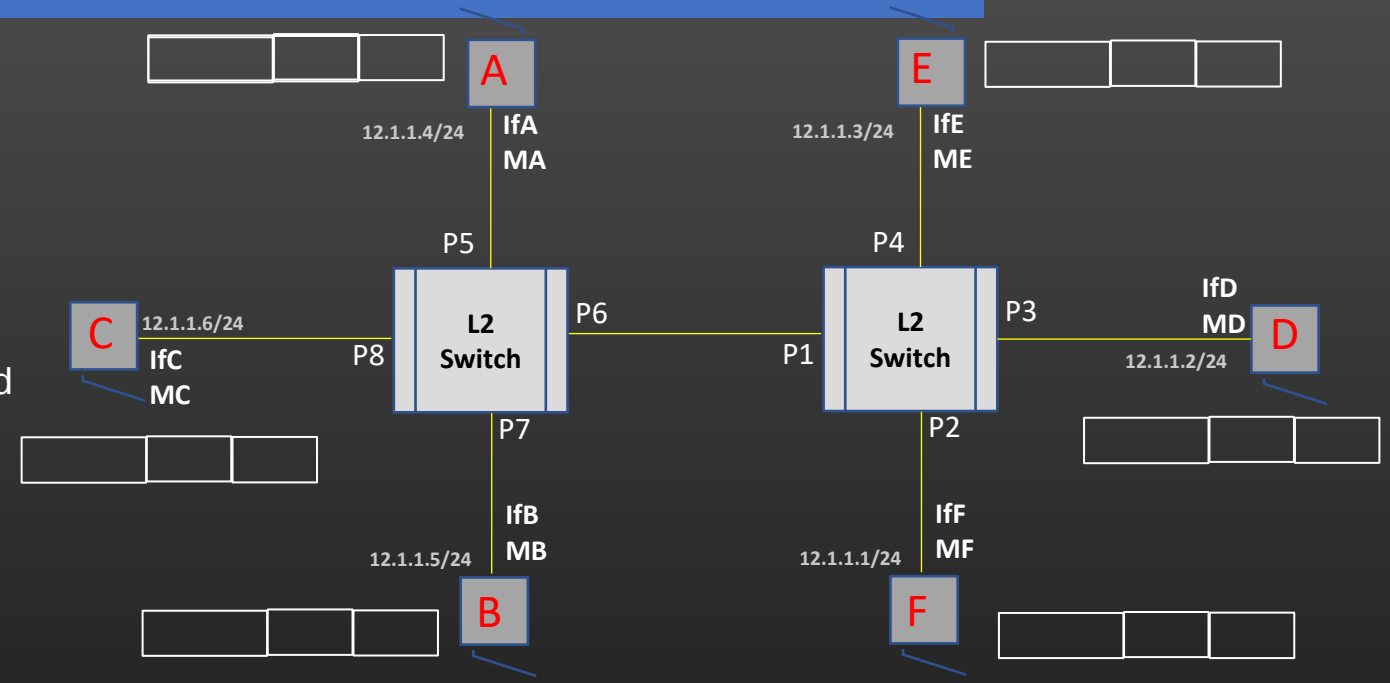
Mac address	Outgoing port no
ME	P4
MR	P1
MF	P2
MD	P3

- Conclusion :
  - L2 switch is a layer two device, and as you can observe it works only with Mac addresses
  - This completes our concept of L2 routing.

Lecture VDO 4

Routing – L2 Routing -> Example

- Lets see end to end example here
- C wants to communicate with D, and subnet is supported by two L2 switches
- Note that, all devices are in same subnet by inspecting their IP addresses – All have same network ID
- Tables used – ARP table for devices, and Mac table for L2 switches, initially all tables are empty



Mac address	Outgoing port no

MAC table

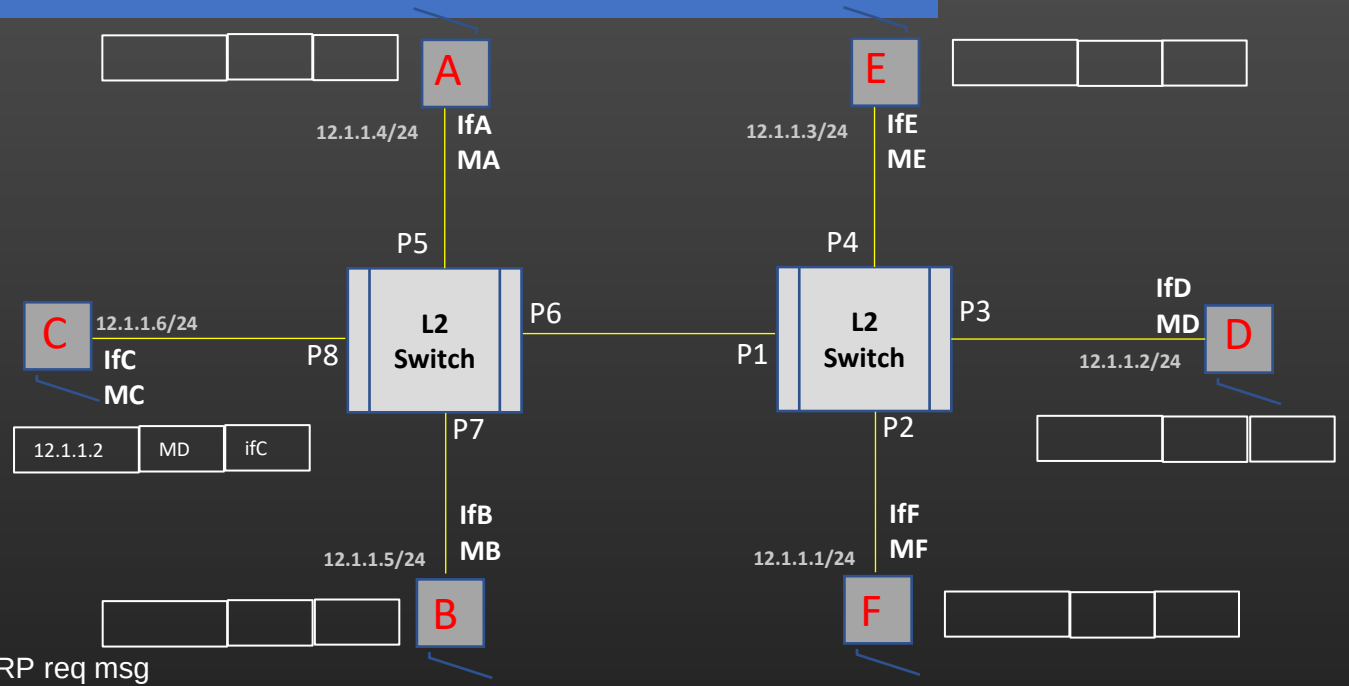
Mac address	Outgoing port no

MAC table

Lecture VDO 4

Routing – L2 Routing -> Example

- C wants to communicate with D, and subnet is supported by two L2 switches



Step 1 : C wants to send Data packet to D whose ip is 12.1.1.2
Step 2 : C verifies that D is in its own local subnet
Step 3 : if step 2 validates, C look up its ARP table with 12.1.1.2 as key
Step 4 : C do not have 12.1.1.2 – MD entry in its ARP table, C launches ARP broadcast req
ARP REQ : src mac = MC, dst MAC = All F's, Mac for ip = 12.1.1.2 ?
Step 5 : L2 switch1 receives ARP req, L2 switch made an entry in its MAC table
Step 6 : L2 switch1 read dst mac, which is broadcast address hence flood the ARP req msg
Step 7 : A,B and L2 Switch2 receives ARP req msg. A,B do not reply
Step 8 : L2 Switch2 receives the ARP req msg and update its mac table. Also floods the pkt on rest of the ports.
Step 9 : E &F receives ARP req pkt but they do not reply.
Step 10 : D replies with ARP reply
ARP REPLY : dst MAC = MC, src mac = MD, Mac for 12.1.1.2 is MD

Step 11 : L2 switch2 receives ARP reply. Update its Mac table.
Step 12 : L2 switch2 also reads dst MAC which is MC, and forwards the packet to port P1 because it has an entry in its MAC table
Step 13 : L2 switch1 receives the packet and does mac learning.
Step 14 : L2switch1 forwards the packet to port P8
Step 15 : C receives the ARP reply and updates its ARP table

Now C can send the data packet to D

Mac address	Outgoing port no
MC	P8
MD	P6

MAC table

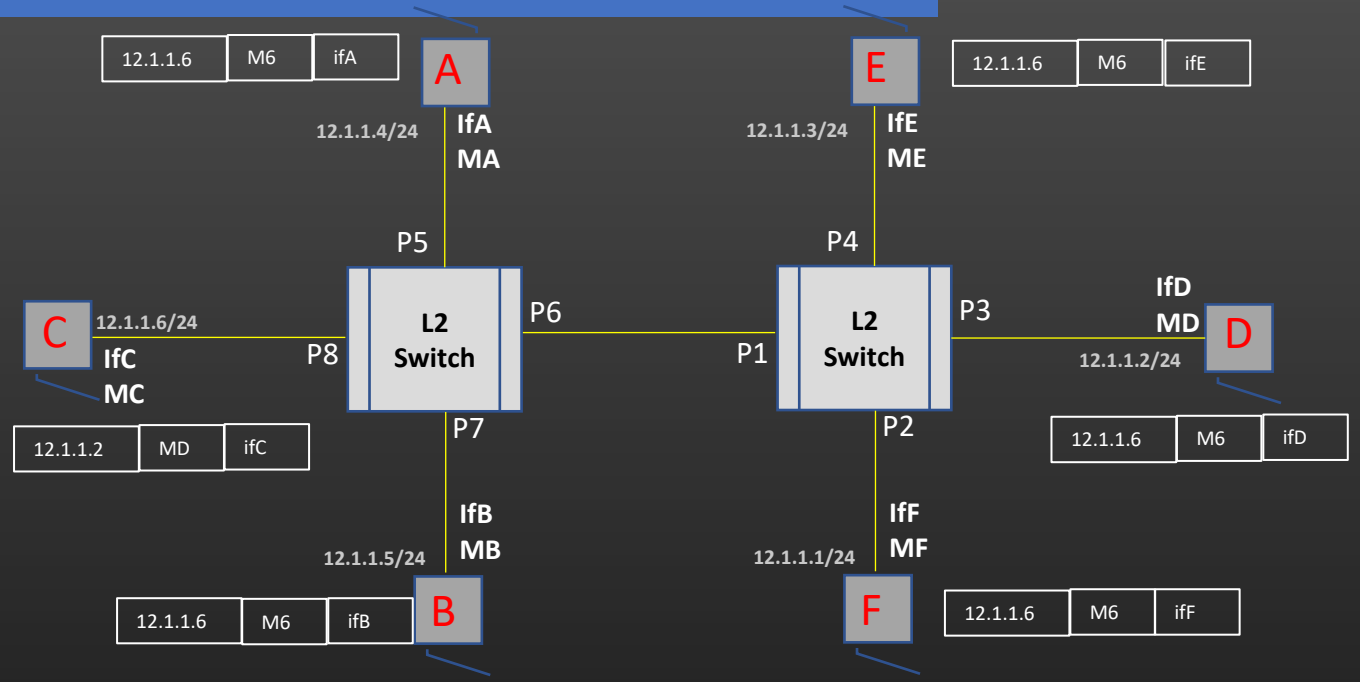
Mac address	Outgoing port no
MC	P1
MD	P3

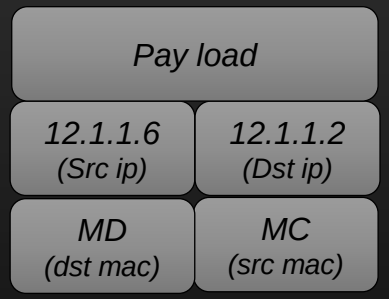
MAC table

Lecture VDO 4

Routing – L2 Routing -> Example

- C wants to communicate with D, and subnet is supported by two L2 switches
- Now, When tables are populated, C can communicate with D





C's packet destined to D

Mac address	Outgoing port no
MC	P8
MD	P6

MAC table

Mac address	Outgoing port no
MC	P1
MD	P3

MAC table

- No Thrashing

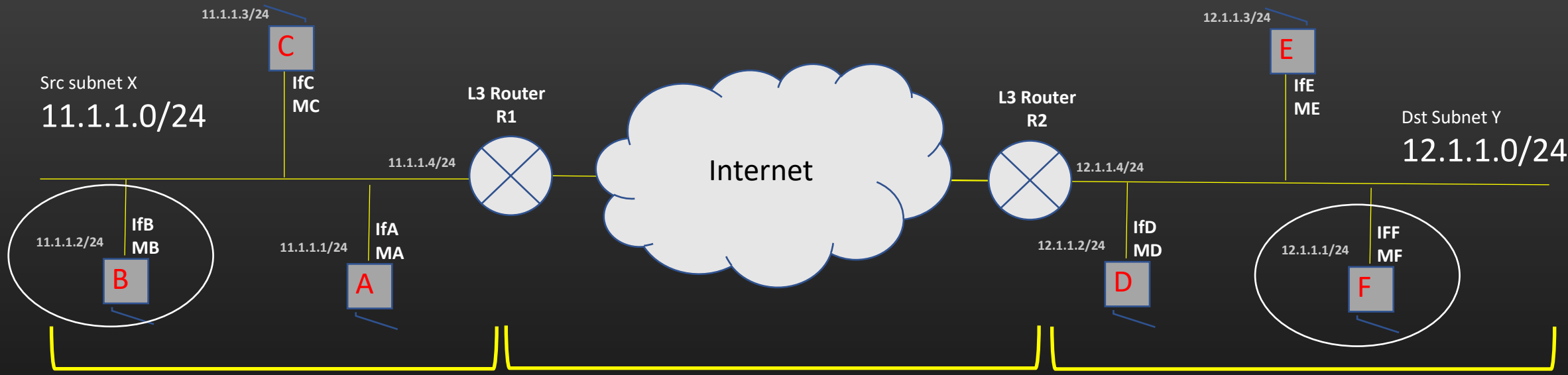
Lecture VDO 5
Routing – L3 Routing

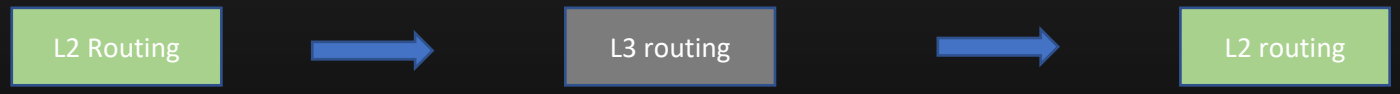
L3 Routing

- We will discuss L3 Routing - Routing done at Network Layer
- Only IP addresses is used to deliver the packet to destination router, forget MAC addresses !
- L3 routing means – Routing from one L3 Router connected to source subnet to L3 router connected to destination subnet

Lecture VDO 5

Routing – L3 Routing





B -- > R1

R1 -- > R2

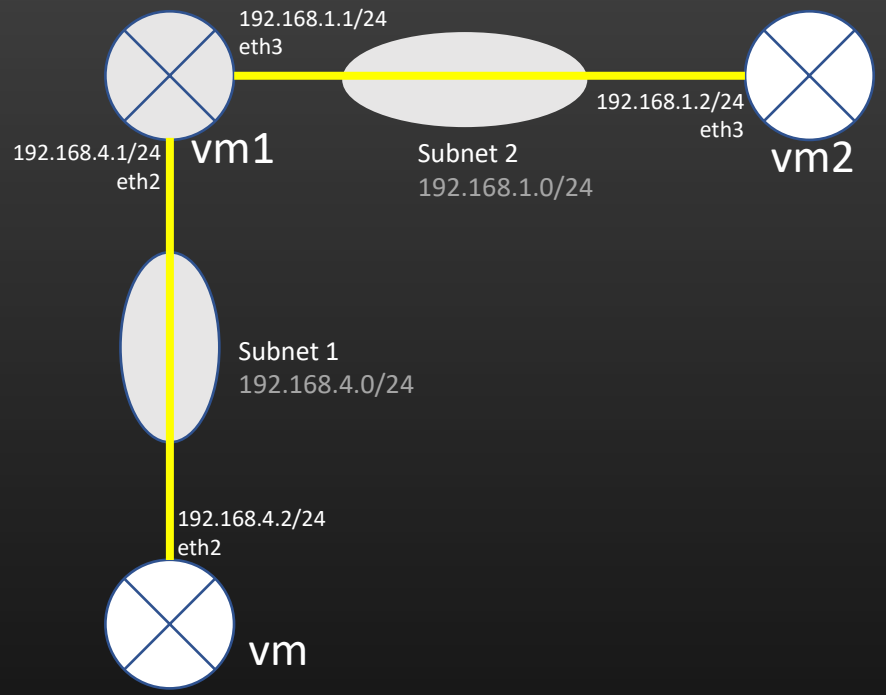
R2 -- > F

- The packet has to go to gateway L3 router if it has to leave its local subnet

Lecture VDO 5

Routing – L3 Routing -> Routing table

- Let us take the example of our own topology which contains three L3 routers
- Are you able to ping from VM to IP 192.168.4.1 ? YES
  - Because only L2 routing is required, src and dst are in same subnet
- Are you able to ping from VM1 to IP 192.168.1.2 ? YES
  - Because only L2 routing is required, src and dst are in same subnet
- Are you able to ping from VM to IP 192.168.1.2 Or 192.168.1.1 ? NO
  - Because L3 routing is required, src and dst are in different subnet
- Directly connected peer interfaces can always be pinged, only ARP is required
- VM do not know how to reach *remote subnet 192.168.1.0/24 (Subnet 2)
  Similarly, VM2 do not know how to reach subnet 192.168.4.0/24 (Subnet 1)

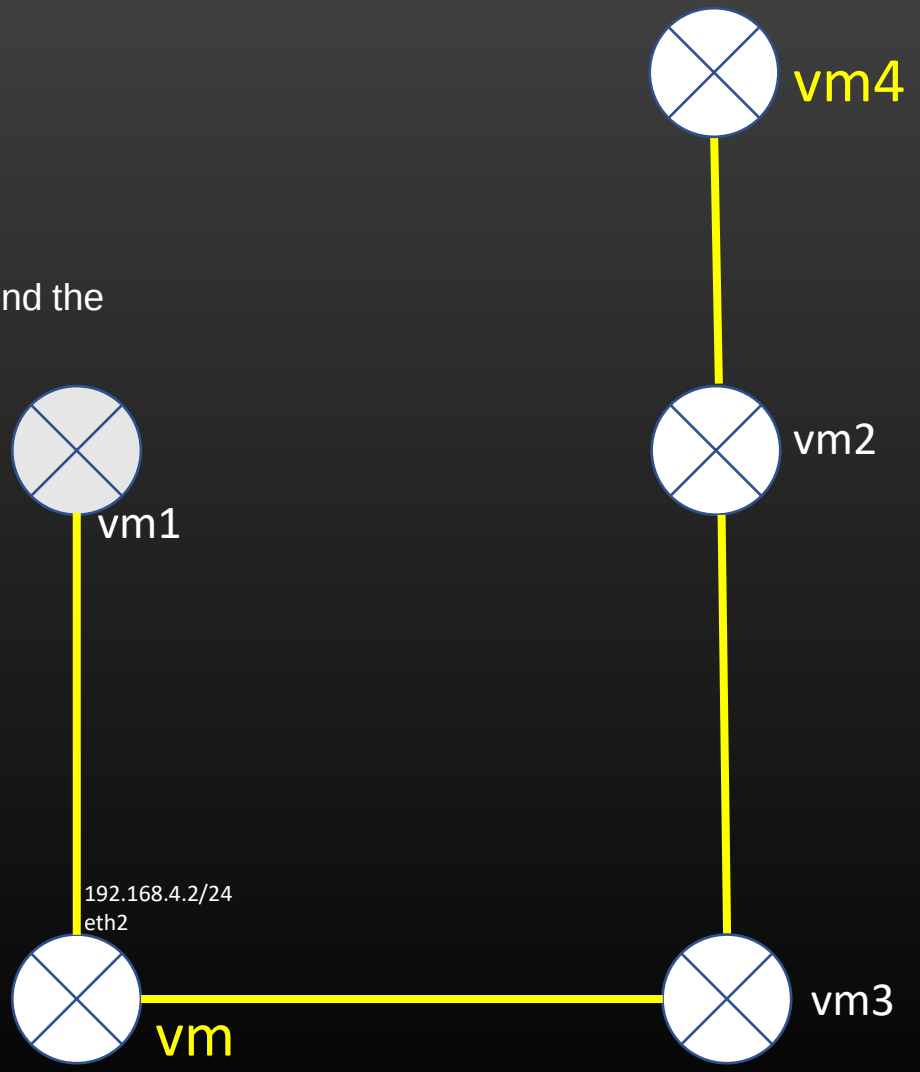


A subnet S is said to be remote subnet
Wrt to a given router A, if none of the local
Interfaces of router A has an ip address
Which is a member of subnet S

Lecture VDO 5

Routing – L3 Routing -> Routing table

- Another example
- If VM need to send the data to VM4, VM do not know where to send the Packet – towards VM1 Or VM3
- This intelligence is borne by L3 Routing tables
- Lets learn to configure routing table



Implementing ARP -> ARP Table -> APIs

Initialize the ARP table, should be called when a node is created during topology creation i.e. from *init_node_nw_prop(. . .)*

```
void init_arp_table (arp_table_t **arp_table);
```

CRUD Operations on ARP table :

C
```
bool_t arp_table_entry_add (arp_table_t *arp_table, arp_entry_t *arp_entry);
```

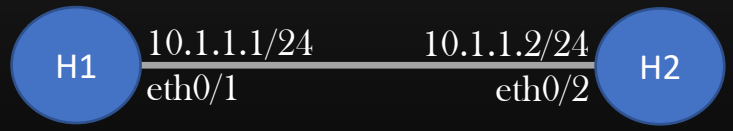
R
```
arp_entry_t * arp_table_lookup (arp_table_t *arp_table, char *ip_addr);
```

U
```
void arp_table_update_from_arp_reply (arp_table_t *arp_table, arp_hdr_t *arp_hdr, interface_t *iif);
```

D
```
void delete_arp_table_entry (arp_table_t *arp_table, char *ip_addr);
```

An API which triggers ARP resolution is :

```
void
send_arp_broadcast_request (node_t *node,
                            interface_t *oif,
                            char *ip_addr);
```



All ARP related APIs shall g in Layer2/layer2.h/.c

Lecture VDO 5

Routing – L3 Routing -> L3 Routing formation

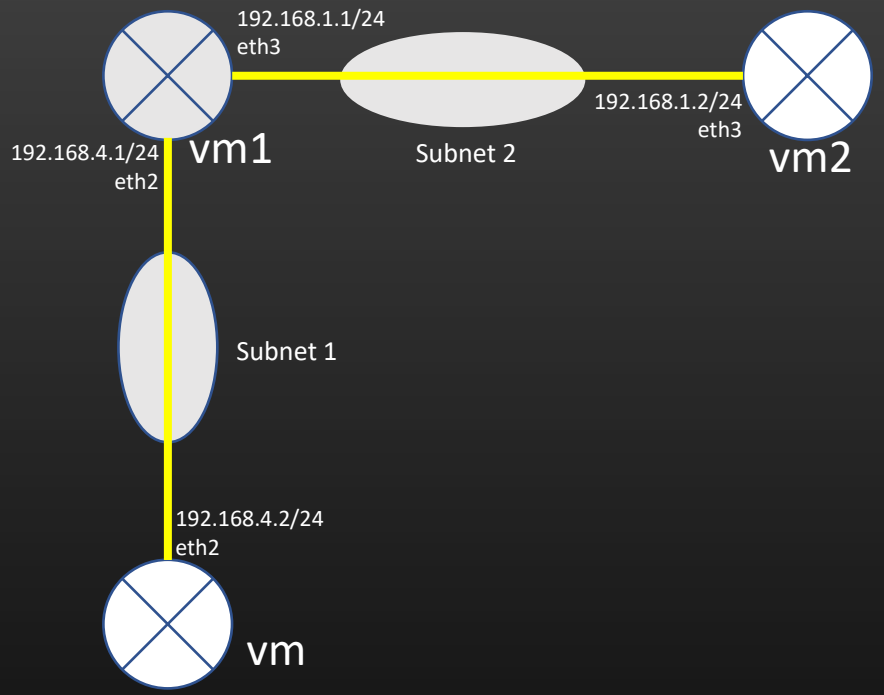
- Let us first understand what are L3 routes
- The L3 route is an entry in the routing table of a router/device which tells router the next router to forward the packet to so that packet eventually reaches the destination subnet
- L3 route is made up of 3 entities :

<destination subnet> <gateway/nxt hop ip> <OIF>

Subnet in which Destination machine resides

Peer interface ip which leads to Destination machine subnet

What is the local Out going interface To reach the peer interface



Example, let us teach VM how to reach remote subnet 192.168.1.0/24

up route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.4.1 dev eth2

From above entry machine VM knows that – Whenever it has to send packet to remote machine with IP = 192.168.1.[1-254], it has to send the packet out of interface eth2 towards peer interface whose ip is 192.168.4.1

Lecture VDO 5

Routing – L3 Routing -> L3 Routing formation

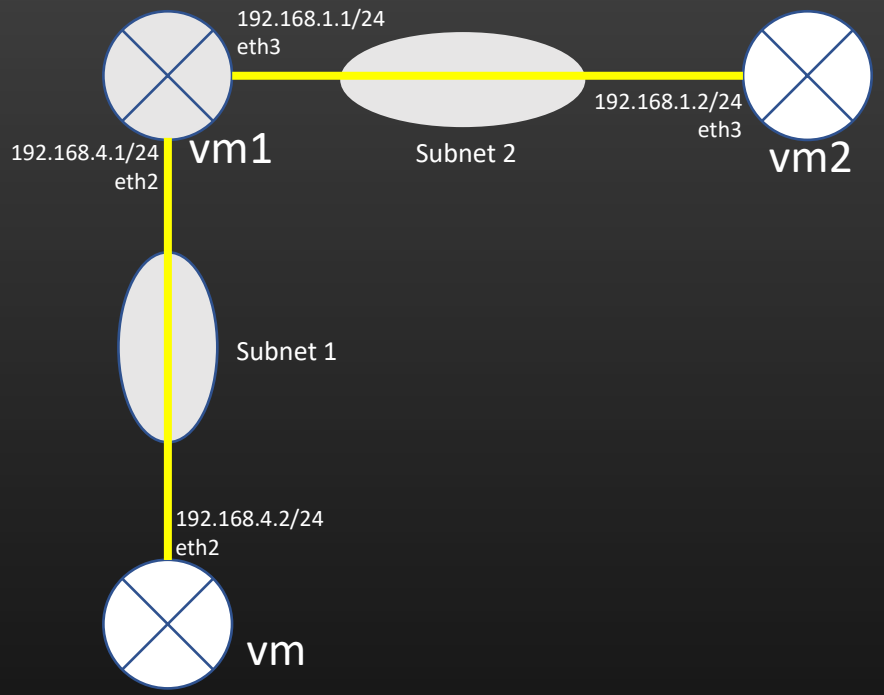
- Let us first understand what are L3 routes
- The L3 route is an entry in the routing table of a router/device which tells router the next router to forward the packet to so that packet eventually reaches the destination subnet
- L3 route is made up of 3 entities :

<destination subnet> <gateway/nxt hop ip> <OIF>

Subnet in which Destination machine resides

Peer interface ip which leads to Destination machine subnet

What is the local Out going interface To reach the peer interface



Example2, let us teach VM2 how to reach remote subnet 192.168.4.0/24

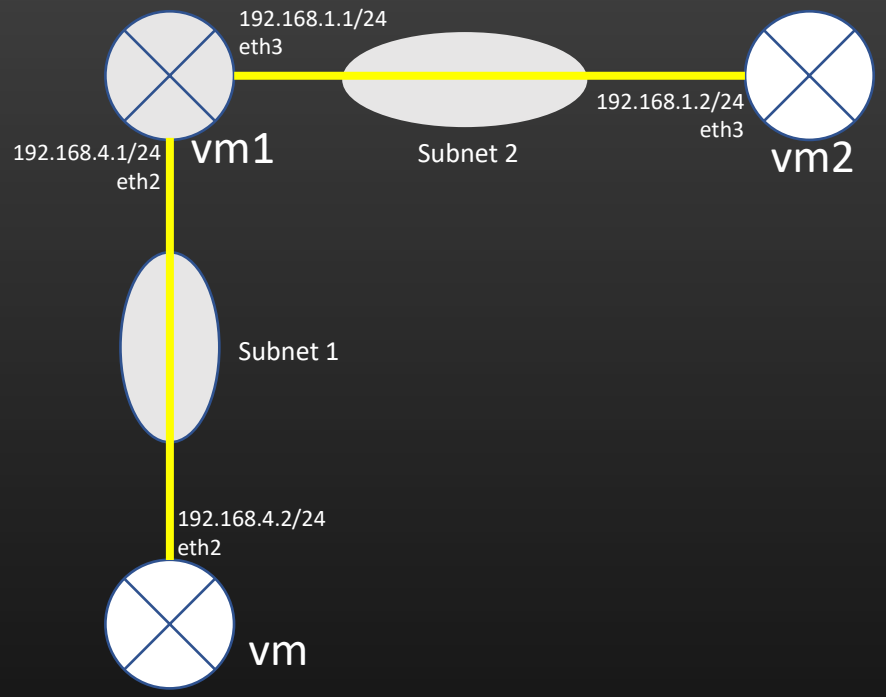
up route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.1.1 dev eth3

From above entry machine VM2 knows that – Whenever it has to send packet to remote machine with IP = 192.168.4.[1-254], it has to send the packet out of interface eth3 towards peer interface whose ip is 192.168.1.1

Lecture VDO 5

Routing – L3 Routing -> Routing table

- Now ping from VM to 192.168.1.1 Or 192.168.1.2
- Now ping from VM2 to 192.168.4.1 Or 192.168.4.2
- Conclusion
  - L3 routing entries are required to reach any non local subnet



Lecture VDO 5

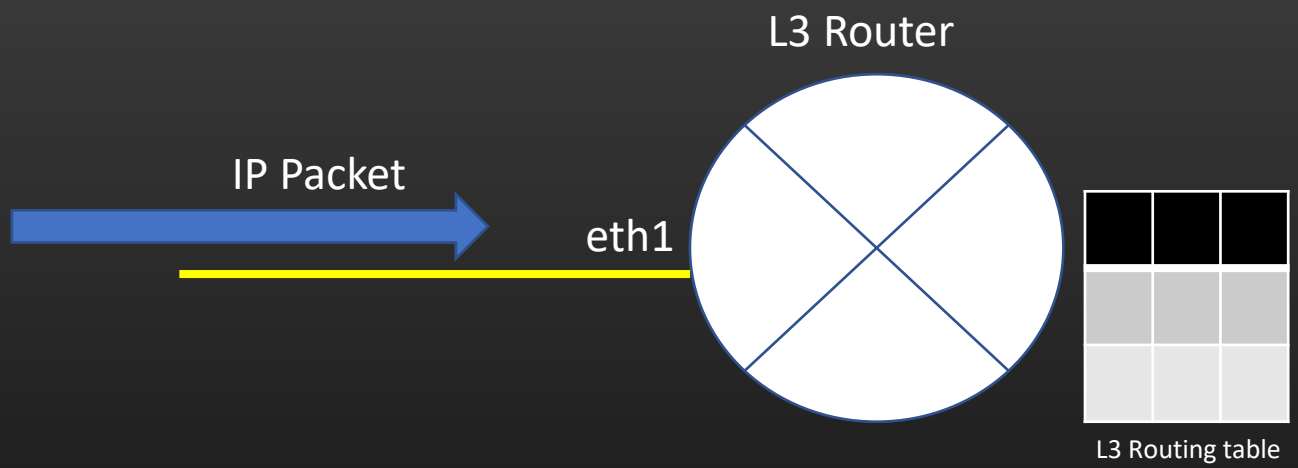
Routing – L3 Routing -> Route look up

Routing table

L3 route look up

Lecture VDO 5

Routing – L3 Routing -> Route look up

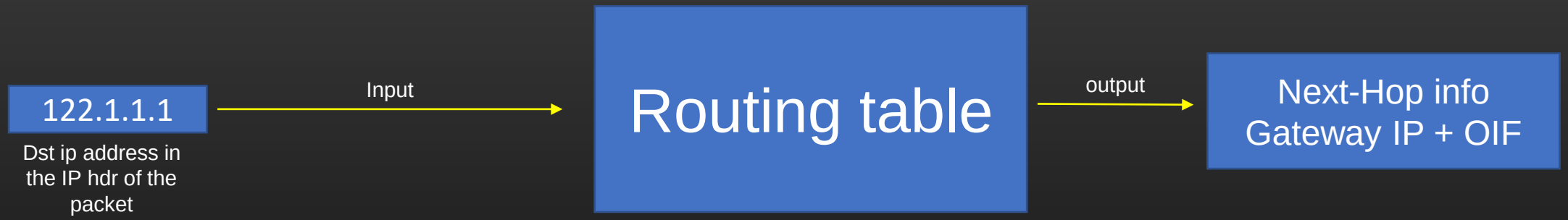


- How does L3 Router decides what it has to do with the packet ?
  - Should it consume the packet ?
  - Should it forward the packet to next Router ?
  - Should it forward the packet to a machine in direct local subnet ?
  - Should it silently discard the packet ?

Router Decides by looking up the matching entry in the L3 routing table !!

Lecture VDO 5

Routing – L3 Routing -> Route look up



L3 route look up

Lecture VDO 5

Routing – L3 Routing -> Route look up

1. Apply mask to input ip address
2. Check if result of step 1 is equal to *Dest* field, if yes match is found
3. If two or more matching entries are found, one with higher subnet mask is chosen.
4. This is called *"longest prefix match"*

Dst ip in pkt

92.168.5.10

Input

122.1.1.15

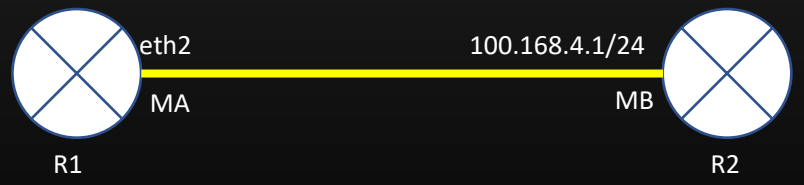
Dest	Gateway	Mask					OIF
122.1.1.0	100.168.4.1	255.255.255.0	UG	0	0	0	eth2
10.1.1.0	192.168.4.1	255.255.255.0	UG	0	0	0	eth2
92.168.5.0	192.168.1.2	255.255.255.0	UG	0	0	0	eth3
122.1.0.0	200.168.1.1	255.255.0.0	UG	0	0	0	eth3
10.1.4.0	190.16.5.1	255.255.255.0	UG	0	0	0	eth2
12.18.1.0	192.168.5.1	255.255.255.0	UG	0	0	0	eth2

Lecture VDO 5

Routing – L3 Routing -> Route look up

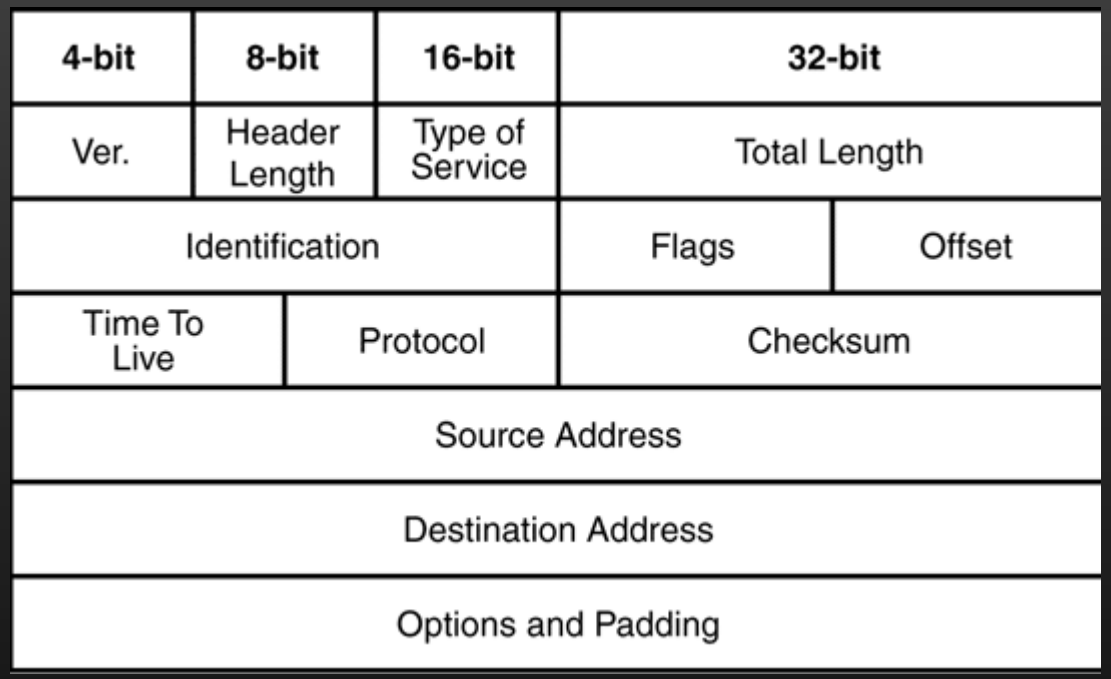
122.1.1.15

Dest	Gateway	Mask					OIF
122.1.1.0	100.168.4.1	255.255.255.0	UG	0	0	0	eth2
10.1.1.0	192.168.4.1	255.255.255.0	UG	0	0	0	eth2
92.168.5.0	192.168.1.2	255.255.255.0	UG	0	0	0	eth3
122.1.0.0	200.168.1.1	255.255.0.0	UG	0	0	0	eth3
10.1.4.0	190.16.5.1	255.255.255.0	UG	0	0	0	eth2
12.18.1.0	192.168.5.1	255.255.255.0	UG	0	0	0	eth2



Lecture VDO 5

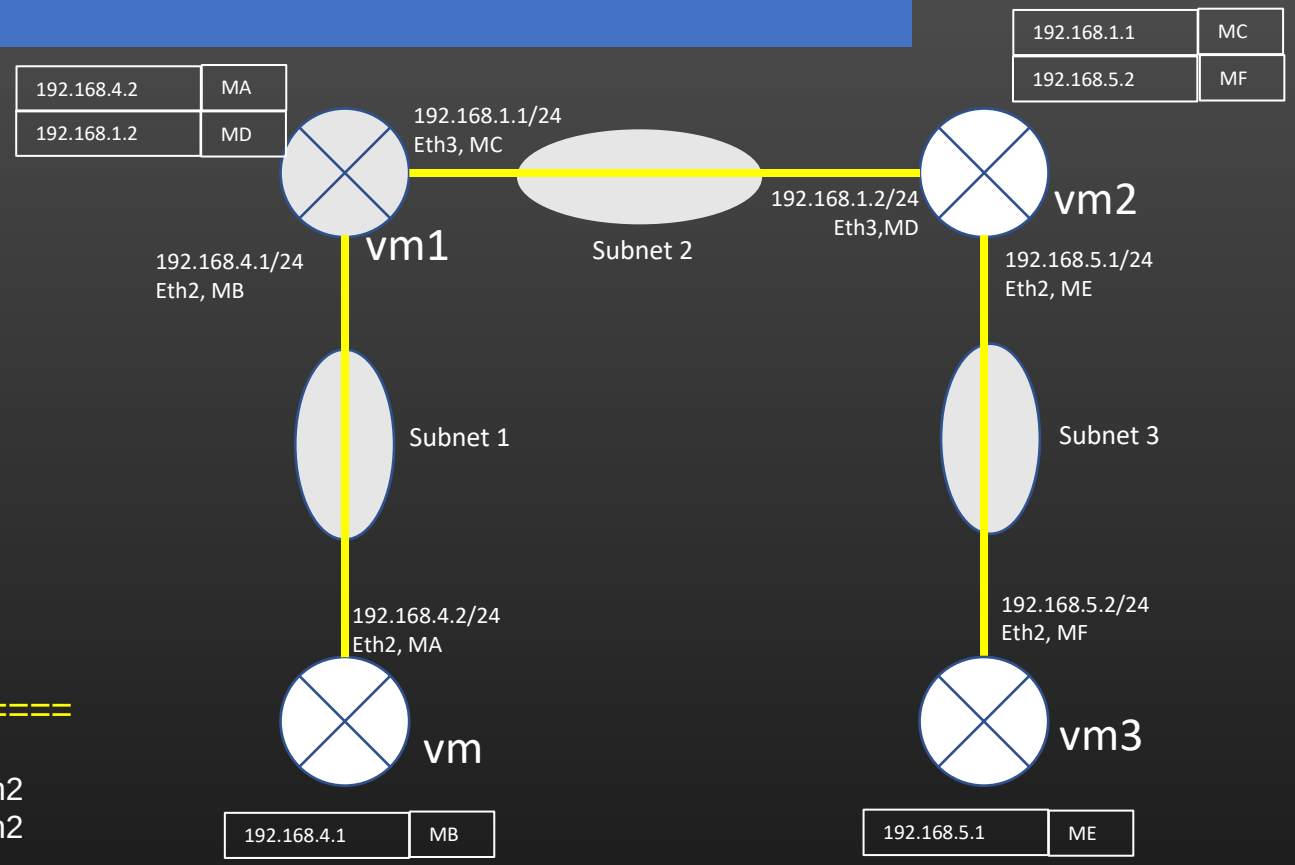
Routing – L3 Routing -> IP Header



Lecture VDO 5

Routing – L3 Routing -> Routing table

- Let us take an example of L3 routing in 4 node topology



For VM, remote subnets are Subnet2 (192.168.1.0/24) and Subnet 3 (192.168.5.0/24)
For VM1, remote subnets is Subnet3 only (192.168.5.0/24)
For VM2, remote subnets is Subnet1 only (192.168.4.0/24)
For VM3, remote subnets are Subnet2 (192.168.1.0/24) and Subnet1 (192.168.4.0/24)

Dest	Gateway	Mask					OIF
VM route entry							
192.168.5.0	192.168.4.1	255.255.255.0	UG	0	0	0	eth2
192.168.1.0	192.168.4.1	255.255.255.0	UG	0	0	0	eth2
VM1 route entry							
192.168.5.0	192.168.1.2	255.255.255.0	UG	0	0	0	eth3
VM2 route entry							
192.168.4.0	192.168.1.1	255.255.255.0	UG	0	0	0	eth3
Vm3 route entry							
192.168.4.0	192.168.5.1	255.255.255.0	UG	0	0	0	eth2
192.168.1.0	192.168.5.1	255.255.255.0	UG	0	0	0	eth2

When a packet is routed in the network, Essentially four fields in the packet we need to closely understand :

Src mac and ***Dst mac*** in Ethernet hdr
Src ip and ***Dst ip*** in IP hdr

There is no L2 switch in our topo, so MAC table is out of Question
Also, we assume ARP tables of all Routers in the network are already populated (we are focusing on L3 routing alone as of now)

Lecture VDO 5

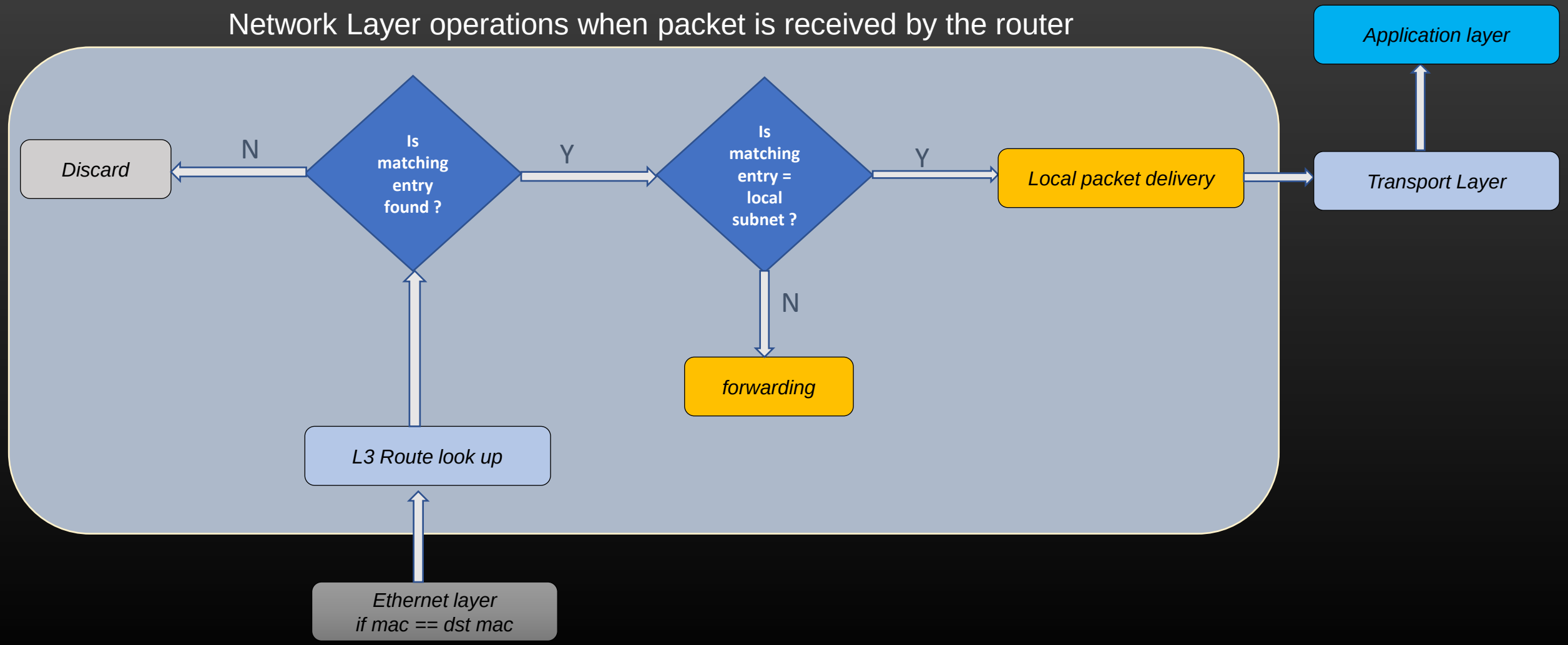
Routing – L3 Routing -> Network Layer operations

- **Summarizing Network Layer operations**
  - Op1 : Route lookup (to decide if the received or locally generated pkt is destined to machine in the local subnet or remote subnet based on dst ip address)
  - Op2 : Local packet delivery (if the dst ip address in the received pkt = ip address of one of the local interface)
  - Op3 : forwarding, (if the dst ip address of received pkt != ip address of one of the local interface AND routing entry is present in routing table for dst ip)
  - Op4 : preparing ip hdr (sending out locally generated packets)
  - Op5 : discard the packet (if dst ip address in the incoming pkt doesn't match any route in routing table)

Let us see how these operations of network layer maps in the next example

Lecture VDO 5

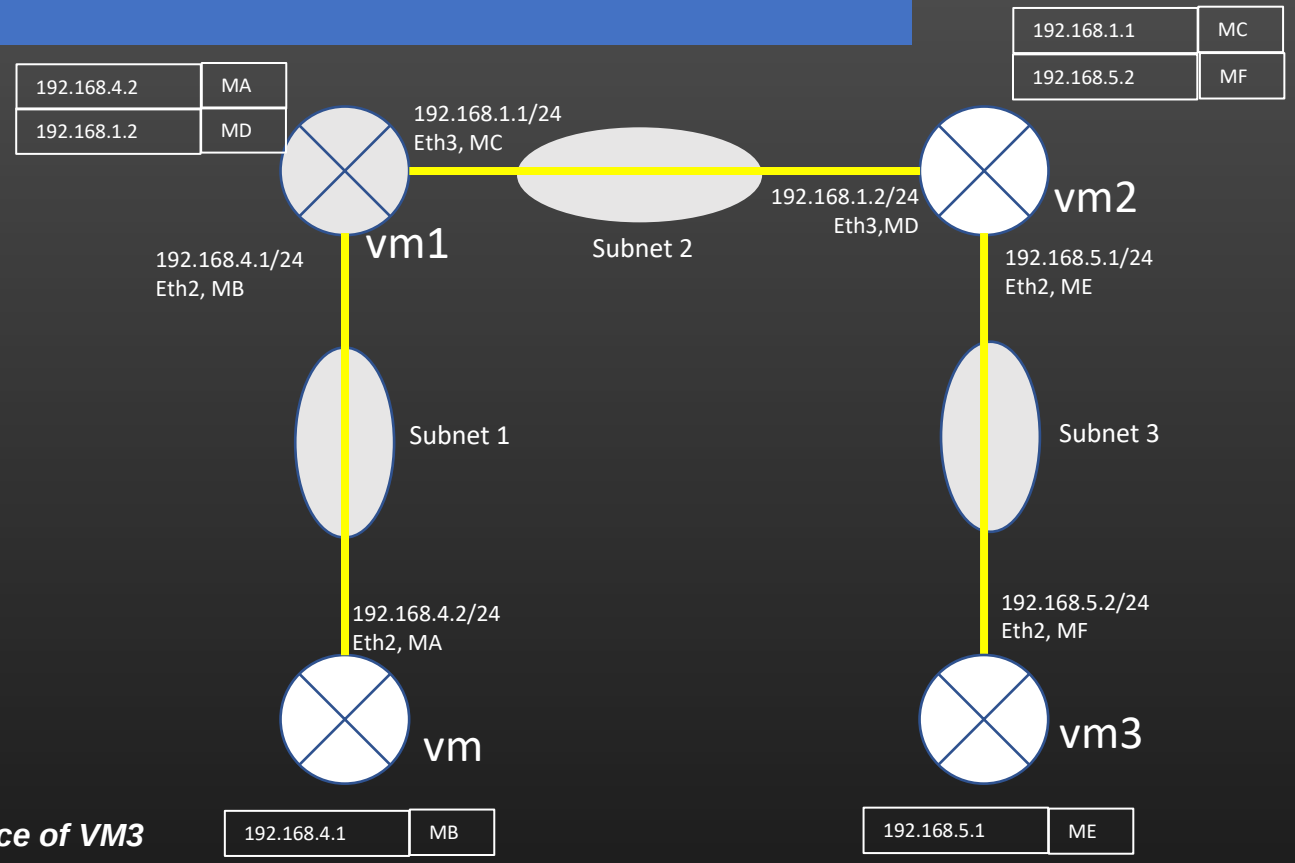
Routing – L3 Routing -> Network Layer operations



Lecture VDO 5

Routing – L3 Routing -> Example

Dest	Gateway	Mask				OIF
VM route entry						
192.168.5.0	192.168.4.1	255.255.255.0	UG	0	0	0	eth2
192.168.1.0	192.168.4.1	255.255.255.0	UG	0	0	0	eth2
VM1 route entry						
192.168.5.0	192.168.1.2	255.255.255.0	UG	0	0	0	eth3
VM2 route entry						
192.168.4.0	192.168.1.1	255.255.255.0	UG	0	0	0	eth3
Vm3 route entry						
192.168.4.0	192.168.5.1	255.255.255.0	UG	0	0	0	eth2
192.168.1.0	192.168.5.1	255.255.255.0	UG	0	0	0	eth2



Example :

Let say VM wants to send data packet to 192.168.5.2 which is local interface of VM3

MA	MB	192.168.4.2	192.168.5.2
Src mac	Dst mac	Src ip	Dst ip

Step 1 : VM wants to send data pkt to 192.168.5.2 (dst ip). VM checks if the dst ip resides to one of its local subnet. In this case, dst ip belongs to remote subnet.

Step 2 : VM checks its routing table and search for a Dest subnet which hosts dst ip. In this case. Entry #1 is picked up.

Step 3 : Now VM knows, it has to send the pkt to GateWay (nxt hop) 192.168.4.1 from OIF eth2

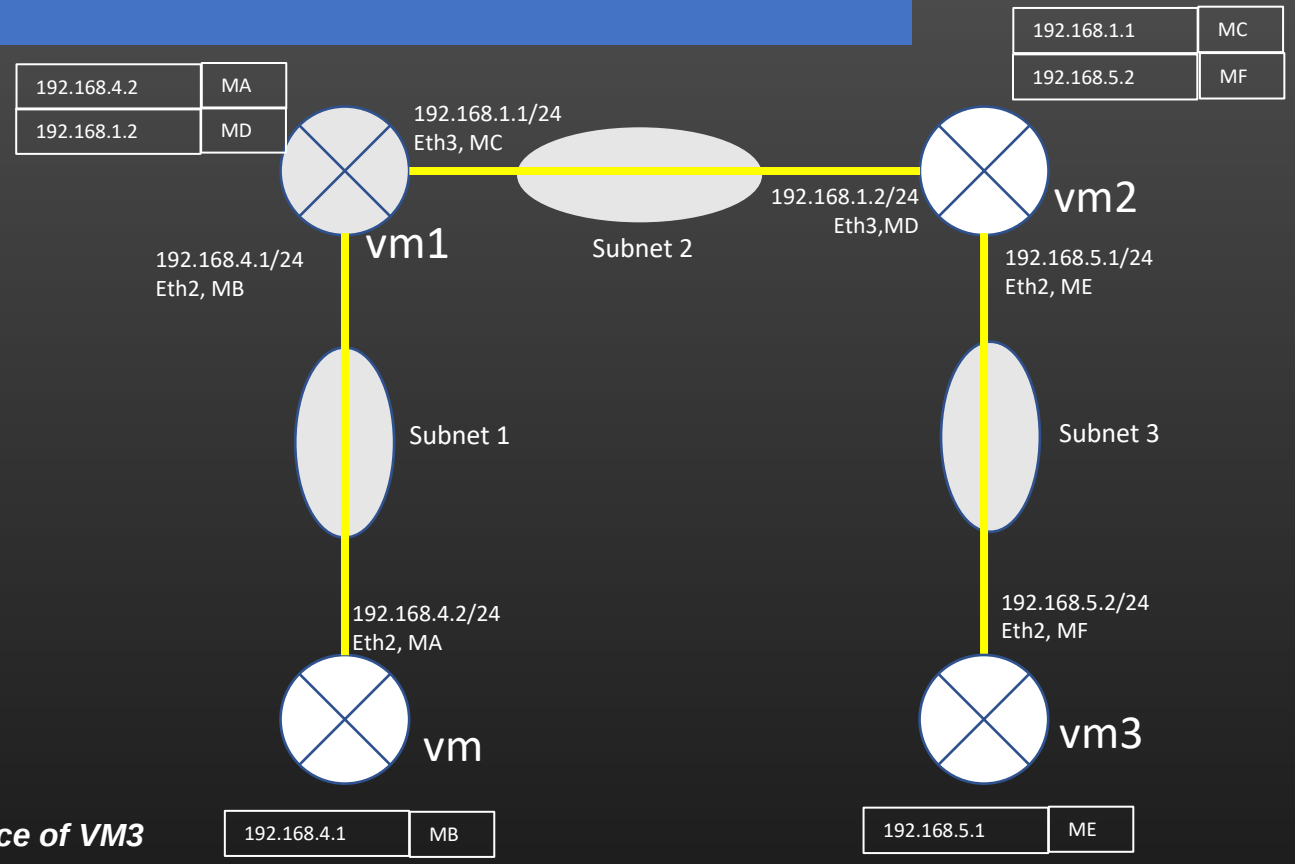
Step 4 : VM now look up its ARP table using 192.168.4.1 as a key to know Nxt hop MAC address which is MB

Step 5 : VM now prepares the packet, and send the packet out of interface eth2 (OIF)

Lecture VDO 5

Routing – L3 Routing -> Example

Dest	Gateway	Mask				OIF
VM route entry						
192.168.5.0	192.168.4.1	255.255.255.0	UG	0	0	0	eth2
192.168.1.0	192.168.4.1	255.255.255.0	UG	0	0	0	eth2
VM1 route entry						
192.168.5.0	192.168.1.2	255.255.255.0	UG	0	0	0	eth3
VM2 route entry						
192.168.4.0	192.168.1.1	255.255.255.0	UG	0	0	0	eth3
Vm3 route entry						
192.168.4.0	192.168.5.1	255.255.255.0	UG	0	0	0	eth2
192.168.1.0	192.168.5.1	255.255.255.0	UG	0	0	0	eth2



Example :

Let say VM wants to send data packet to 192.168.5.2 which is local interface of VM3

MC	MD	192.168.4.2	192.168.5.2
Src mac	Dst mac	Src ip	Dst ip

Step 6 : VM1 receives the packet. Ethernet layer checks if dest mac = mac of receiving interface ? Yes, then chop off the Ethernet hdr and handover the packet to Network Layer

Step 7 : VM1 (Network Layer) checks whether dst ip in the packet belongs to any of its local subnet. In this case, No. Then, VM1 checks routing table and search for a Dest subnet which hosts dst ip. In this case. Entry #1 (the only entry) is picked up.

Step 8 : Now VM1 knows, it has to send the pkt to GateWay (nxt hop) 192.168.1.2 from OIF eth3

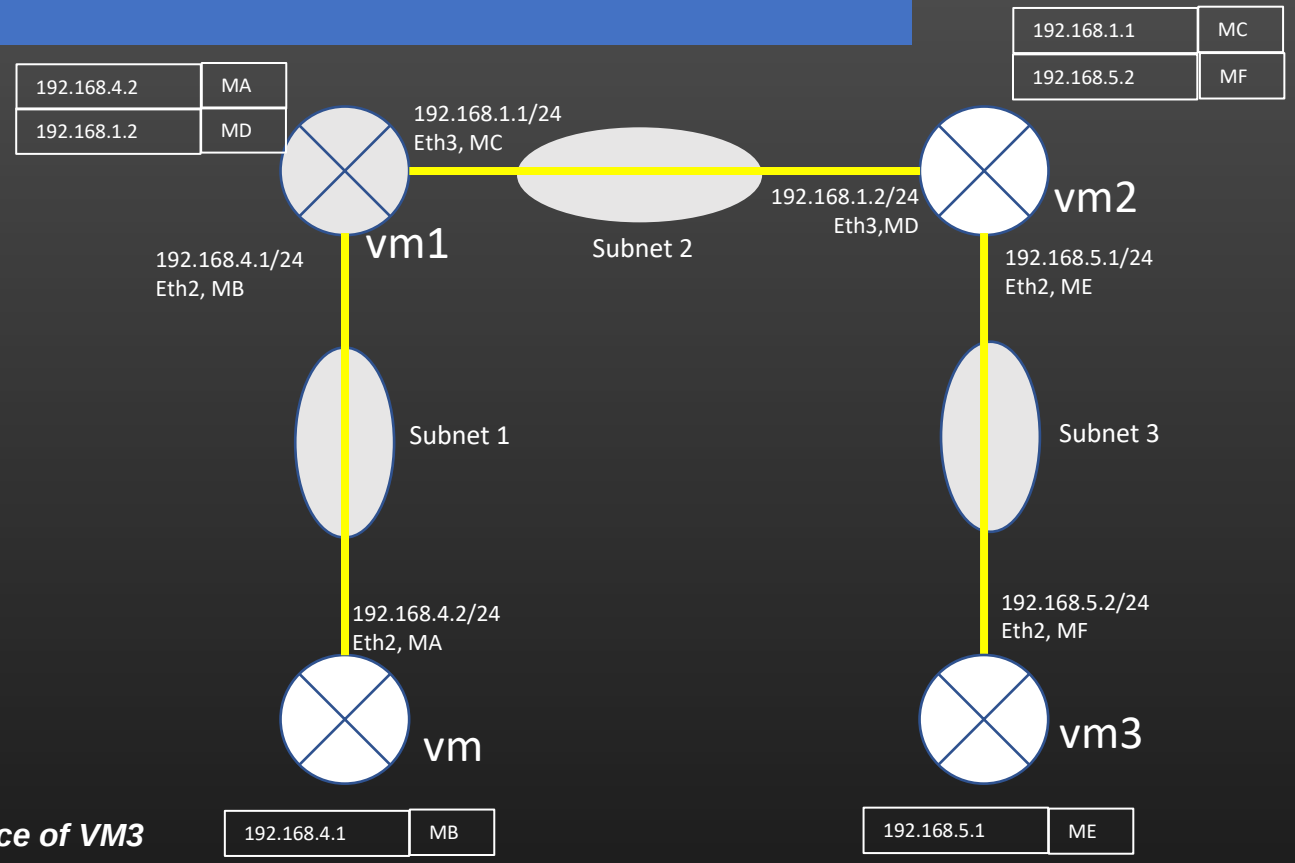
Step 9 : VM1 now look up its ARP table using Nxt hop Ip = 192.168.1.2 as a key to know Nxt hop MAC address which is MD

Step 10 : VM1 now prepares the packet, and send the packet out of interface eth3 (OIF)

Lecture VDO 5

Routing – L3 Routing -> Example

Dest	Gateway	Mask				OIF
VM route entry						
192.168.5.0	192.168.4.1	255.255.255.0	UG	0	0	0 eth2
192.168.1.0	192.168.4.1	255.255.255.0	UG	0	0	0 eth2
VM1 route entry						
192.168.5.0	192.168.1.2	255.255.255.0	UG	0	0	0 eth3
VM2 route entry						
192.168.4.0	192.168.1.1	255.255.255.0	UG	0	0	0 eth3
Vm3 route entry						
192.168.4.0	192.168.5.1	255.255.255.0	UG	0	0	0 eth2
192.168.1.0	192.168.5.1	255.255.255.0	UG	0	0	0 eth2



Example :

Let say VM wants to send data packet to 192.168.5.2 which is local interface of VM3

ME	MF	192.168.4.2	192.168.5.2
Src mac	Dst mac	Src ip	Dst ip

Step 11 : VM2 receives the packet. Ethernet layer checks if dest mac = mac of receiving interface ? Yes, then chop off the Ethernet hdr and handover the packet to Network Layer

Step 12 : VM2 (Network Layer) checks whether dst ip in the packet belong to any of its local subnet. In this case, Yes, dst ip belongs to subnet 3.

Step 13 : Now VM2 will deliver the packet to VM3 using L2 routing only.

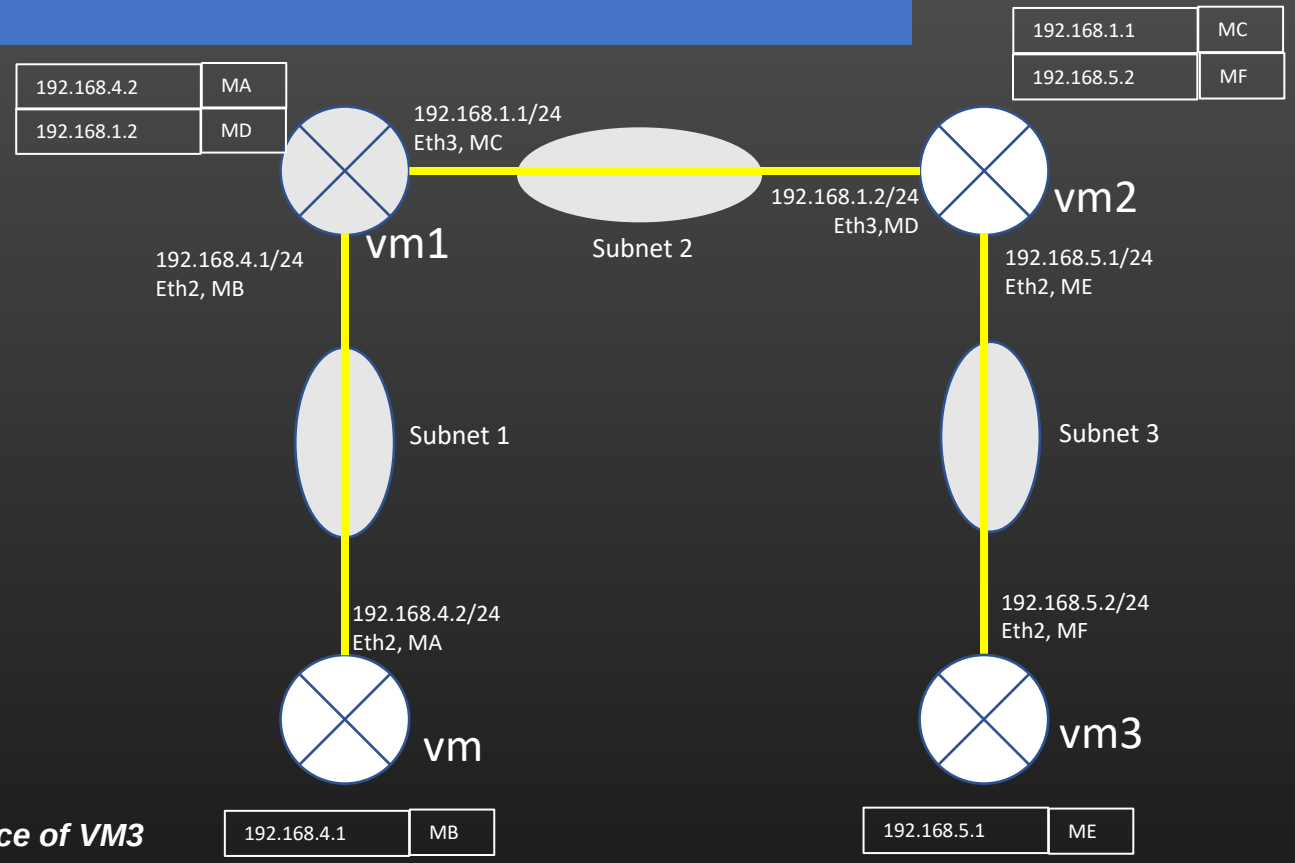
Step 14 : VM2 look up its ARP table using dst ip = 192.168.5.2 as a key to know Nxt hop MAC address which is MF.

Step 15 : VM2 now prepares the packet, and send the packet out of interface eth2 (OIF)

Lecture VDO 5

Routing – L3 Routing -> Example

Dest	Gateway	Mask				OIF
VM route entry						
192.168.5.0	192.168.4.1	255.255.255.0	UG	0	0	0 eth2
192.168.1.0	192.168.4.1	255.255.255.0	UG	0	0	0 eth2
VM1 route entry						
192.168.5.0	192.168.1.2	255.255.255.0	UG	0	0	0 eth3
VM2 route entry						
192.168.4.0	192.168.1.1	255.255.255.0	UG	0	0	0 eth3
Vm3 route entry						
192.168.4.0	192.168.5.1	255.255.255.0	UG	0	0	0 eth2
192.168.1.0	192.168.5.1	255.255.255.0	UG	0	0	0 eth2



Example :

Let say VM wants to send data packet to 192.168.5.2 which is local interface of VM3

X	X	X	X
Src mac	**Dst mac**	**Src ip**	**Dst ip**

Step 16 : VM3 receives the packet. Ethernet layer checks if dest mac = mac of receiving interface ? Yes, then chop off the Ethernet hdr and handover the packet to Network Layer

Step 17 : VM3 (Network Layer) checks whether dst ip in the packet belong to any of its local subnet. In this case, Yes, dst ip belongs to subnet 3.

Step 18 : VM3 also checks that dst ip in the packet exactly matches to the IP of one of its local interface (eth2). VM3 now knows that packet was destined to itself.

Step 19 : VM3 chop off the network layer hdr and deliver rest of the packet to Transport Layer.

Step 20 : Transport layer processes the packet

Lecture VDO 5

Routing – Complete Routing Example

WATCH this YOUTUBE Video

This is the best Video I could find for you on internet

Watch the VDO again and again until you can explain the VDO to a friend without any second thoughts

In Interviews, often you will be asked to explain the concept of this entire Video

That’s All About Routing.

Lecture VDO 5

Routing – Assignment

- We will do Assignment on L3 route installation
- In this assignment, we will learn Creating a Topology of Routers
- We shall install the L3 routes in the router's L3 routing table
- We will see how routers route the traffic destined for a given destination IP address
- It is a lot of exercise to deploy and configure the topology using VMs every time
- So, we shall be using a software using which we can create a topology quickly and install the L3 routes
- Through this assignment, you will learn how to install L3 routes in the routers, and
- You can create traffic loops if you install inconsistent routes in the router's L3 routing table
- Let's begin . . .

Lecture VDO 5

Routing – Assignment

1. Create a topology

```
conf topo /
node /
/*Create nodes*/
create R0
create R1
create R2
create R3
create R4
/*Configure loopback IPs*/
R0 loopback 192.168.0.1
R1 loopback 192.168.0.2
R2 loopback 192.168.0.3
R3 loopback 192.168.0.4
R4 loopback 192.168.0.5
/*insert links*/
R0 from-if eth0/0 peer R1 to-if eth0/1
R1 from-if eth0/0 peer R2 to-if eth0/1
R2 from-if eth0/2 peer R3 to-if eth0/10
R3 from-if eth0/1 peer R4 to-if eth0/2
R4 from-if eth0/1 peer R0 to-if eth0/2
/*configure ip addresses*/
R0 interface eth0/0 ip 10.1.1.1 30
R0 interface eth0/2 ip 4.1.1.2 30
R1 interface eth0/1 ip 10.1.1.2 30
R1 interface eth0/0 ip 1.1.1.1 30
R2 interface eth0/1 ip 1.1.1.2 30
R2 interface eth0/2 ip 2.1.1.1 30
R3 interface eth0/1 ip 3.1.1.1 30
R3 interface eth0/10 ip 2.1.1.2 30
R4 interface eth0/1 ip 4.1.1.1 30
R4 interface eth0/2 ip 3.1.1.2 30
cd
```

Lecture VDO 5

Routing – Assignment

2. Install L3 routes

conf node /
R0 static-route 192.168.0.4 32 10.1.1.2 eth0/0
R1 static-route 192.168.0.4 32 1.1.1.2 eth0/0
R2 static-route 192.168.0.4 32 2.1.1.2 eth0/2
R3 static-route 192.168.0.4 32 - -
cd

3. Verify the routing table entries

show instance node <node-name> inet.0 forwarding

Eg :
show instance node R0 inet.0 forwarding

4. Verify the routing path

show instance node <node-name> traceroute <ip-address>

Eg :
show instance node R0 traceroute 192.168.0.4

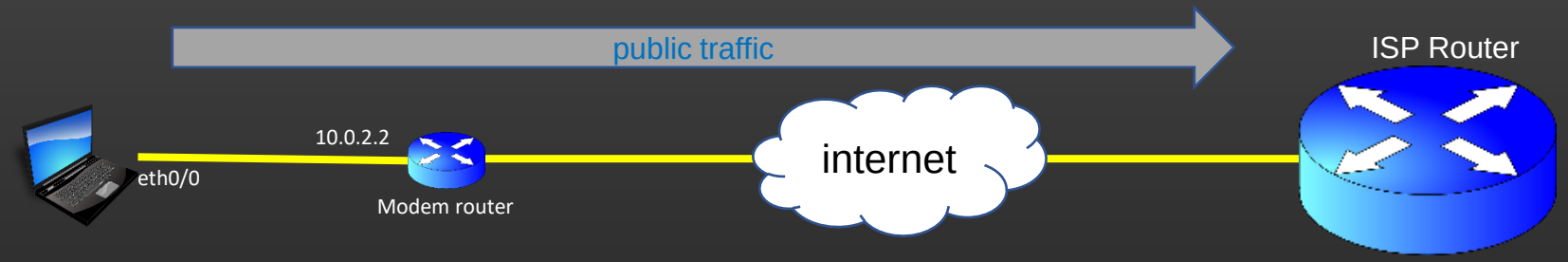
Lecture VDO 5

Routing – Default Route

- If you are connected to the internet, and you ping www.google.com , you are always able to ping
- Similarly, try pinging facebook.com , etc
- Try . . .
- Every public website resolves to the public IP Address, How are you able to ping all of them if you have never taught your machine to reach those public IP addresses ?
- We are still able to ping any public IP address without installing any L3 route in our routing table. How is it possible ? How are you able to access google.com / facebook.com ?
- That's What *default route* does

Lecture VDO 5

Routing – Default Route

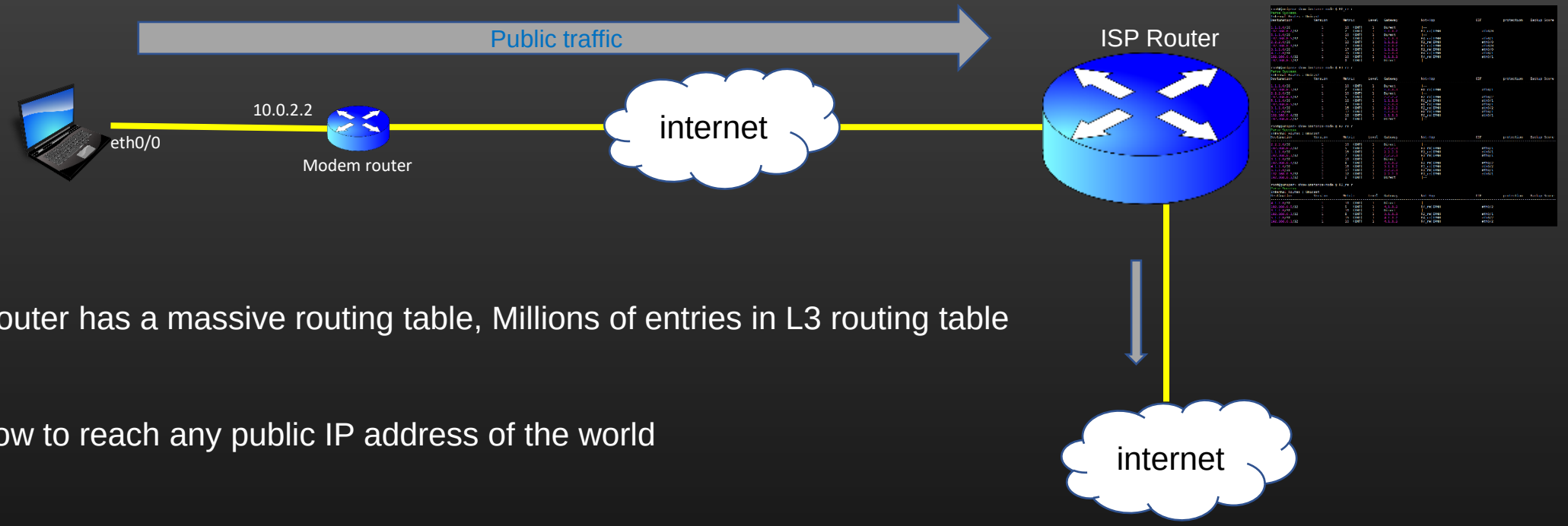



```
vm@vm:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    0      0        0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U     1      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth1
192.168.1.0     192.168.4.1     255.255.255.0   UG    0      0        0 eth2
192.168.4.0     0.0.0.0         255.255.255.0   U     0      0        0 eth2
192.168.56.0    0.0.0.0         255.255.255.0   U     0      0        0 eth1
vm@vm:~$
```

- 0.0.0.0/0 is the default routing entry
- Any IP address which fails longest prefix match criteria with all other entries of the routing table,
- finally matches the default route (apply LPM)
- Default entry simply route the unknown traffic to your ISP router

Lecture VDO 5

Routing – Default Route



- The ISP Router has a massive routing table, Millions of entries in L3 routing table
- It knows how to reach any public IP address of the world
- This is how, you are able to access any public servers/websites without having to worry about your routing table
- Default route is used to forward the traffic when destination ip address do not match any L3 routing entry in L3 routing table

Lecture VDO 5

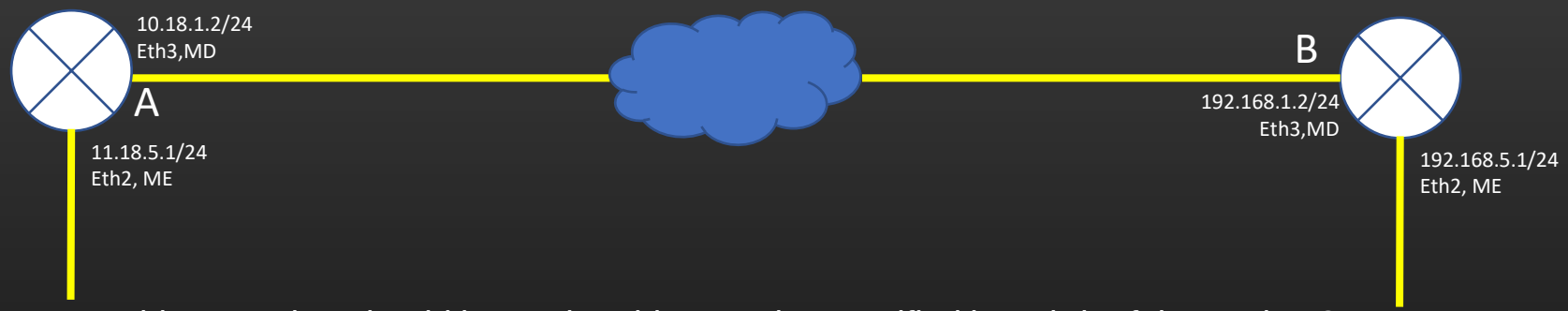
Routing – L3 Routing -> Loopback Interface

Loopback interface

Lecture VDO 5

Routing – L3 Routing -> Loopback Interface

- When we send a packet destined to a remote machine, What is the IP address we should specify as Source and Destination IP address ?



- If A wants to send data to machine B, What should be src ip address to be specified in IP hdr of the packet ?
  10.18.1.2 Or 11.18.5.1

- If A wants to send a data to router B, what should be dest IP address to be specified in the packet ?
  192.168.1.2 Or 192.168.5.1

  Ans is : A can specify Src ip = IP of any local interface, and
  Dst ip = IP of any local interface of B , Lets call it IP-RULE

  Thus, any combination of src ip = 10.18.1.2 Or 11.18.5.1 AND dst ip = 192.168.1.2 Or 192.168.5.1 would lead to same result : That is packet Would be delivered to B by the network layer

Lecture VDO 5

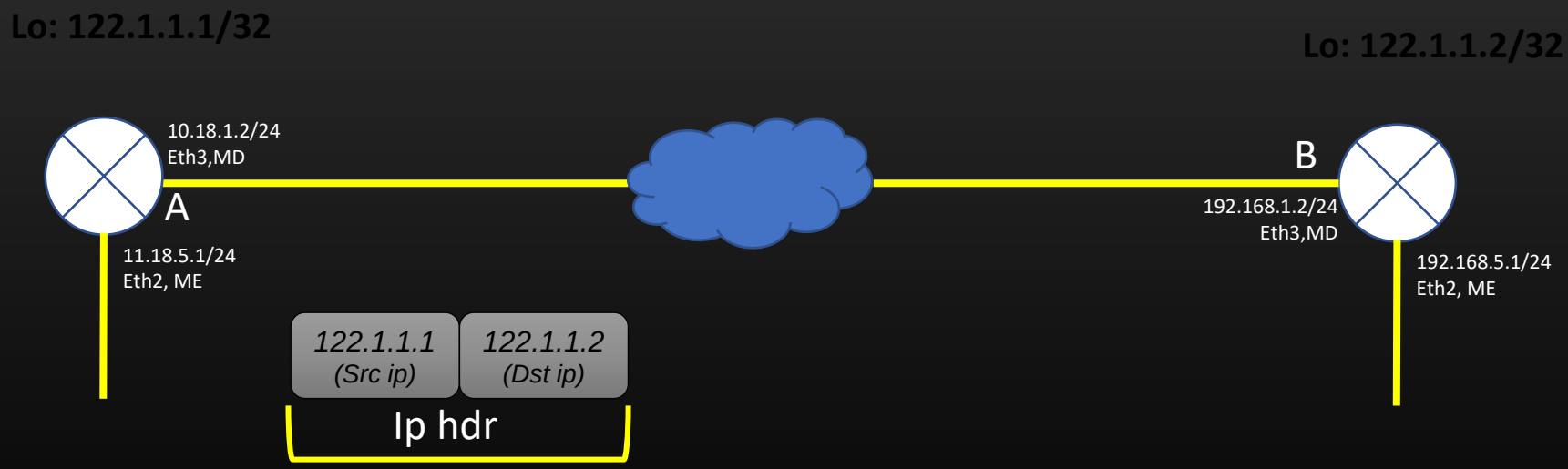
Routing – L3 Routing -> Loopback Interface

- Network layer view the addresses as some tags/labels attached to L3 routers.
- Network layer is not aware that IP addresses are tied to interfaces.

Lecture VDO 5

Routing – L3 Routing -> Loopback Interface

- There is a need to come up with special ip addresses which identify the L3 router as a device in the network and should be unique.
- Lo Addresses help achieve this goal.
- Lo addresses are used to represent the identity of a Networking device



Lecture VDO 5

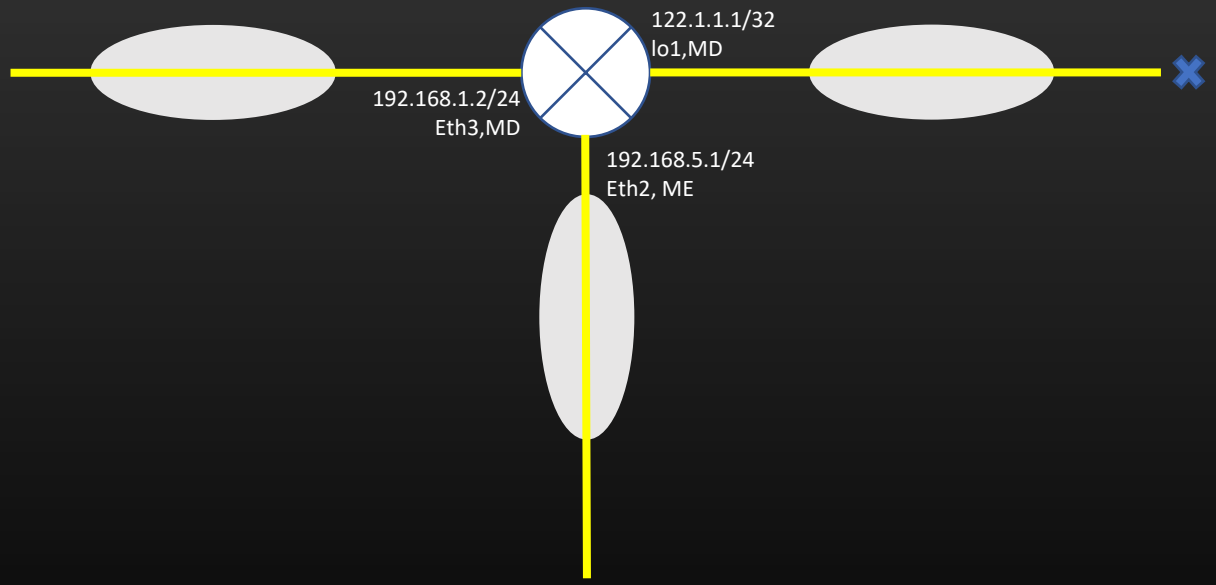
Routing – L3 Routing -> Loopback Interface

- Only Interfaces have IP addresses, Not machines.
- But there should be some way to identify the machine uniquely through IP address in the network.
- For this purpose, we create a special ***software interface*** called loopback interface on the machine, and assign IP address to it.
- loopback interface is not a hardware , but software based interface, and therefore you can create as many as you want
- Loopback interfaces are not prone to hardware failures, because they are not hardware's – High Availability
- Therefore, if A creates a lo1 interface with ip = 12.1.1.1/32 and , B creates a lo1 interface with ip = 13.1.1.1/32,
  - A can send packet to B with src ip = 12.1.1.1 and dst ip = 13.1.1.1 also
  - IP-Rule is still valid, since loopback are also local interface of the machine
- It is a good practice to assign src ip = ip of loopback interface of src machine, dst ip = ip of loopback interface of destination machine.
- Lo interfaces are not used for any traffic exchange – it serves the purpose of device reachability in the network
  - Eg : Machine A can send the data to machine B using B's loopback address as Destination address.
  - Once packet reaches to Machine B, Machine B consumes the packet.
  - B do not forward the packet out of loopback interface because loopback interface is a conceptual concept, and not a physical interface.

Lecture VDO 5

Routing – L3 Routing -> Loopback Interface

- View Loopback interface as just another interface
- Loopback address itself is one subnet with mask value always 32



- loopback subnet (A.B.C.D/32) is always a local subnet for the local router and remote subnet for any other Network device

Lecture VDO 5

Routing – L3 Routing -> L3 routing using loopback ip address

L3 Routing

Using Loopback IP address

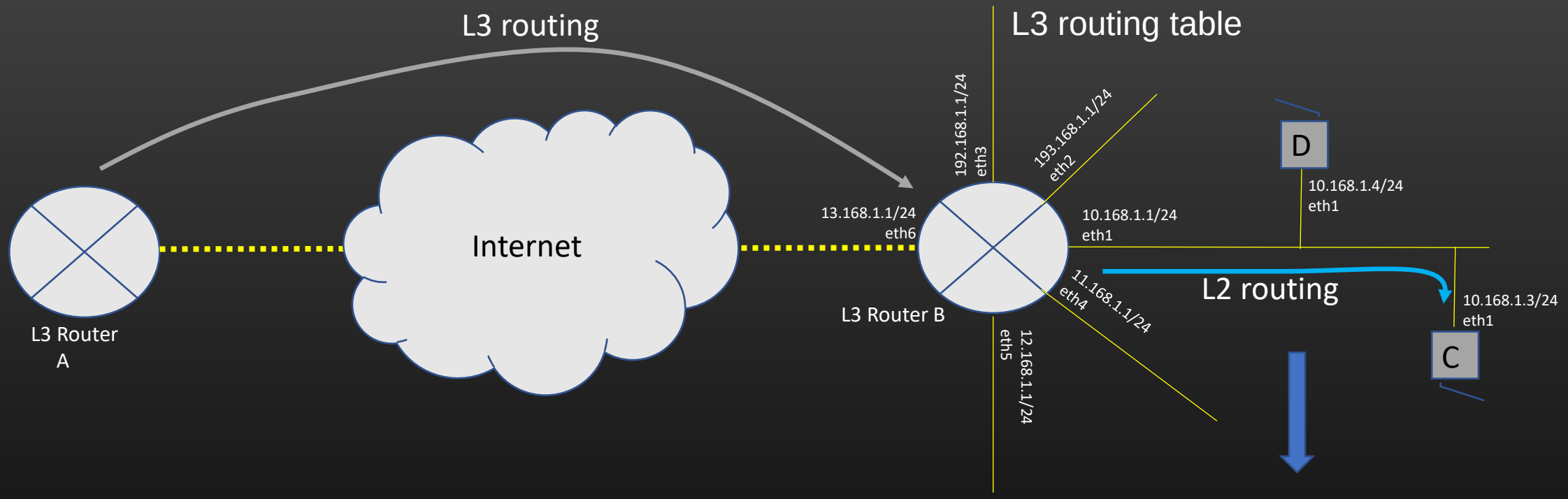
As

Destination Address

Lecture VDO 5

Routing – L3 Routing -> L3 routing using loopback ip address

Recap



Destination ip in the packet = 10.168.1.3

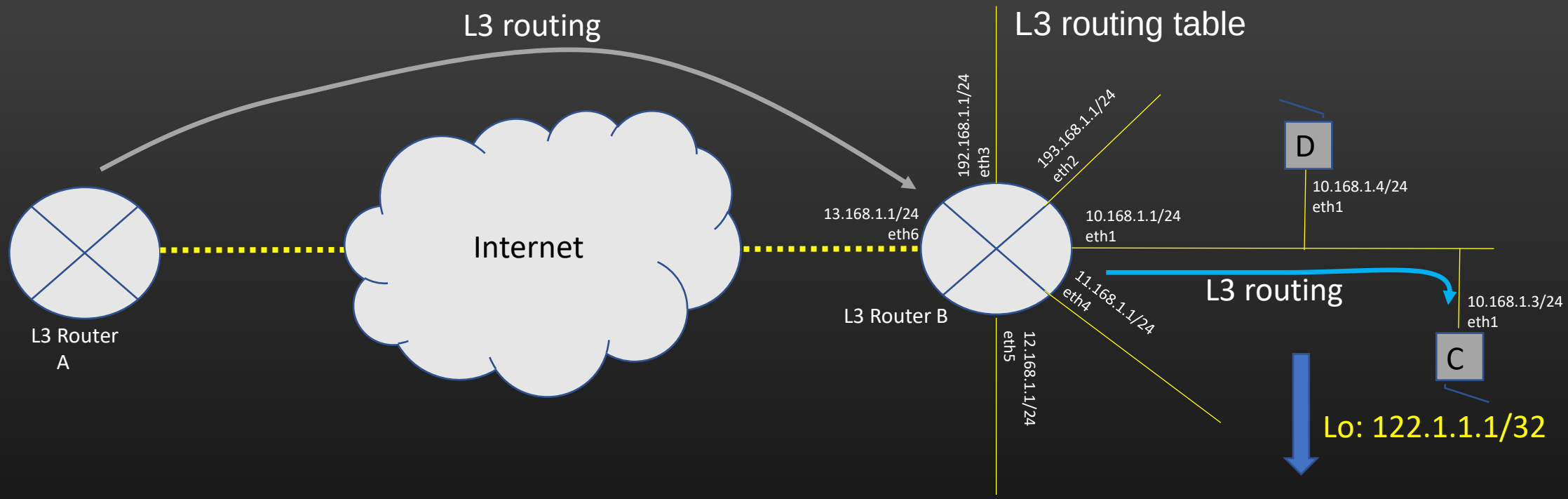
Src mac = MAC(eth1) of B	Dst mac = MAC(eth1) of C	Src ip = X	Dst ip = 10.168.1.3

Dst ip address falls in the local subnet of router B, therefore router B propagates the Packet to the destination C via L2 routing

Lecture VDO 5

Routing – L3 Routing -> L3 routing using loopback ip address

Destination Subnet	Gateway Ip	OIF
10.168.1.0/24	-	-
122.1.1.1/32	10.168.1.3	eth1



Src mac = MAC(eth1) of B	Dst mac = MAC(eth1) of C	Src ip = X	Dst ip = 122.1.1.1

Destination ip in the packet = 122.1.1.1

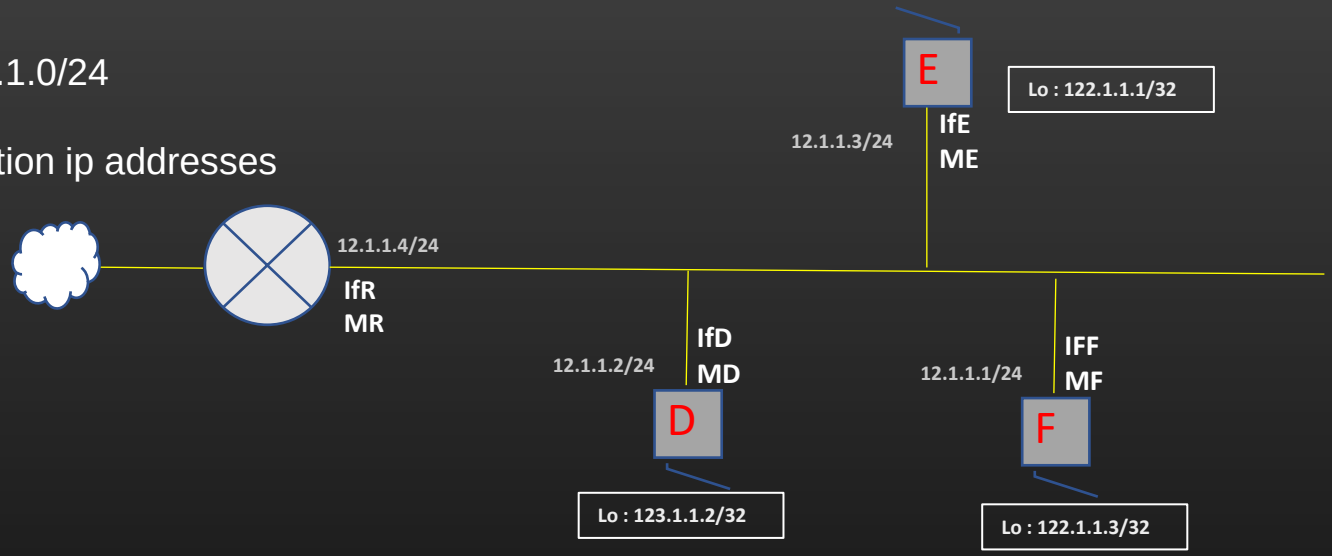
Dst ip address falls in the subnet (122.1.1.1/32), which is remote subnet wrt to router B, therefore router B propagates the Packet further to the destination C via L3 routing

If dest address is loopback address of dst machine, packet never undergoes L2 routing !!

Lecture VDO 5

Routing – L3 Routing -> L3 routing using loopback ip address

- Machines D,E and F are present in same local subnet 12.1.1.0/24
- D,E and F can communicate with each other using Destination ip addresses as 12.1.1.x using pure L2 routing
- Ex : D can send a packet to E using Dst ip address as 12.1.1.3
- But can D send a packet to E using Dst ip = 122.1.1.1 ?
  - Let's see . . .
  - 122.1.1.1 do not falls in any local subnet of D
  - Therefore, D is trying to send a packet to a remote subnet
  - Hence, D can send a packet to E(122.1.1.1) using only L3 routing and for that D should have following L3 routing entry



L3 Entry : 122.1.1.1/32	12.1.1.3	IfD

If dest address is loopback address of dst machine, packet never undergoes L2 routing even if communication machines are physically present in same subnet

Lecture VDO 5

Routing – L3 Routing -> Loopback Interface -> Lo interface creation in linux

- In our example, create a loopback interfaces lo1 on All VMs and
  assign ip address to it

  cmd line : *sudo ifconfig lo:1 122.1.1.1 netmask 255.255.255.255 up*

  To add permanently, add the following in a file /etc/network/interfaces and reboot

```
auto lo:1
iface lo:1 inet static
    address 122.1.1.1
    network 122.1.1.1
    netmask 255.255.255.255
```

- Verify using *ifconfig* command

Layer 3 Routing table Calculation

Lecture VDO 6

Routing – ping & tcpdump

Ping
&
tcpdump

Lecture VDO 6

Routing – ping

How does ping works. . .

- *Ping* is a command used to test the reachability of certain ip address in the network from current machine
- Synopsis : *ping <ip address>*
- *ping* is a just a front end command. The backend protocol for *ping* command is *ICMP protocol – Internet control message protocol.*
- *ICMP* is very simple protocol which operates with the help of two ICMP sub-messages :
  - ICMP echo request messages
  - ICMP echo reply messages
- *ICMP echo request msg* is a kind of msg saying "*Hello you , how are you ? Are you john ?*" (replace john with ip address being pinged)
- *ICMP echo reply* is same as if John has replied back – "*Yes I am good.*"
- If *protocol* field value in IP Hdr is 1 , then it means Network layer protocol is shipping *ICMP msgs as an application msg (ICMP echo req Or ICMP echo reply)*

Lecture VDO 6

Routing – ping

How does ping works. . .

- ping <ip address> is used to test the reachability of certain ip address in a network.
- ping command sends special msg called ICMP echo request to the machine hosting ip address being pinged.
- If Destination machine hosting pinged ip-address receives icmp echo request, the Destination machine replies back with ICMP echo reply
- *ICMP echo request msg* is a kind of msg saying "*Hello you , how are you ? Are you john ?*" (replace john with ip address being pinged)
- *ICMP echo reply* is same as if John has replied back – "*Yes I am good.*"
- Usual L3 routing is used to route the ICMP echo req and reply msgs in the network.

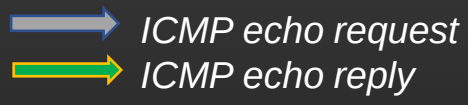
"Destination Hosts an ip address" means that ip address is an address of one of Its local interfaces (incl lo)

ICMP is an application layer protocol called – Internel control message protocol.

Lecture VDO 6

Routing – ping

How does ping works. . .



PING Algorithm (executed on each router receiving ping echo request)

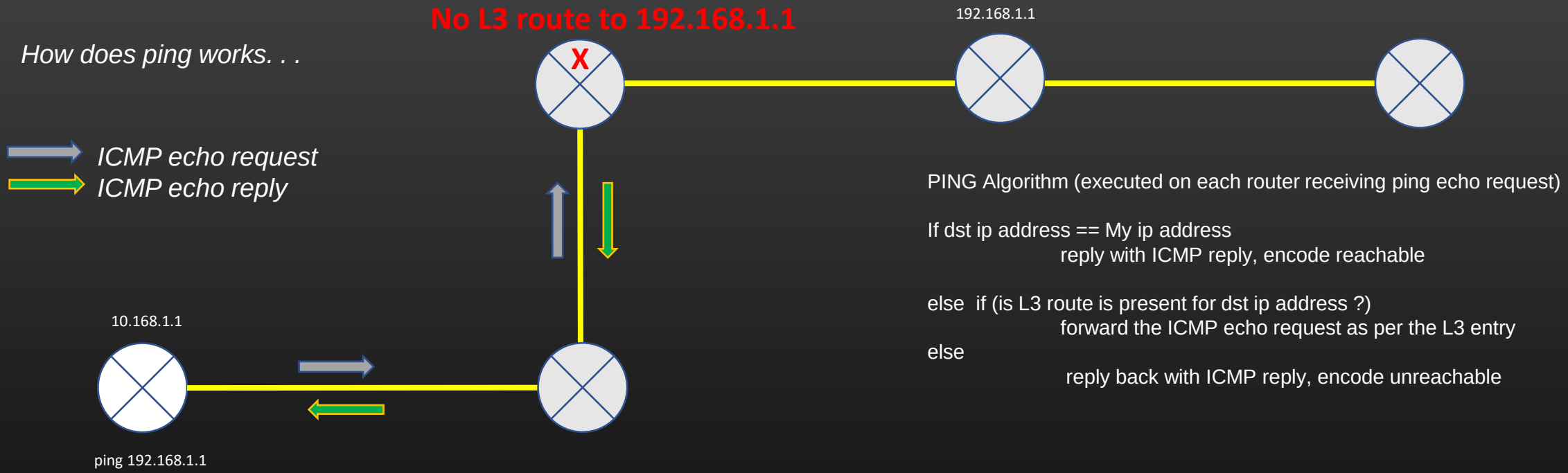
If dst ip address == My ip address
reply with ICMP reply, encode reachable

else if (is L3 route is present for dst ip address ?)
forward the ICMP echo request as per the L3 entry
else
reply back with ICMP reply, encode unreachable

Source Router Concludes that ip address 192.168.1.1 is reachable

Lecture VDO 6

Routing – ping



PING Algorithm (executed on each router receiving ping echo request)

If dst ip address == My ip address
reply with ICMP reply, encode reachable

else if (is L3 route is present for dst ip address ?)
forward the ICMP echo request as per the L3 entry
else
reply back with ICMP reply, encode unreachable

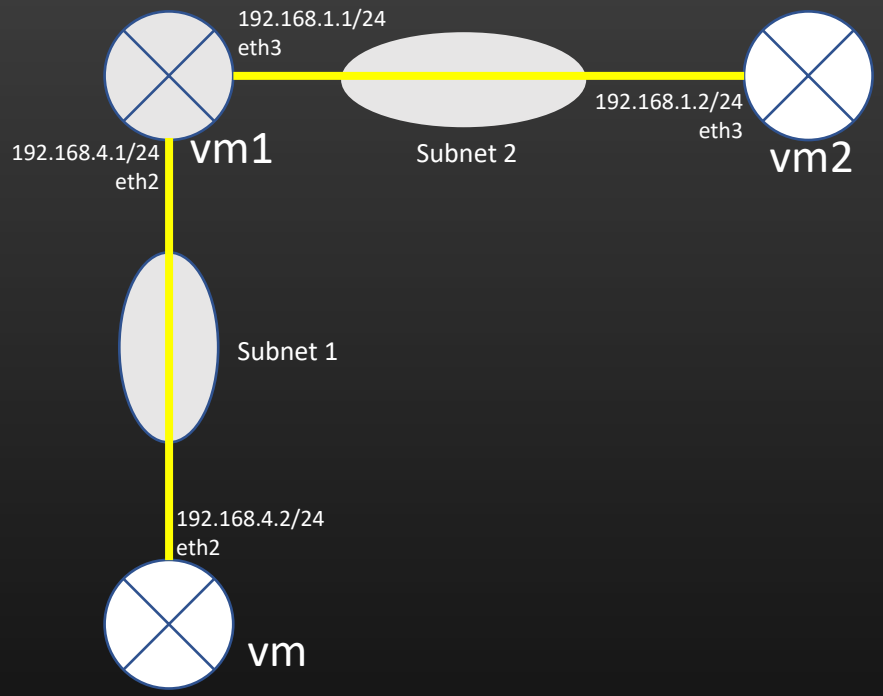
Source Router Concludes that ip address 192.168.1.1 is NOT reachable

Lecture VDO 6

Routing – TCP Dump

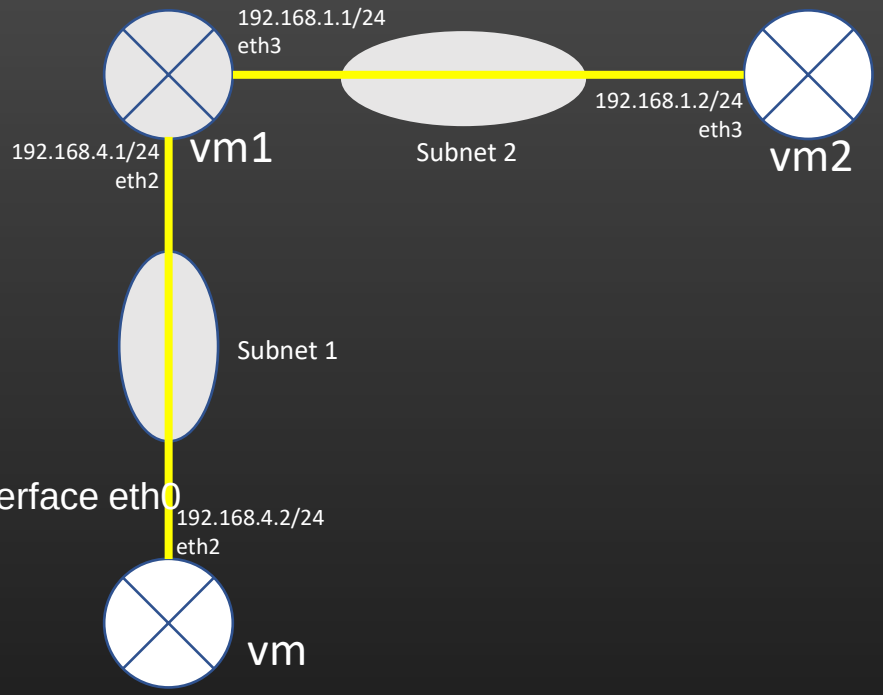
linux TCP dump utility Demonstration . . .

- tcpdump command shows what packets are going out or coming in on an interface on which tcpdump is running
- tcpdump can show you all the fields of all standard headers contained in the packet (Ethernet hdr, IP hdr, ARP hdr, etc)
- tcpdump works at data link layer, it captures the packet as soon as packet is placed or received on an interface
- Great Utility for debugging
- Let us ping 192.168.1.2 from VM, and run *tcpdump* command on interface eth2 of VM1
- Alternative to tcpdump is software : Wireshark which is GUI based unlike tcpdump which is CLI based (Good to know about it, Explore yourself)



Lecture VDO 6

Routing – TCP Dump



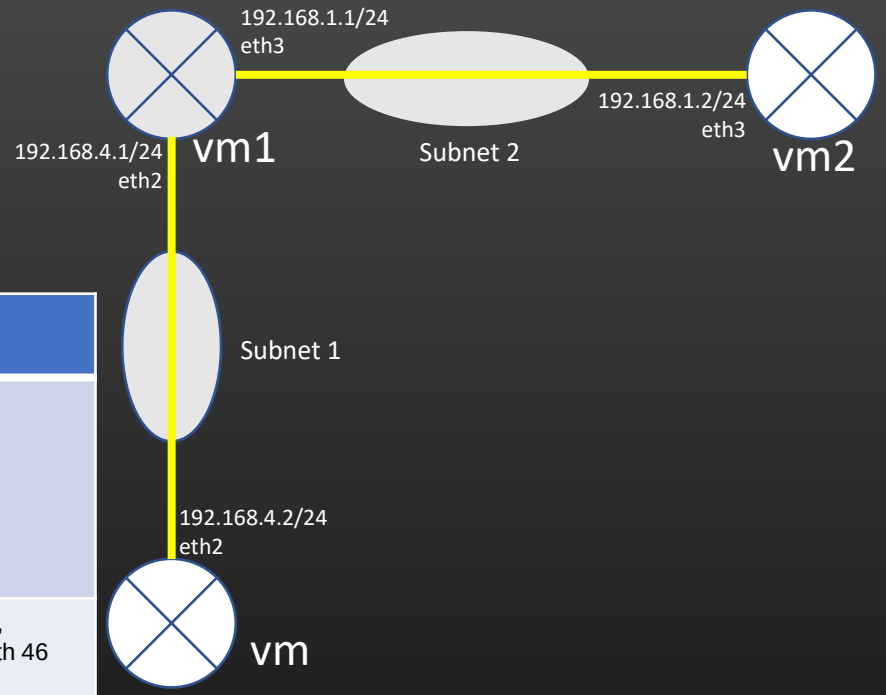
- tcpdump captures all kind of packet, not necessarily tcp packets.
  Don't get mislead by name of the command.

- Ex :

  tcpdump -c 1 -vv -i eth0
  This cmd will capture 1 packet coming in Or going out of interface eth0

Lecture VDO 6

Routing – ping and tcpdump Demonstration

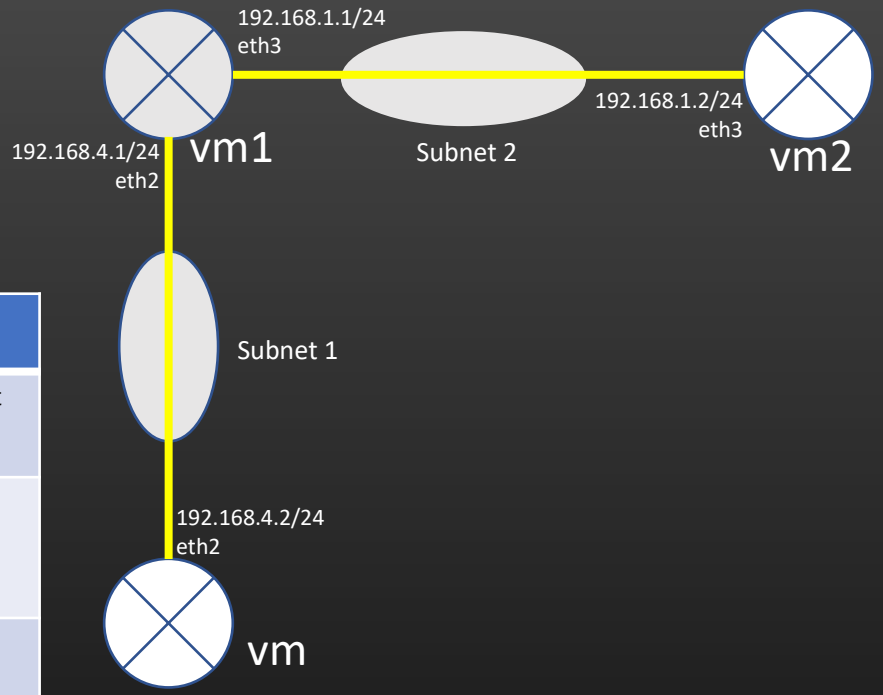


ping 192.168.1.2 from VM

Events	VM1 : tcpdump -c 1 -vv -i eth2
VM wants to send data (ICMP) to 192.168.1.2. It decides that it has to use L3 routing to send data to this IP address (because 192.168.1.2 do not fall in any local subnet of VM). It looks up its routing table and know nexthop information (OIF= eth2, GW : 192.168.4.1) VM checks its ARP for GW ip, but do not find MAC entry VM sends ARP request msg to know mac for GW ip(192.168.4.1) on interface eth2	
VM1 receives ARP req msg on eth2	02:37:48.363983 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.4.1 tell 192.168.4.2, length 46
VM1 sends out ARP reply out of interface eth2	02:37:48.364017 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.4.1 is-at 08:00:27:46:dd:8d (oui Unknown), length 28
VM receives ARP reply. VM sends data out of intf eth2 with Dest = 192.168.1.2	
VM1 receives the data on interface eth2	192.168.4.2 > 192.168.1.2: ICMP echo request, id 4308, seq 1, length 64 02:37:48.364888 IP (tos 0x0, ttl 63, id 7311, offset 0, flags [none], proto ICMP (1), length 84)
Data reaches to VM2 eventually. VM2 replies back with ICMP reply msg destined to 192.168.4.2. VM1 receives ICMP reply. VM1 sees destination IP address and checks that address = 192.168.4.2 falls in its local subnet 192.168.4.0/24. So, VM1 now uses L2 routing to further prorogate the packet to its destination (VM).	

Lecture VDO 6

Routing – ping and tcpdump Demonstration

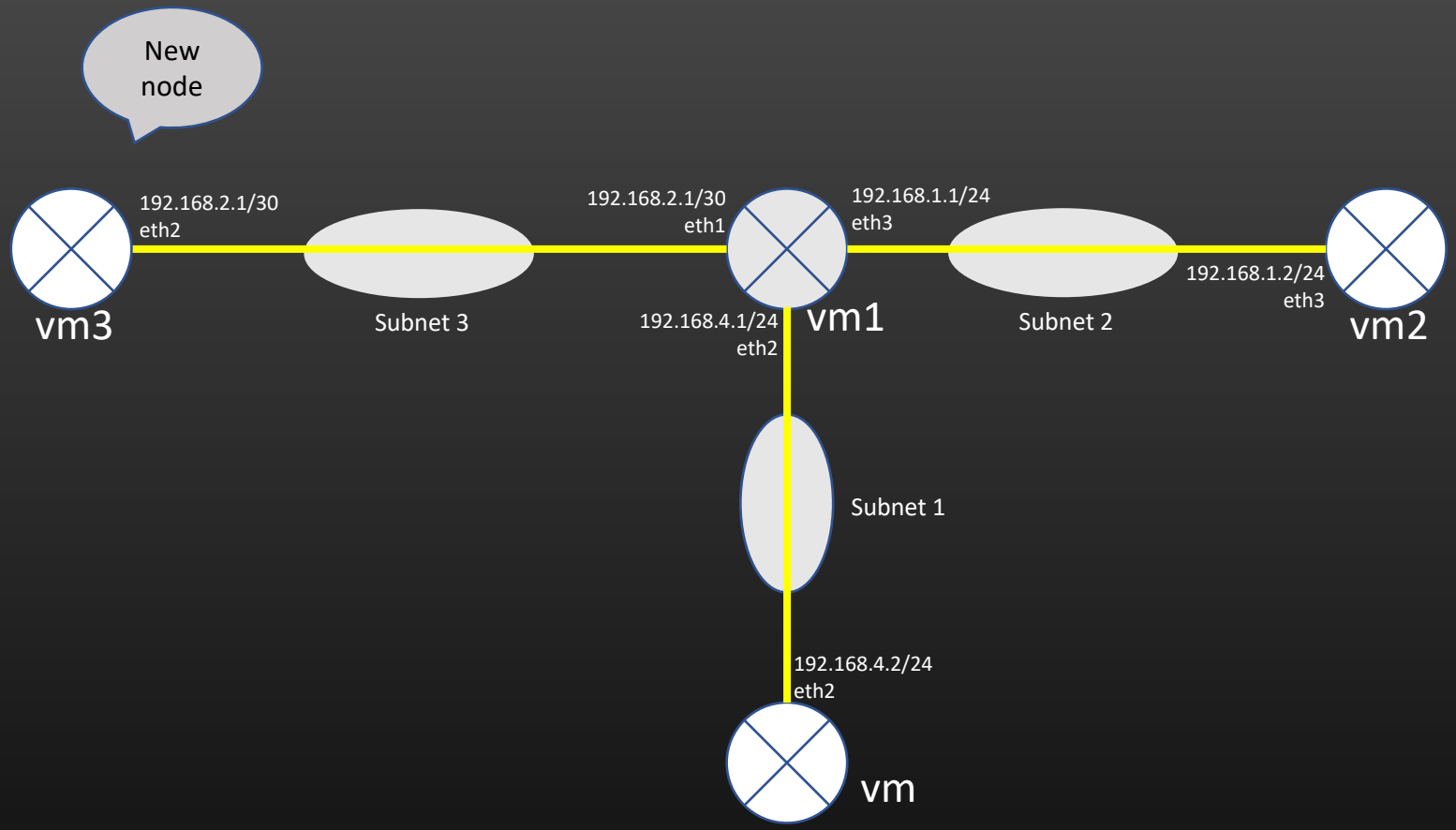


ping 192.168.1.2 from VM

Events	VM1 : tcpdump -c 1 -vv -i eth2
VM1 checks its ARP cache to find mac for 192.168.4.2. It do not find. VM1 sends ARP req msg on interface which is operating in subnet 192.168.4.0/24 (in which 192.168.4.2 falls) i.e. eth2.	02:37:53.374691 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.4.2 tell 192.168.4.1, length 28
VM1 receives ARP reply from VM	02:37:53.375542 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.4.2 is-at 08:00:27:3e:97:62 (oui Unknown), length 46
VM1 forwards out ICMP reply which it had received from VM2 out of interface eth2.	02:37:48.364888 IP (tos 0x0, ttl 63, id 7311, offset 0, flags [none], proto ICMP (1), length 84) 192.168.1.2 > 192.168.4.2: ICMP echo reply, id 4308, seq 1, length 64
VM receives ICMP reply, ping journey completes	

Lecture VDO 6

Routing – Assignment

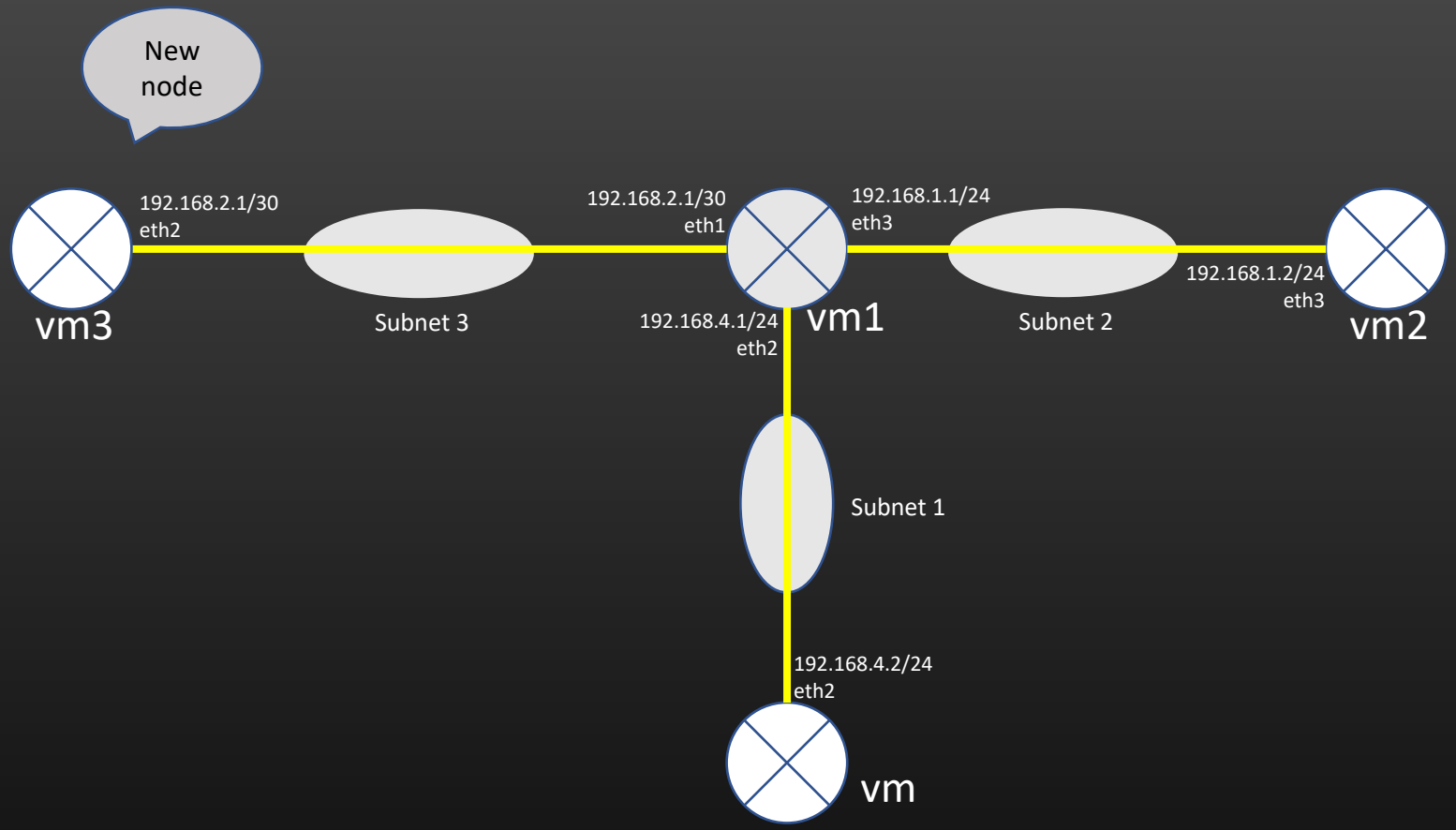


In our 3 node topology, add 4th node VM3 as shown in the diagram.
Install the routes in all required VMs so that from any given VM you should be able To ping any other VM in the topology

Q1. Analyze ARP and routing table entries in all VMs and understand their meaning

Lecture VDO 6

Routing – Assignment



Q2. Analyze the incoming and outgoing packets using linux TCP dump utility

Q3. Assign lo addresses to all VMs and install the routes in routing table to reach lo addresses Of any VM from any VM

Lecture VDO 6

Routing – L3 Routing

Points to Note :

- Src Mac and Dst Mac fields in Ethernet Hdr changes hop by hop
- Src ip and Dst IP fields in IP hdr never changes
- L2 routing is used to take the packet from current router/host to immediate next router
- L3 routing takes care of packet delivery from Source router to destination router

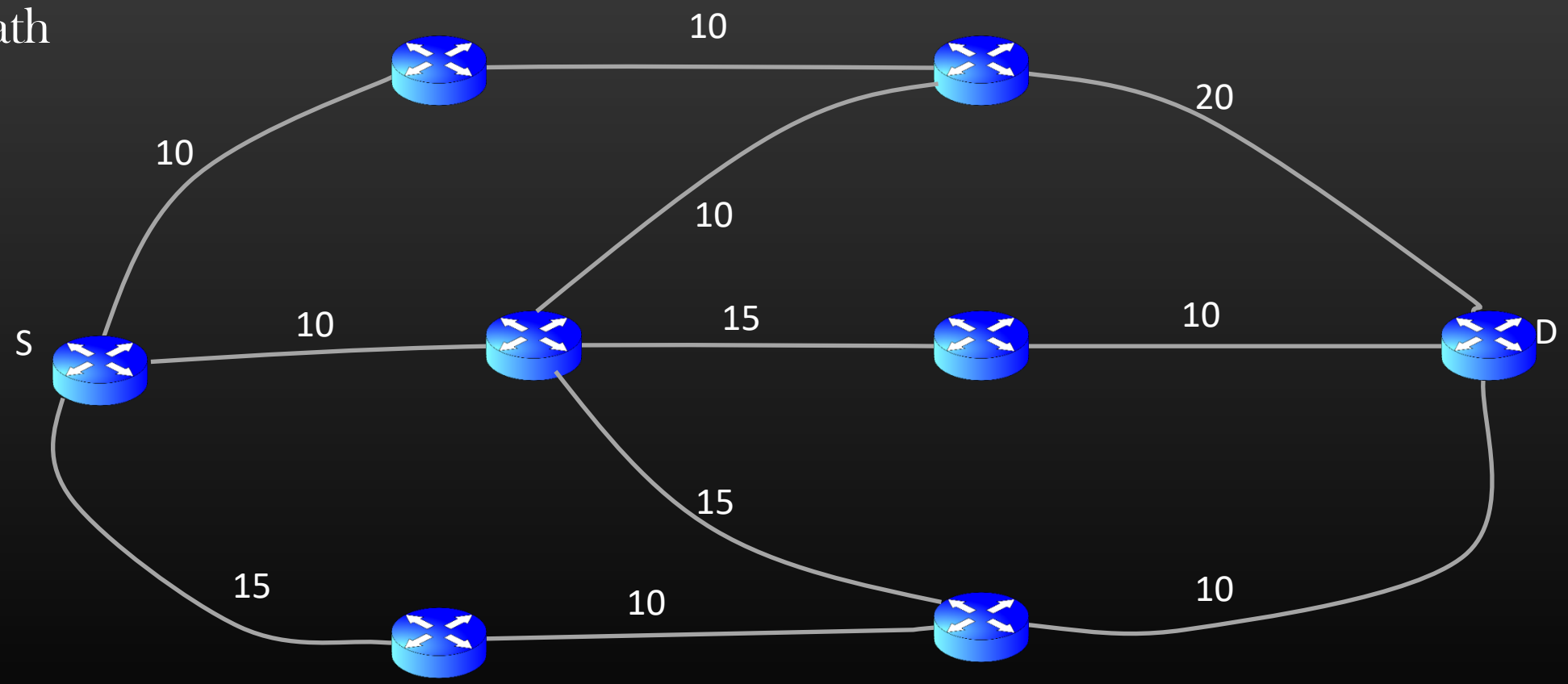
L3

Routing table Construction

Construction of L3 Routing Table

How L3 Devices automatically know where to forward the traffic so that :

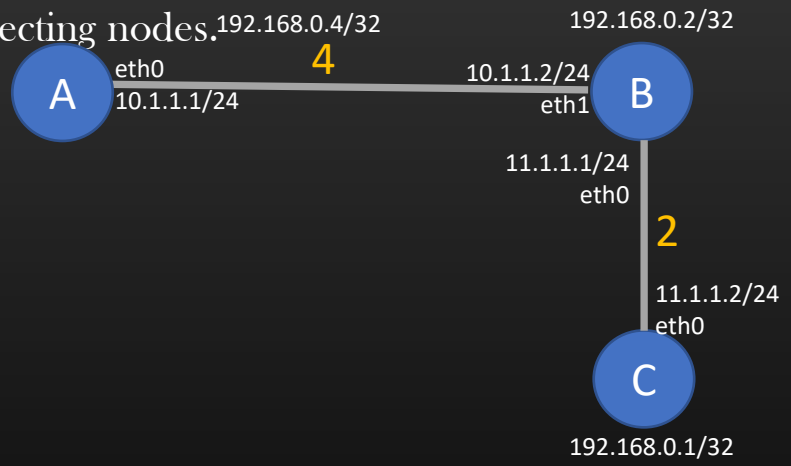
- Traffic reaches to its intended Destination via Cooperation of L3 devices
- Via Shortest Path
- Loop Free
- ECMP



Goals

Problem Statement : Given a Network Graph, Compute the routing table of each node in the Graph

- Input : Network Graph
  - A Graph where nodes represent the L3 routers, and edges represent the Wires connecting nodes.
  - All Edges are P2P links, and has cost attached to it
  - Edge ends have IP addresses configured, and have interface name
  - Nodes have Name, and Loopback Address.
- Output : Routing Table
  - Each node in the Graph must compute Routing table
  - Routing table means – Every node must know how to reach (nexthop) a given remote subnet in the graph (loopback addr of other L3 devices)
  - Node(i) must be reachable to node(j) through shortest path in the graph, for all i,j



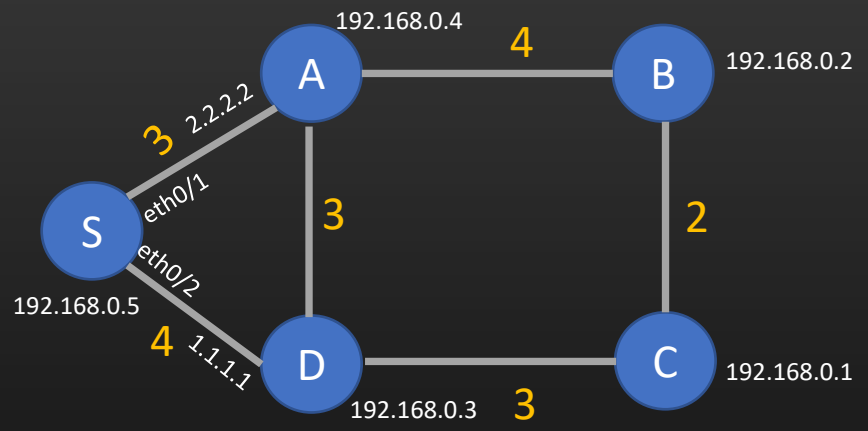
Construction of L3 Routing Table

L3 route entry format :

<Dest> <Gateway-ip> <OIF> <cost> <nexthop-node>

Routing Table of node S

Dest IP	Oif	gateway	cost	Nexthop node
192.168.0.1/32	eth0/2	1.1.1.1	7	D
192.168.0.2/32	eth0/1	2.2.2.2	7	A
192.168.0.3/32	eth0/2	1.1.1.1	4	D
192.168.0.4/32	eth0/1	2.2.2.2	3	A
191.268.0.5/32	Nil	Nil	0	Nil



- Least Cost to reach any other node
- Loop free
- Nodes recomputes RT if topology changes

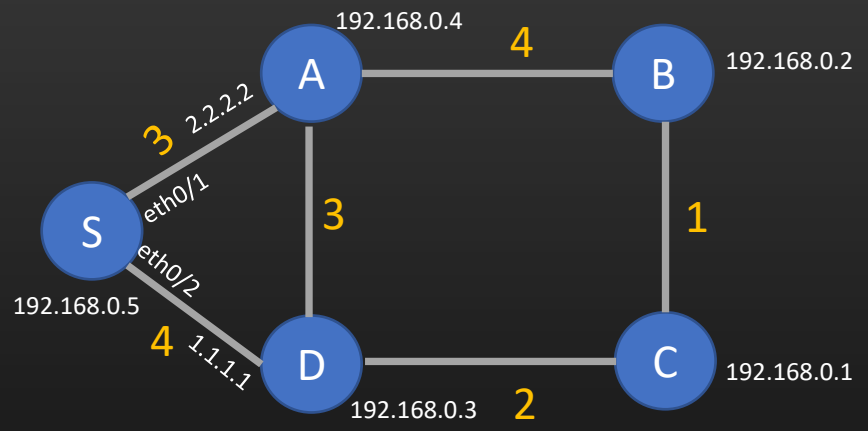
Construction of L3 Routing Table

L3 route entry format :

<Dest> <Gateway-ip> <OIF> <cost> <nexthop-node>

Routing Table of node S with ECMP

Dest IP	Oif	gateway	cost	Nexthop node
192.168.0.1/32	eth0/2	1.1.1.1	6	D
192.168.0.2/32	eth0/1 eth0/2	2.2.2.2 1.1.1.1	7	A D
192.168.0.3/32	eth0/2	1.1.1.1	4	D
192.168.0.4/32	eth0/1	2.2.2.2	3	A
191.268.0.5/32	Nil	Nil	0	Nil



- Least Cost to reach any other node
- Loop free
- Nodes recomputes RT if topology changes

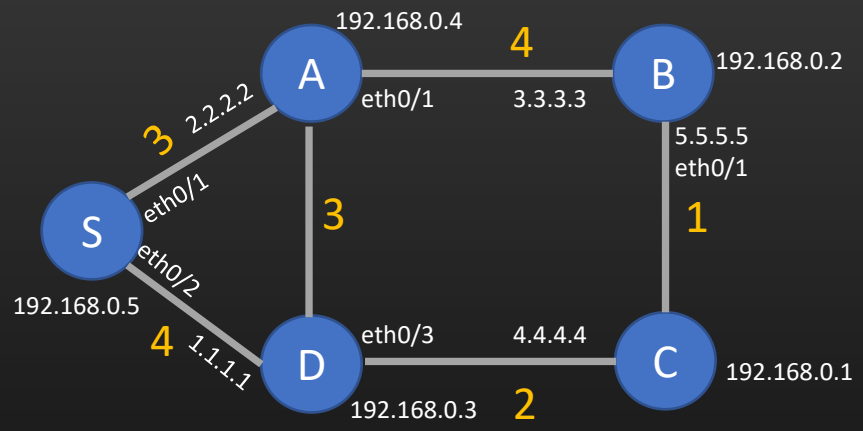
Construction of L3 Routing Table

L3 route entry format :

<Dest> <Gateway-ip> <OIF> <cost> <nexthop-node>

Routing Table Entry for 192.168.0.2/32 in all nodes of Network

Node	Dest IP	Oif	gateway	cost	Nexthop node
S	192.168.0.2/32	eth0/1 eth0/2	2.2.2.2 1.1.1.1	7	A D
A	192.168.0.2/32	eth0/1	3.3.3.3	4	B
D	192.168.0.2/32	eth0/3	4.4.4.4	3	C
C	192.168.0.2/32	eth0/1	5.5.5.5	1	B
B	191.268.0.2/32	Nil	Nil	0	Nil



- Least Cost to reach any other node
- Loop free
- Nodes recomputes RT if topology changes

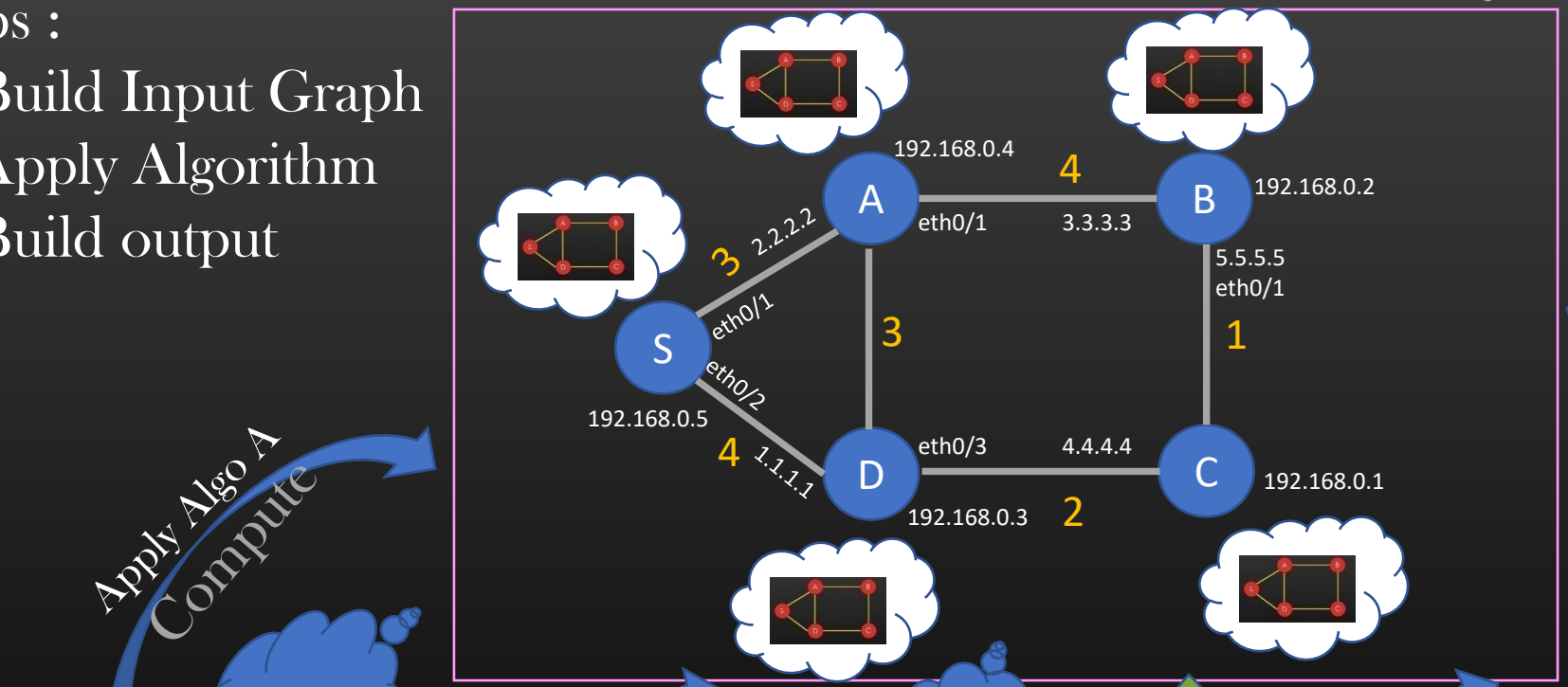
Construction of L3 Routing Table

Each L3 Device Must have its own Graph Data structure which represents entire Network Graph

Physical Topology

Steps :

1. Build Input Graph
2. Apply Algorithm
3. Build output



Apply Algo A Compute

Output RT

Apply Algo A Compute

Output RT

Apply Algo A Compute

Output RT

L3

Routing table Construction

Take Away from this Section :

- You would learn
  - How Physical Topology is assimilated as Graph Data structure in local Memory of Each L3 Routing Device
  - The algorithms used to construct the routing table
    (Used by Interior Gateway Routing Protocols such as OSPF, ISIS)
  - Final Routing Table Computation

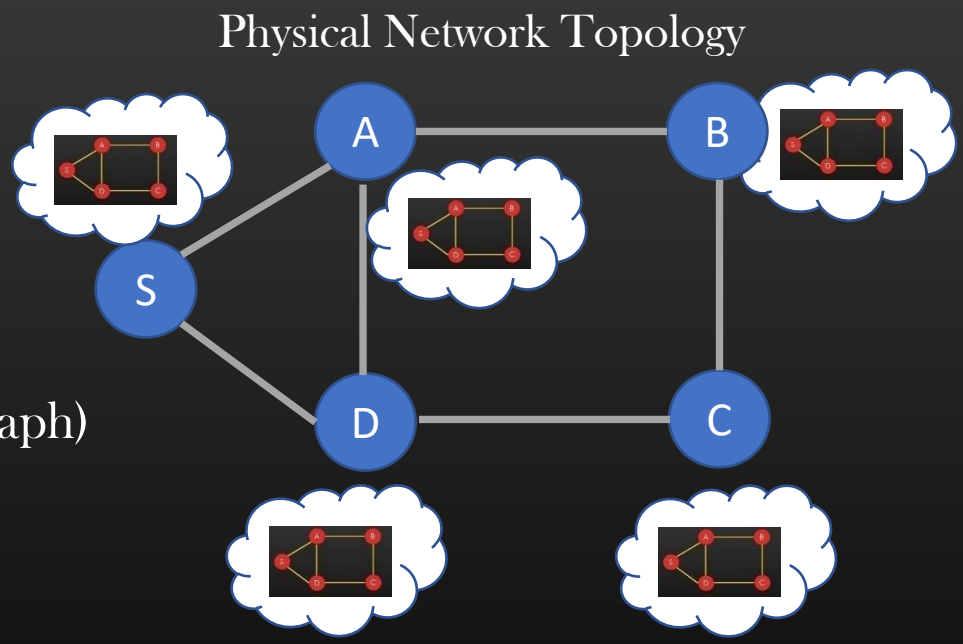
Construction of L3 Routing Table

Let us Break down the problem into phases

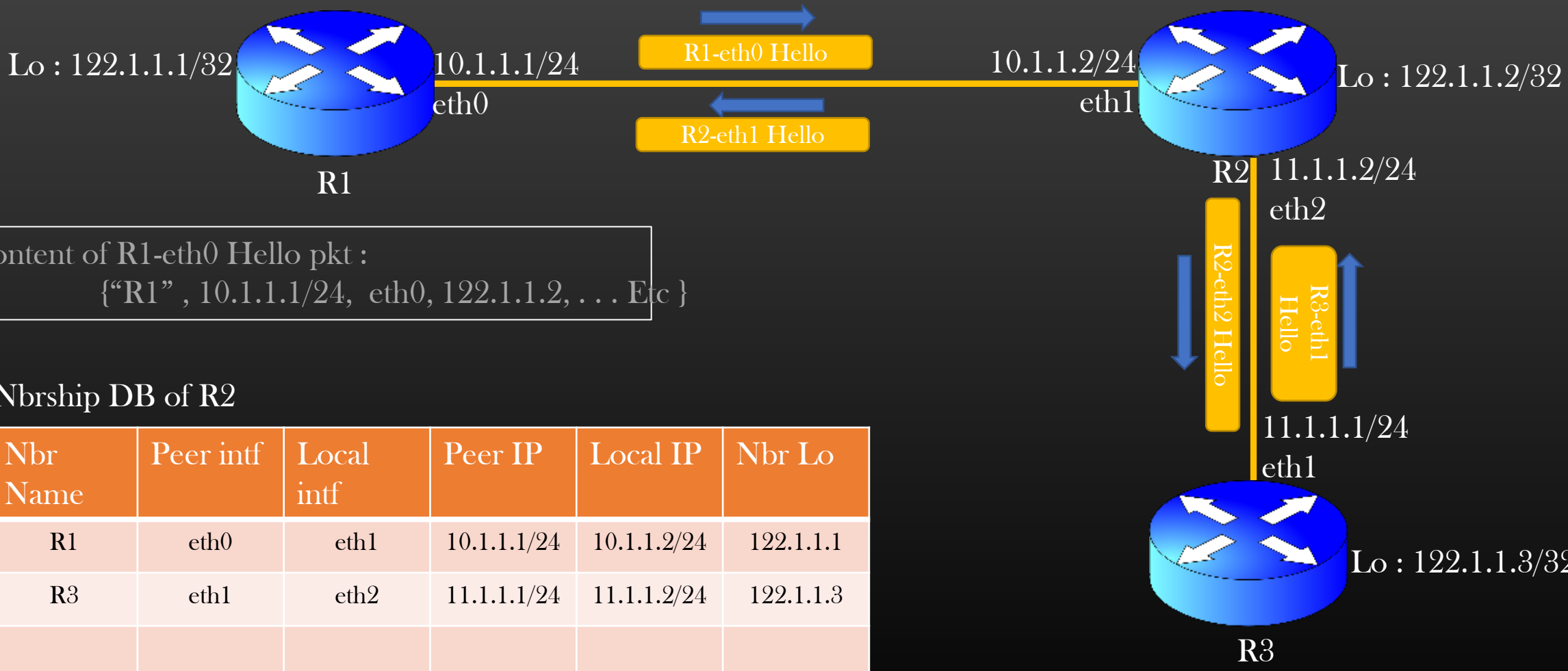
- Phase 1 : Build the input graph
  - How Networking Routing Devices Build the Input Graph Data Structure
- Phase 2 : Run the shortest path algorithm (Dijkstra Algorithm) on the Input Graph
  - You must understand the Dijkstra algorithm, refer to standard book
- Phase 3 : Routing Table Calculation
  - Convert the output of phase 2 into actual routing table

- Phase 1 : Build the input graph
  - Step 1 : Exchange of Hello Pkts
  - Step 2 : Build Nbrship Database
    - Every Routing Device knows its nbrs
  - Step 3 : Flood Nbrship Database
  - Step 4 : Build Cumulative Nbrship Database (= Network Graph)

Let us understand these steps through Example ..



Step 1 & 2 : Exchange Hello Pkts & Build Nbrship Database



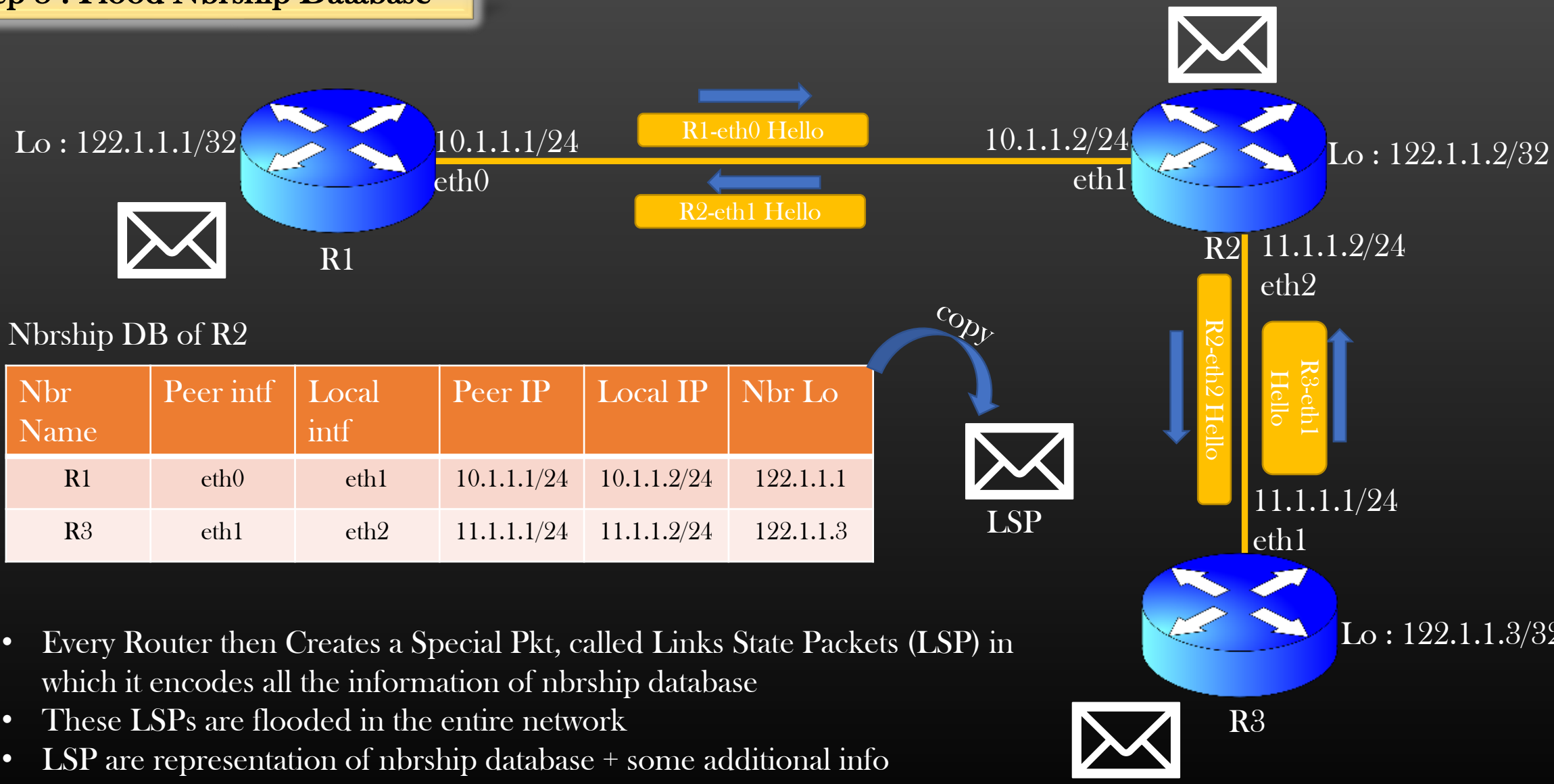
Content of R1-eth0 Hello pkt :
{"R1" , 10.1.1.1/24, eth0, 122.1.1.2, . . . Etc }

Nbrship DB of R2

Nbr Name	Peer intf	Local intf	Peer IP	Local IP	Nbr Lo
R1	eth0	eth1	10.1.1.1/24	10.1.1.2/24	122.1.1.1
R3	eth1	eth2	11.1.1.1/24	11.1.1.2/24	122.1.1.3

Similarly, Every L3 Routing Device build its own nbrship Database by Exchanging Hello Pkts

Step 3 : Flood Nbrship Database

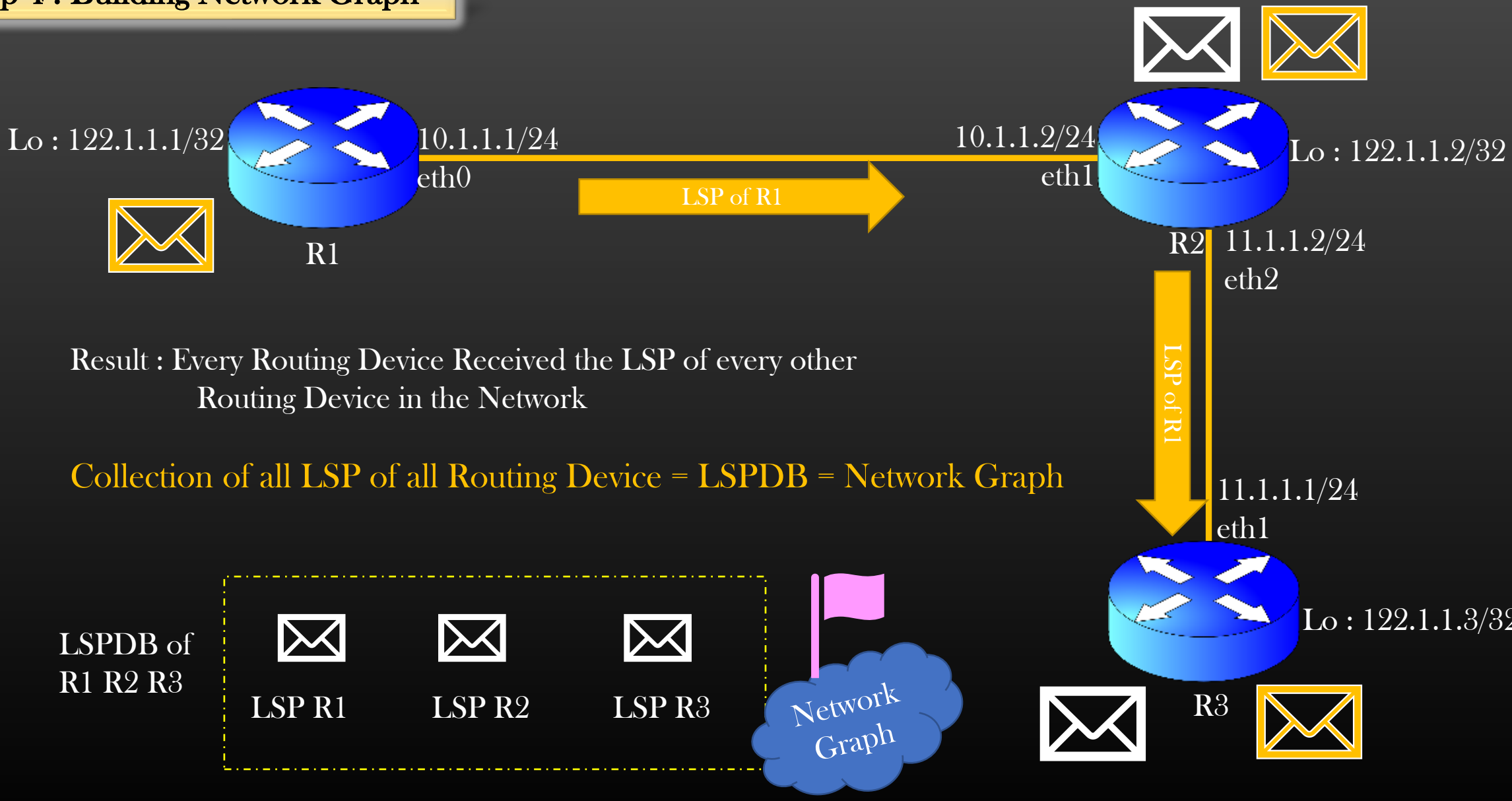


Nbrship DB of R2

Nbr Name	Peer intf	Local intf	Peer IP	Local IP	Nbr Lo
R1	eth0	eth1	10.1.1.1/24	10.1.1.2/24	122.1.1.1
R3	eth1	eth2	11.1.1.1/24	11.1.1.2/24	122.1.1.3

- Every Router then Creates a Special Pkt, called Links State Packets (LSP) in which it encodes all the information of nbrship database
- These LSPs are flooded in the entire network
- LSP are representation of nbrship database + some additional info
- Result ?

Step 4 : Building Network Graph



Result : Every Routing Device Received the LSP of every other Routing Device in the Network

Collection of all LSP of all Routing Device = LSPDB = Network Graph

Step 4 : Building Network Graph

S recvs LSP of S

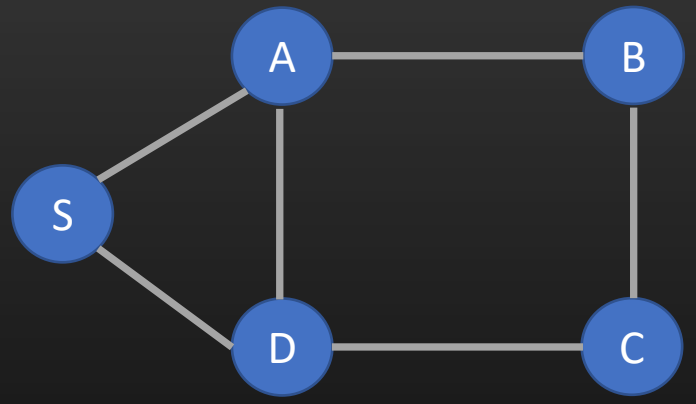
S recvs LSP of A

S recvs LSP of D

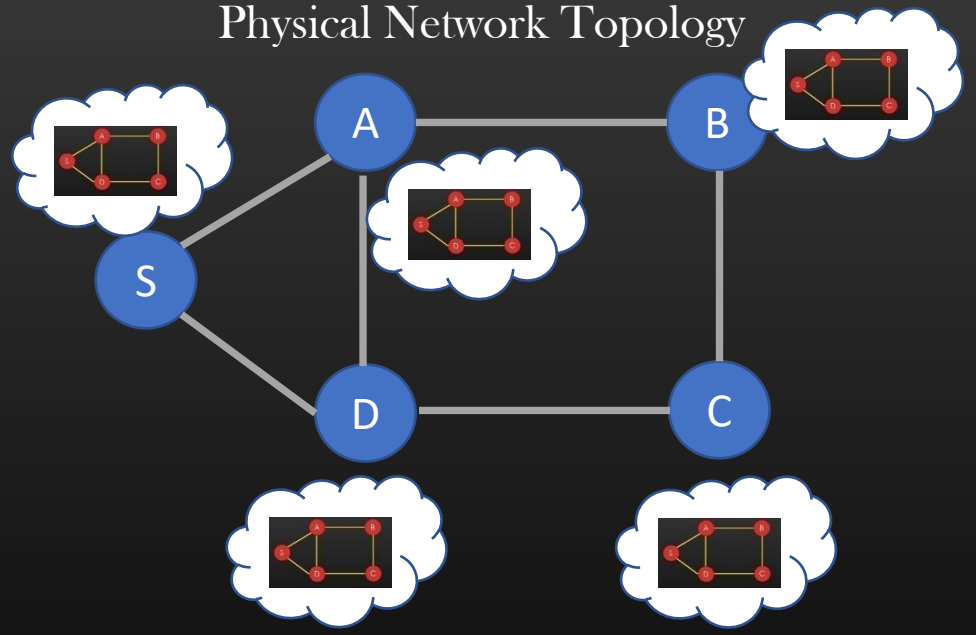
S recvs LSP of B

S recvs LSP of C

S's Internal Graph Topology Data Structure

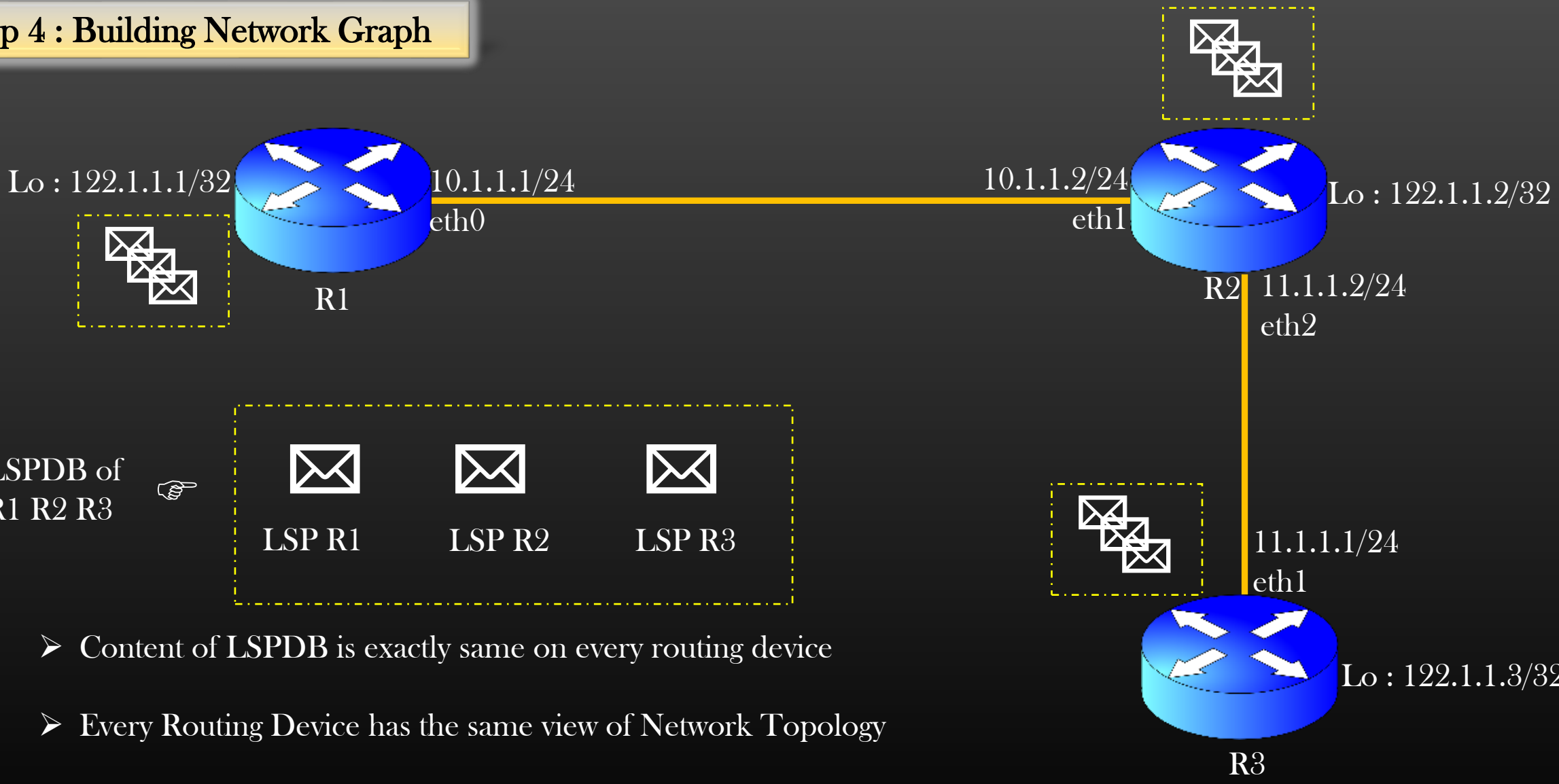


Physical Network Topology



- Because every node builds up same LSPDB, Every node has the same view of physical Nw Topology in the form of internal Graph data structure

Step 4 : Building Network Graph



- Content of LSPDB is exactly same on every routing device
- Every Routing Device has the same view of Network Topology
- Every Routing Device then Runs several Algorithms on Local Network Graph to compute results such as its local Routing Table

Construction of L3 Routing Table

Let us Break down the problem into phases

- Phase 1 : Build the input graph
  - How Networking Routing Devices Build the Input Graph Data Structure
- Phase 2 : Run the shortest path algorithm (Dijkstra Algorithm) on the Input Graph
  - You must understand the Dijkstra algorithm, refer to standard book
- Phase 3 : Routing Table Calculation
  - Convert the output of phase 2 into actual routing table

- Phase 2 : Run SPF Algorithm
  - Each Routing Device Runs SPF (Shortest Path first , Dijkstra) Algorithm on its Local LSDB
  - Result : Shortest path to reach each routing device in the Network Graph is computed
  - Revise Dijkstra Algorithm before Proceeding forward

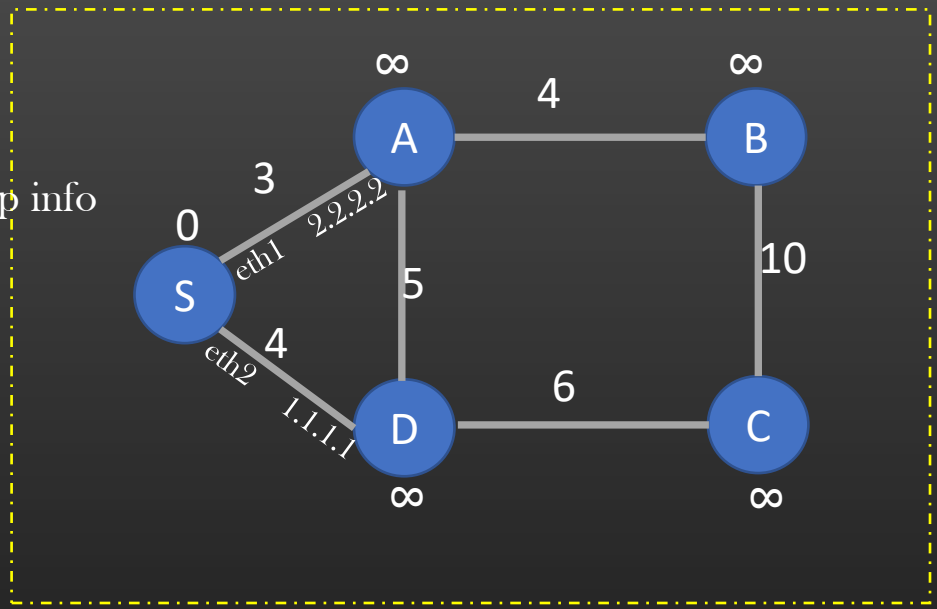
Construction of L3 Routing Table -> Phase 2 -> SPF Algorithm

Data Structures :

Each node has an array which store pointer to nexthop :
nexthop{node} << represent an array which stores nexthop info
nexthop{node}.flush() << flush the entire array
nexthop{node}.append(nexthop) << add a new nexthop into array

Each node as a cost which represents the shortest distance of a node
From spf_root of the graph
cost(node)
cost(node1, node2) << represents a link cost connecting node1 & node2

Spf_root node has a table which can store SPF results
S.record_result(nxthop{node}.list()); << copy the list of nexthops of given node into



LSDB of S
(This DS is present in local memory Of Device S)

Spf_root S Result

Dest Node	Oif	Gateway	cost	Nexthop
A				
D				
B				
C				

Construction of L3 Routing Table -> Phase 2 -> SPF Algorithm

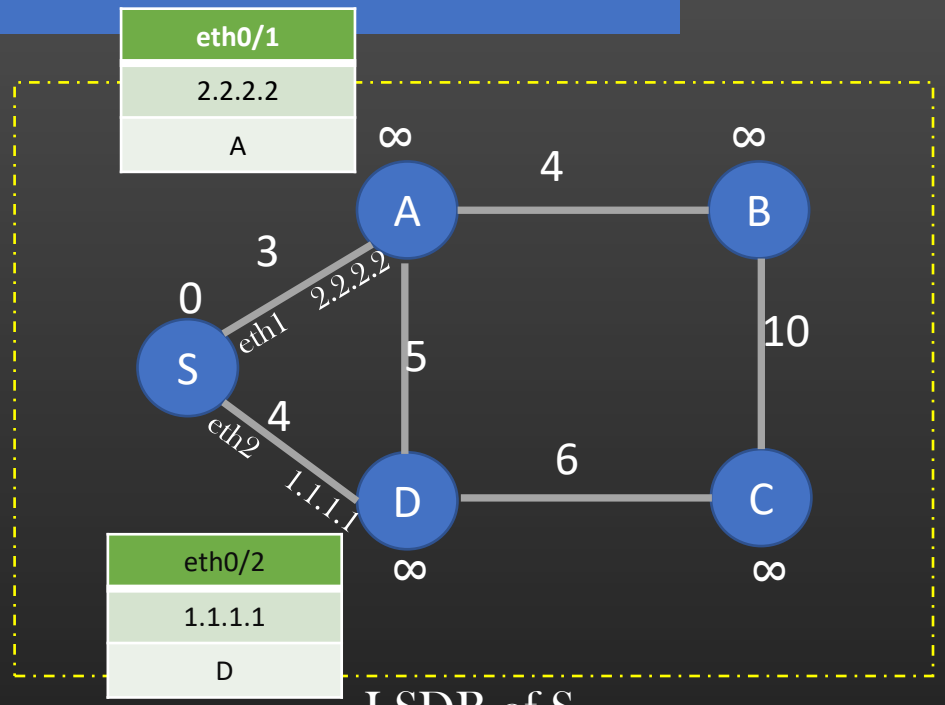
```
init_spf_algo(S, G)

spf_root = S
cost(spf_root) = 0
flush_spf_result(spf_root)
For each node N of Graph G, where N != spf_root :
        cost(N) = ∞
        N.fush_spf_result(N)
for each nbr of spf_root

        if(cost(spf_root) + cost(spf_root, nbr) < cost(nbr))
                cost(nbr) = cost(spf_root) + cost(spf_root, nbr)
                nxthop = build_nxt_hop(spf_root, nbr)  // ex : eth1, 2.2.2.2, A
                nxthop{nbr}.flush()
                nxthop{nbr}.append(nxthop)

        else if(cost(spf_root) + cost(spf_root, nbr) == cost(nbr)) /*ECMP case*/
                nxthop = build_nxt_hop(spf_root, nbr)
                nxthop{nbr}.append(nxthop)

Initialize Priority Q to Empty
Enque(Q, spf_root)
```

1

2.1

2.2

3



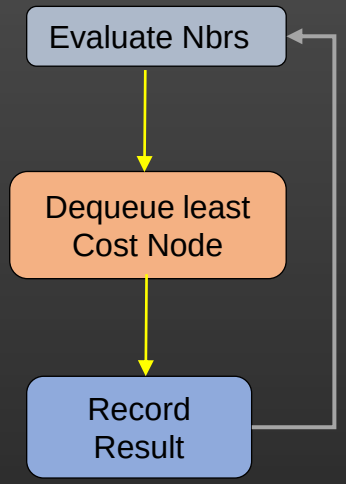
LSDB of S

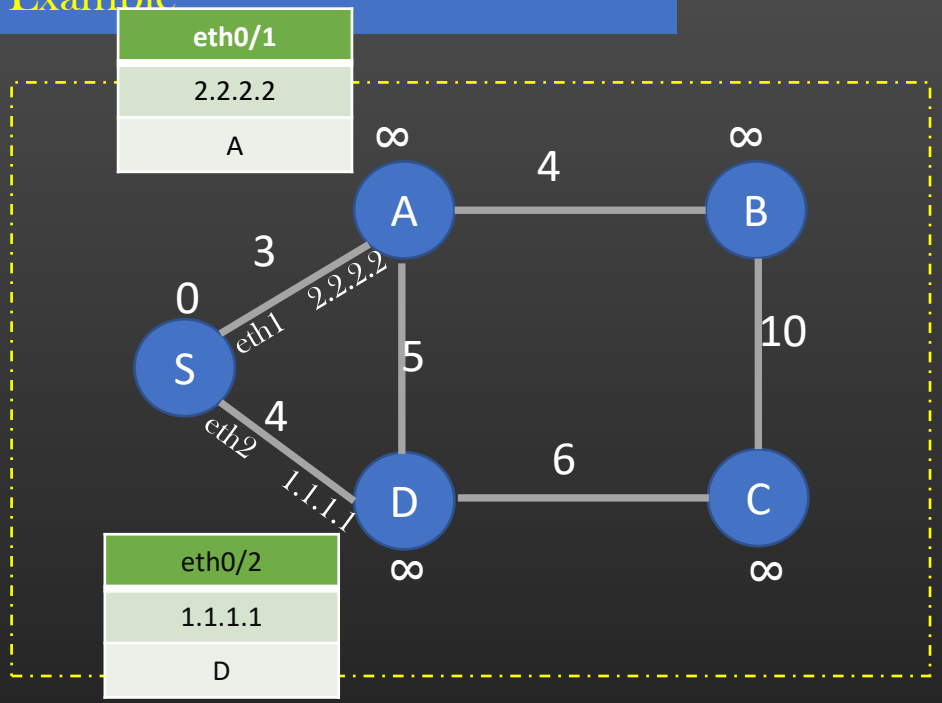
(This DS is present in local memory Of Device S)

Spf_root S Result

Dest Node	Oif	Gateway	cost	Nexthop
A				
D				
B				
C				

Construction of L3 Routing Table -> Phase 2 -> SPF Algorithm -> Dry Run Example

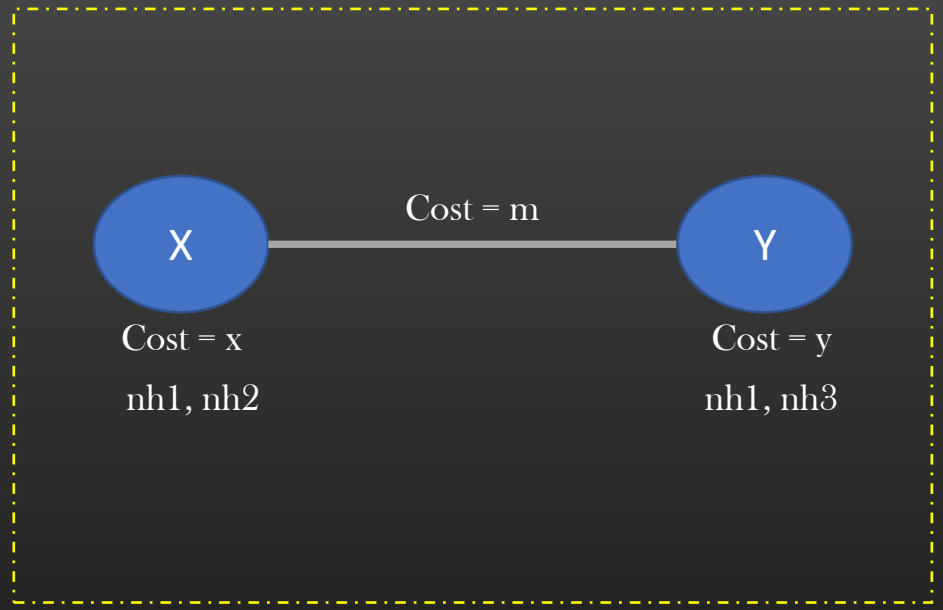




spf_root Result

Dest Node	Oif	Gateway	cost	Nexthop
A				
D				
B				
C				

Algorithm Rules When a new node is explored



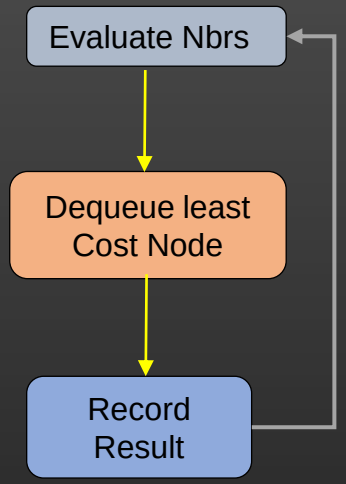
- If node Y is explored from predecessor node X with a better metric
  i.e. x + m < y
  - Remove nexthops of Y
  - Copy nexthops of X into Y
  - Remove Y from PQ if present, and add it back to PQ

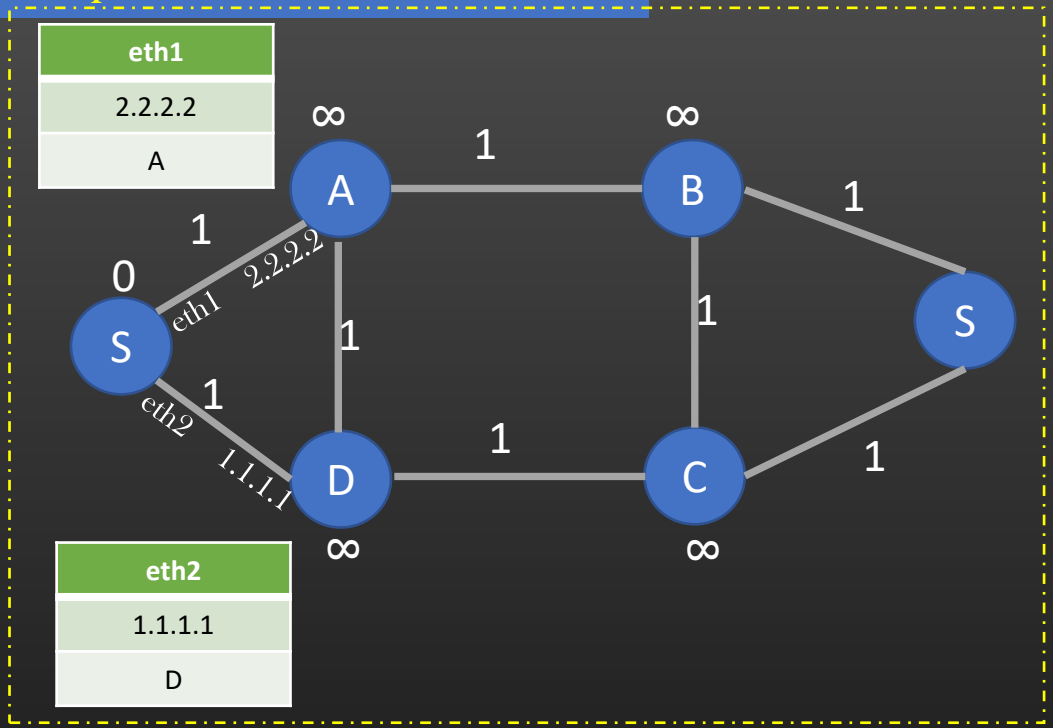
- If node Y is explored from predecessor node X with a same metric
  i.e. x + m = y (ECMP case)
  - Keep existing nexthops of Y
  - Copy nexthops of X into Y
  - Remove Duplicates in next-hop list of Y, if any

- If node Y is explored from predecessor node X with higher metric
  i.e. x + m > y
  - No Action

- The Algorithm Stops when PQ is empty

Construction of L3 Routing Table -> Phase 2 -> SPF Algorithm -> Dry Run Example





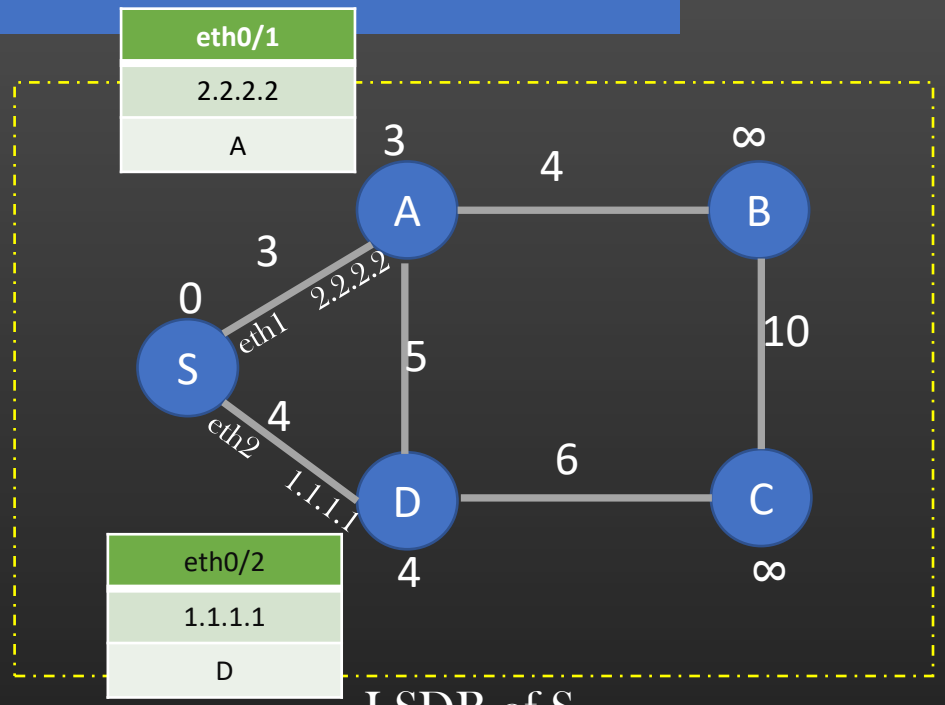
spf_root Result

Dest Node	Oif	Gateway	cost	Nexthop
A				
D				
B				
C				
S				

Construction of L3 Routing Table -> Phase 2 -> SPF Algorithm

Execution :

```
While(Q not Empty) {
        curr_node = dequeue(Q)                                   4

        if(curr_node == spf_root){
                for each nbr of spf_root
                        Enqueue(Q, nbr)
                continue;
        }

        spf_root.record_result( nxthop{curr_node}.list(          5

        for each nbr of curr_n                                   6
                if(cost(curr_node) + cost(curr_node, nbr)
                < cost( nbr ))
                        cost(nbr) = cost(curr_node) + cost(curr_node, nbr)
                        nxthop{nbr}.flush()                      6.1
                        nxthop{nbr}.append(nxtho      node})
                        re-enque(nbr)

                else if(cost(curr_node) + cost(curr_node, nbr) == cost(nbr))   /*ECM
                        nxthop{nbr}.append(nxthop      ode})     6.2

} /*While loop ends*/
```



LSDB of S

(This DS is present in local memory Of Device S)

spf_root Result

Dest Node	Oif	Gateway	cost	Nexthop
A	eth1	2.2.2.2	3	A
D	eth2	1.1.1.1	4	D
B	eth1	2.2.2.2	7	A
C	eth2	1.1.1.1	10	D

Construction of L3 Routing Table

Let us Break down the problem into phases

- Phase 1 : Build the input graph
  - How Networking Routing Devices Build the Input Graph Data Structure
- Phase 2 : Run the shortest path algorithm (Dijkstra Algorithm) on the Input Graph
  - You must understand the Dijkstra algorithm, refer to standard book
- Phase 3 : Routing Table Calculation
  - Convert the output of phase 2 into actual routing table

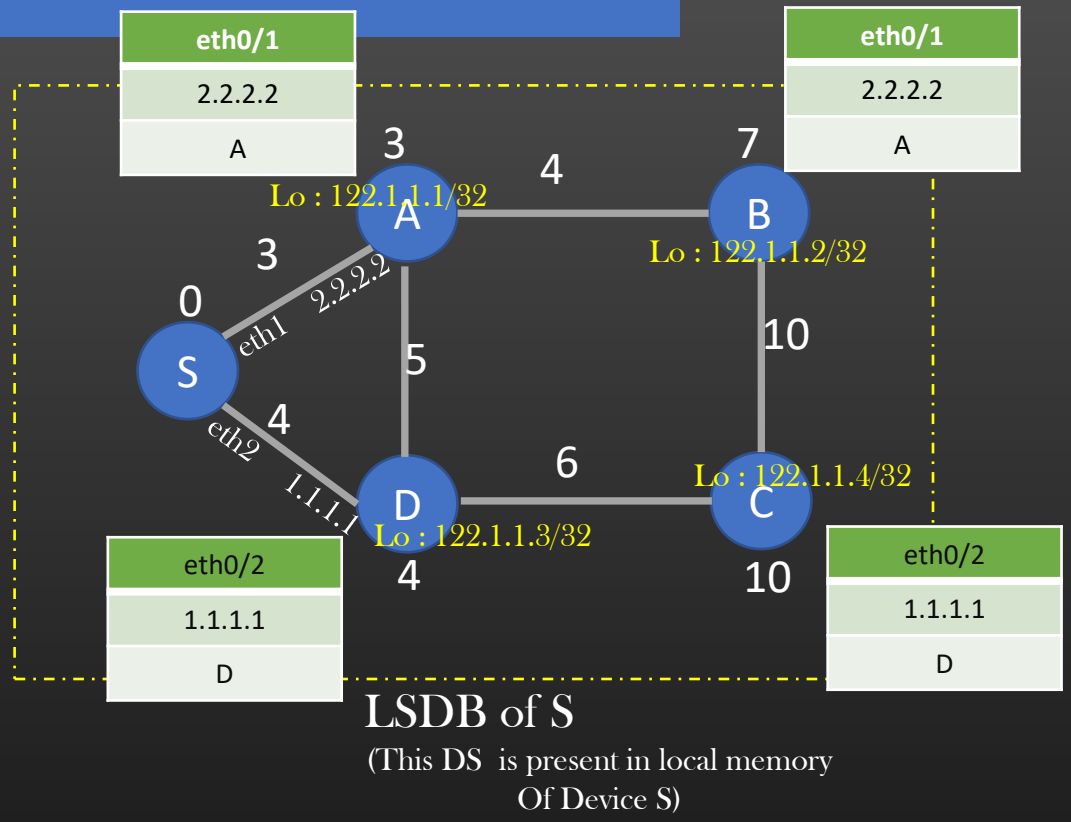
Phase 3 : Routing Table Calculation

- Phase 2 Results is a Routing Table of spf root node in Raw form

Spf_root S Result

Phase 2 Result

Dest Node	Oif	Gateway	cost	Nexthop
A	eth1	2.2.2.2	3	A
D	eth2	1.1.1.1	4	D
B	eth1	2.2.2.2	7	A
C	eth2	1.1.1.1	10	D



LSDB of S

(This DS is present in local memory Of Device S)

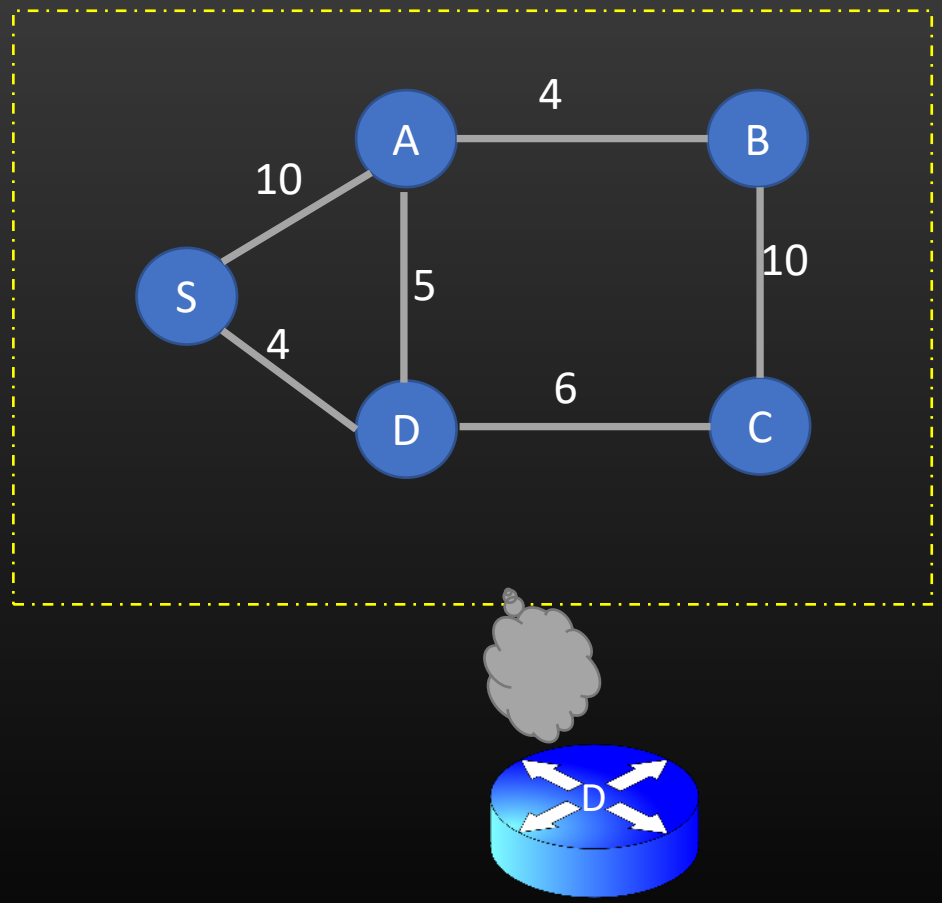
Replace Dest Nodes With their loop-back Address

Dest Address	Oif	Gateway	cost	Nexthop
122.1.1.1/32	eth1	2.2.2.2	3	A
122.1.1.3/32	eth2	1.1.1.1	4	D
122.1.1.2/32	eth1	2.2.2.2	7	A
122.1.1.4/32	eth2	1.1.1.1	10	D

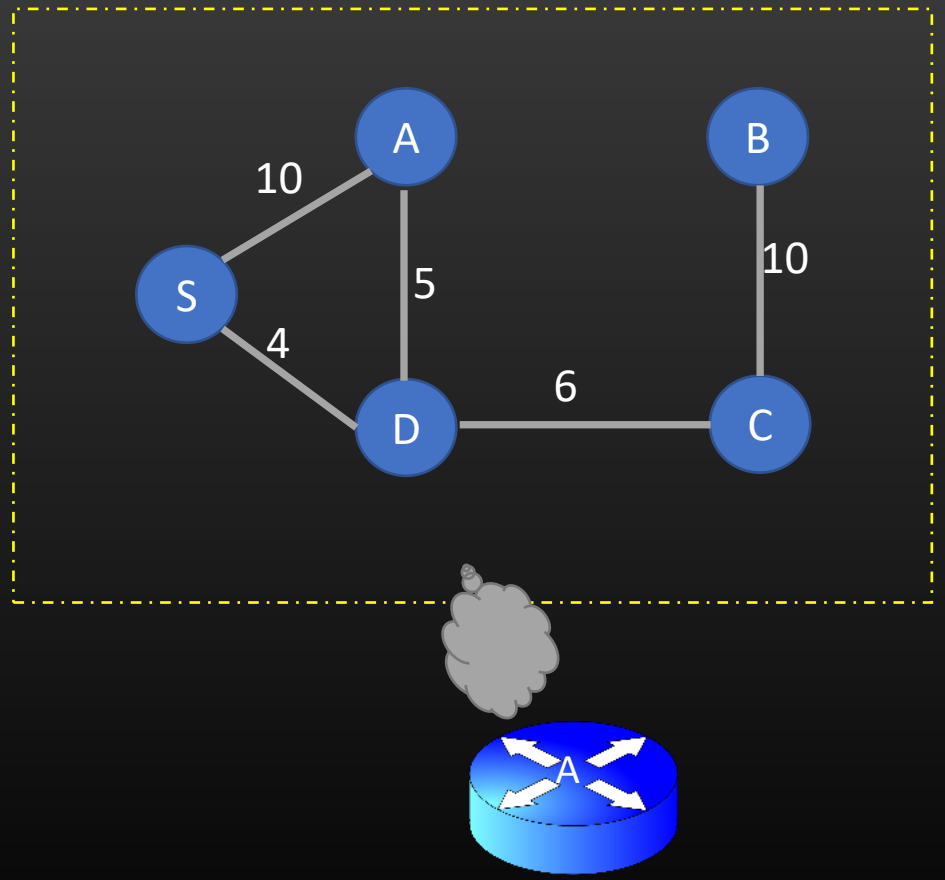
Final Routing Table of Spf Root S

Construction of L3 Routing Table -> Loop Free Guaranteed

- Routing Table of all L3 devices ensures Loop Free traffic flow because :
  - Every Device has the same view of the network Topology
  - Every device computes the shortest path to every other reachable device



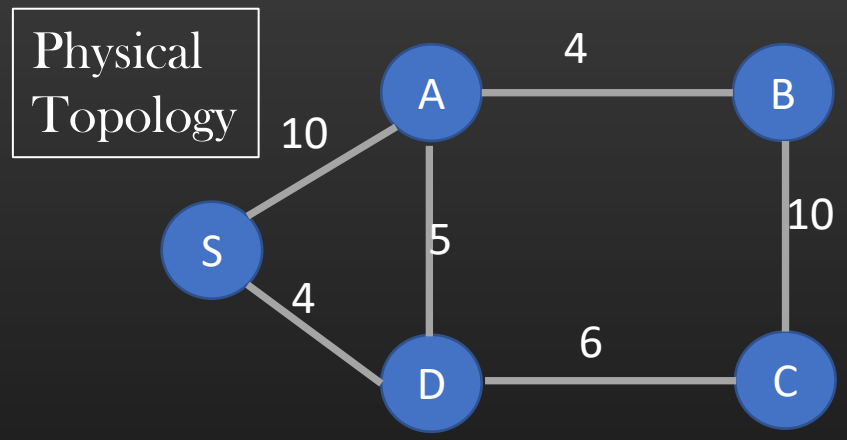
To Reach B : NH is A



To Reach B : NH is D

Construction of L3 Routing Table -> Convergence

- All Devices must update their Routing tables in a consistent way whenever there is a change in topology
  - Addition/removal of new device
  - Node/link failure
  - Change of cost of links by admin



If at time t = 0 , link fails and
At time t + t1 , all other devices recvs new LSPs of A & B
Then, t1 is the transient time interval when traffic is
Subjected to loops

A will withdraw B's entry from its nbrship db, and regenerate & flood LSPs
B will withdraw A's entry from its nbrship db, and regenerate & flood LSPs
Every other device (including A and B themselves) would recv new LSPs of A and B, and update their LSDB
Result : Every Device attains the new consistent view of the Network Graph(LSDB), re-triggers SPF algorithm
And update routing tables. This is called Convergence

Construction of L3 Routing Table -> Summary

- Each Node in the Network Graph, independently run SPF algorithm on its LSPDB to compute its local routing Table
- Spf Result Table (Phase 2 output) is stored with in Spf root node's local data structures
- Result : Every Node knows the nexthop information to reach a remote Destination
  - In other words, Every node can ping every other node via shortest path
- Algorithm looks after ECMP case
- HomeWork : Time & Space Complexity of the Algorithm – Interview Quesiton !
- Interior Gateway Routing Protocols – OSPF, ISIS uses SPF algo to compute Routing Tables

L3

Thank you

Routing table

Construction

Broadcast Domain
&
Collision Domain

Broadcast Domain and Collision Domain

- *A BD for a given hosts/machine X is the set of all machines in the subnet which shall receive the ARP broadcast msg generated by machine X*

- *A CD for a given hosts/machine X is the set of all machines in the subnet which suffers collision with X if they also chose to transmit frame at the same time*

- For a given host in the network, you can be asked to find the BD and CD

Tip : While finding the BD is easy, follow the below analogy to find CD

To find Collision Domain, Assume the wire is like electrical wire which carries electric current

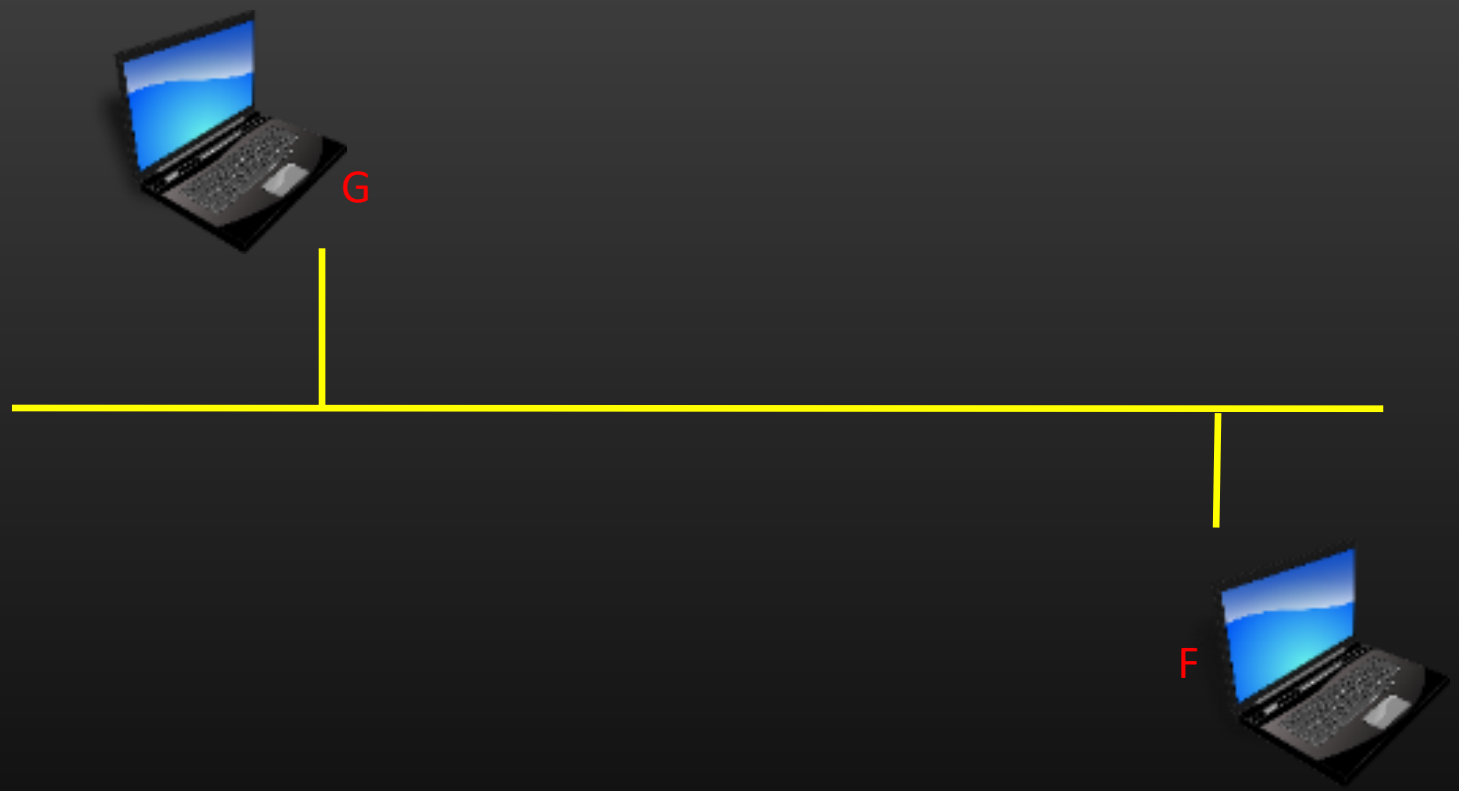
If some host transmit frame on the wire (= wire carries current), and you will die if you touch it (= try to transmit your frame)

You can touch it only when it has no current (you can transmit only when no other host is transmitting)

Example ? Ok . . .

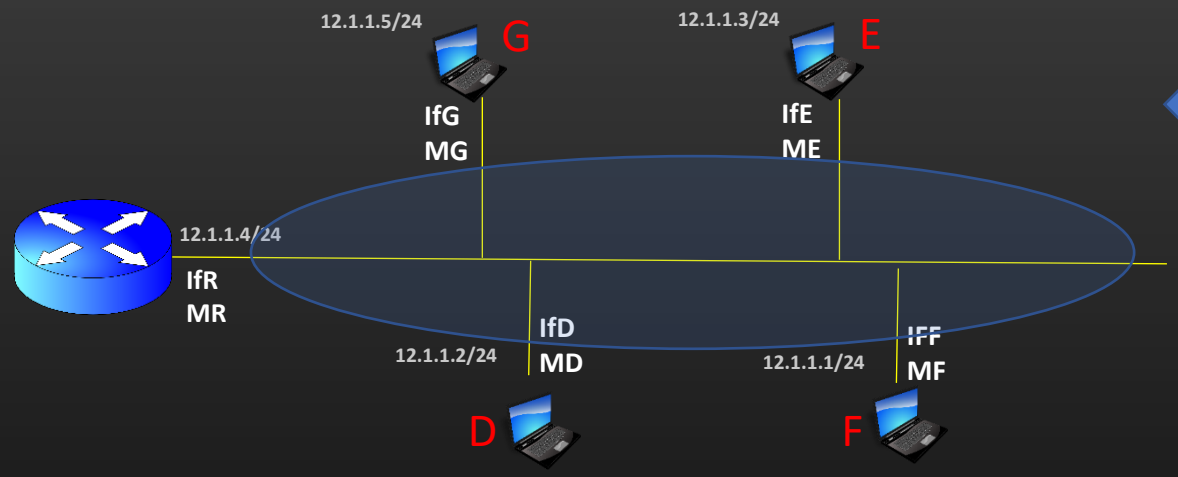
Broadcast Domain and Collision Domain

What Collision is ?



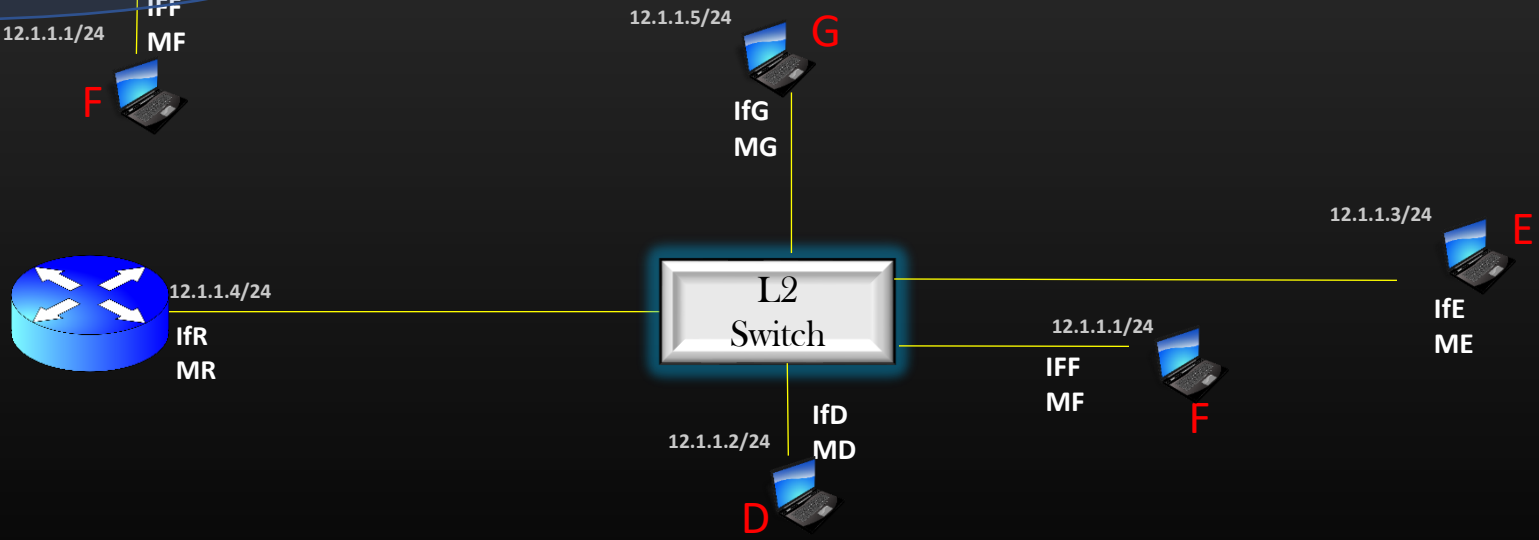
Broadcast Domain and Collision Domain

- *Example*



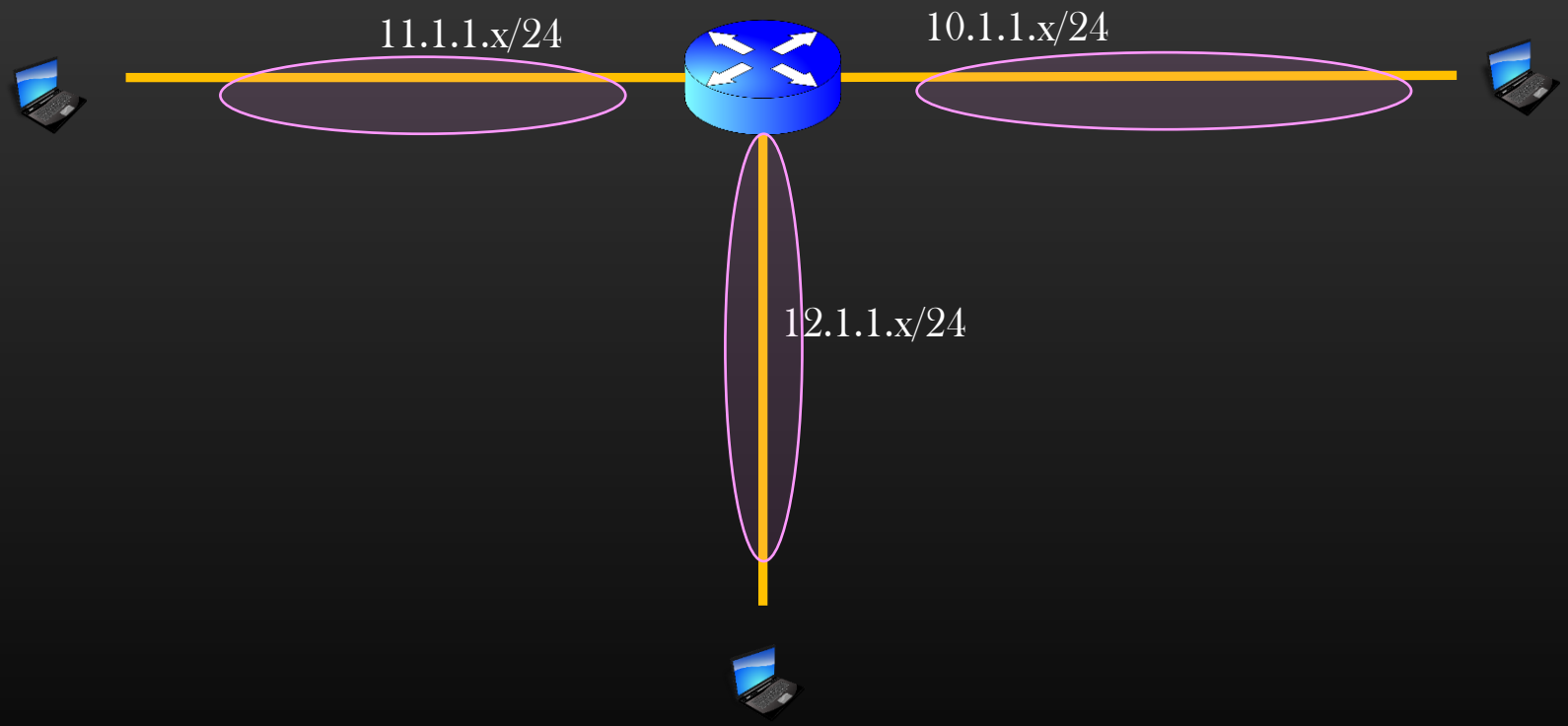
If G Transmits, E,F,D has to wait
1 CD, 1 BD

Any host can transmit any time
1 BD, 5 CD
1 BD consisting 5 CDs



Broadcast Domain and Collision Domain

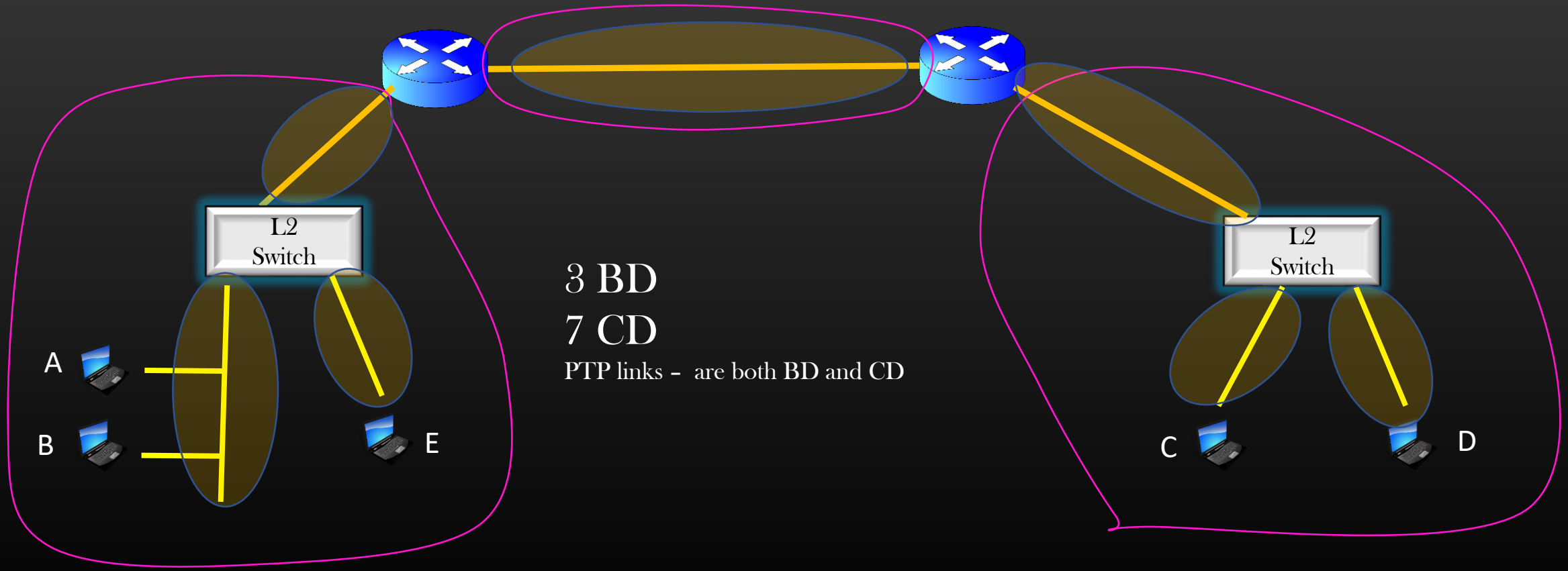
- *L2 Switches Separates Collision Domain*
- *L3 routers separate Broadcast Domains*



Broadcast Domain and Collision Domain

- *A BD for a given hosts/machine X is the set of all machines in the subnet which shall receive the ARP broadcast msg generated by machine X*

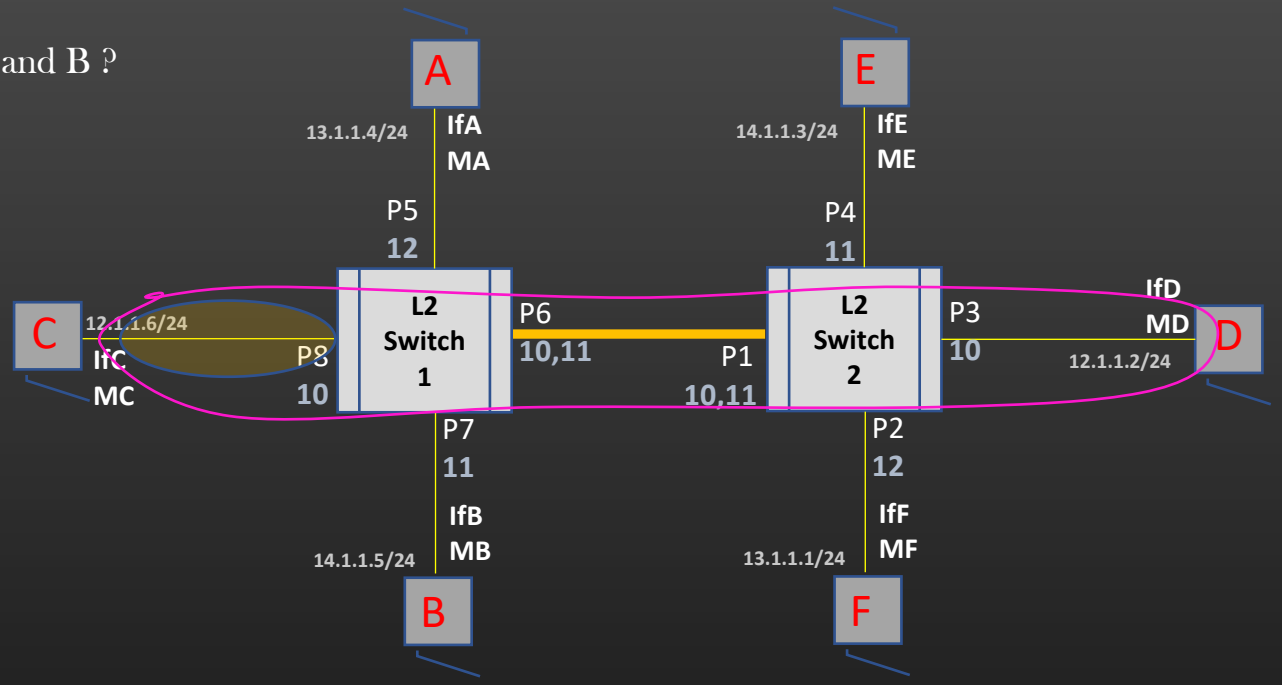
- *A CD for a given hosts/machine X is the set of all machines in the subnet which suffers collision with X if they also chose to transmit frame at the same time*



Broadcast Domain and Collision Domain

Identify BD and CD for hosts A, C and B ?

For C



Assignment , find BD and CD for C and B

Broadcast Domain

&

Collision Domain

Thank you

VLANS

Virtual Local Area Network

Agenda

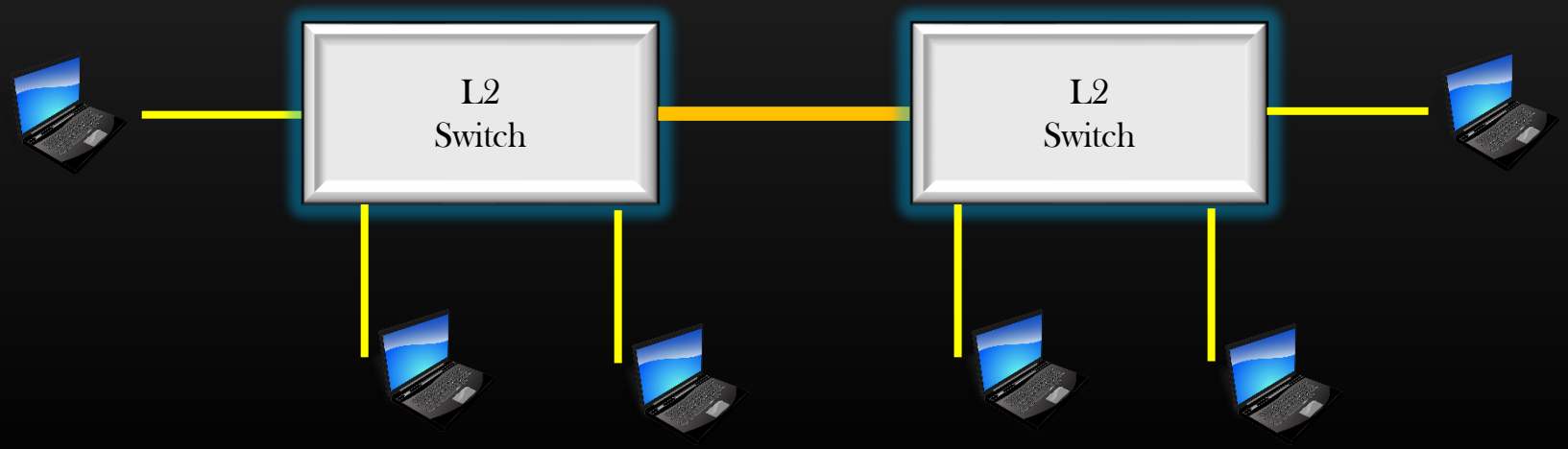
What are LANs ?

Drawback of LANs

How Vlans addresses LANs drawbacks ?

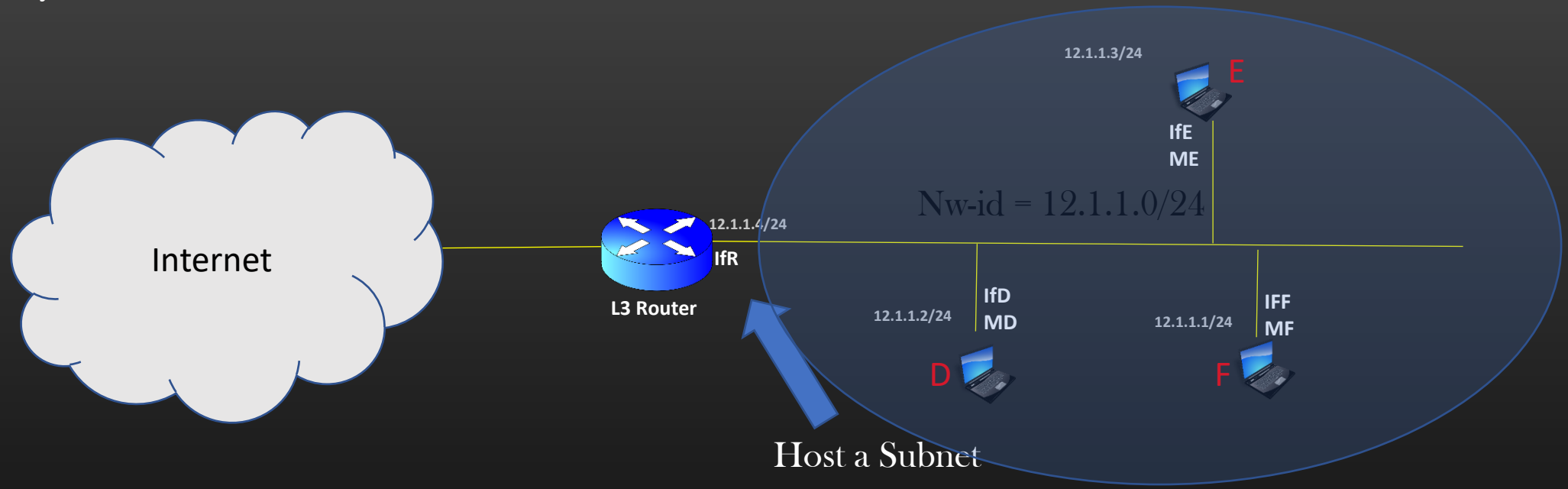
How Vlans are deployed using L2 Switches

L2 routing using VLANs



What is LAN ?

- We already have learnt what subnet is , let us do some revision



- Hosts, D, E and F are in same subnet because
  - They are assigned ip address which are in same subnet
  - They are connected to same physical wire

- Hosts D,E and F forms traditional local area network (LAN)
- LAN is an outdated technology, Vlans has taken over
- LAN enforces the mobility constraint on hosts/machines for to become a member of a subnet (Drawback !!)

When should multiple hosts be placed in the same subnet ?

- *What do you think, what would you choose to place two or more hosts in the same subnet ?*

- When they perform same logical operations
  - All CSE students are in one subnet 1, and all ECE are in other subnet 2

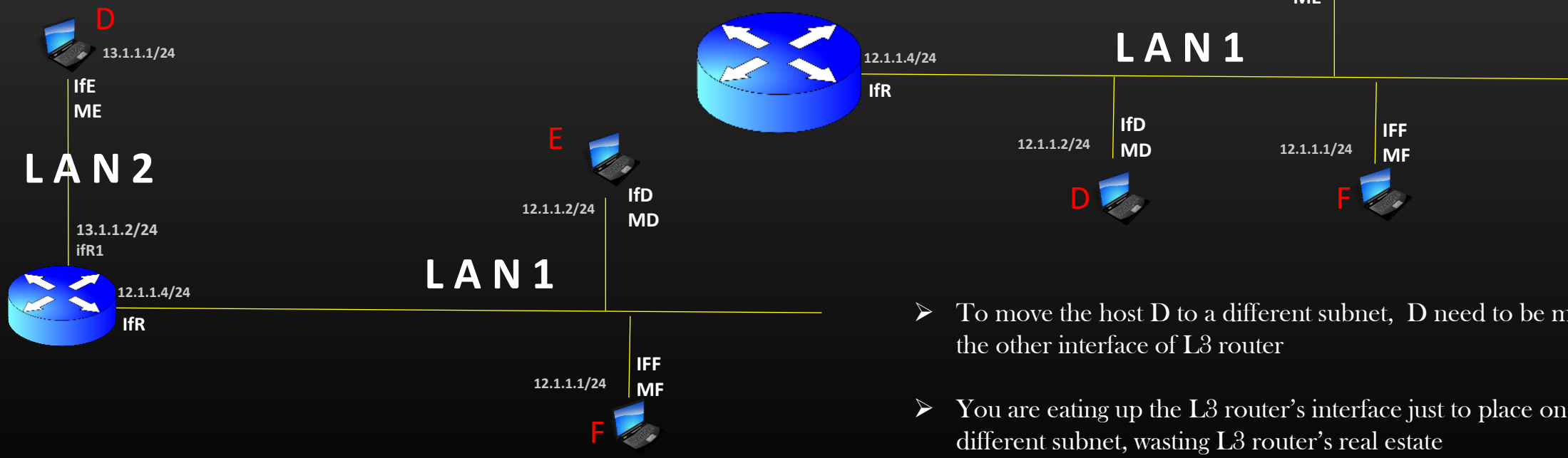
- Need same Network resources
  - Allow access to all hosts in Subnet 1 to website www.csepracticals.wixsite.com/csepracticals

- Need same quality QoS from rest of the network
  - Allow all hosts in subnet 1 a download speed of 50 mbps, whereas only 10 mbps for all hosts in subnet 2

- Fast and secure communication
  - All hosts in subnet 1 can securely communicate with each other because no traffic from outside subnet 1 Or any traffic from inside of subnet 1 ever leaks !

Problem with LANs - Immobility

- Can you place host D in different subnet, and host F and E in different subnet ?
- All hosts have to be assigned an ip address which shares the same network Id as that of gateway Router
- So the only way to move host D to a different subnet is to physically move the host D
  (So, you need to change the physical cabling !! Painful !!)



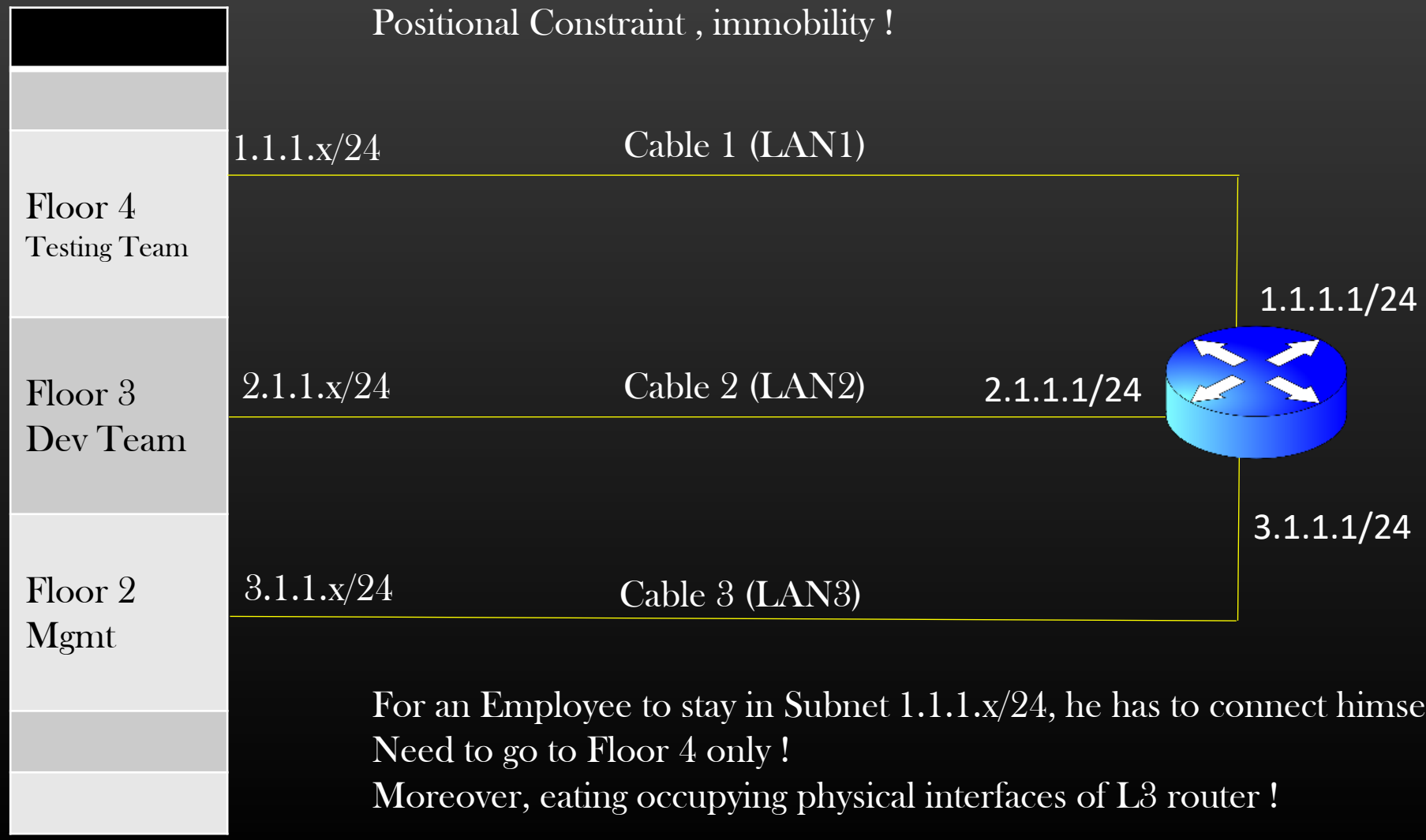
- To move the host D to a different subnet, D need to be moved behind the other interface of L3 router
- You are eating up the L3 router's interface just to place one host in a different subnet, wasting L3 router's real estate

Let us try to understand the real world implication of this problem ...

Problem with LANs - Immobility

Amazon Co

Amazon Administration is forced to keep all the employees of same Dept/Team at same floor !
Positional Constraint , immobility !



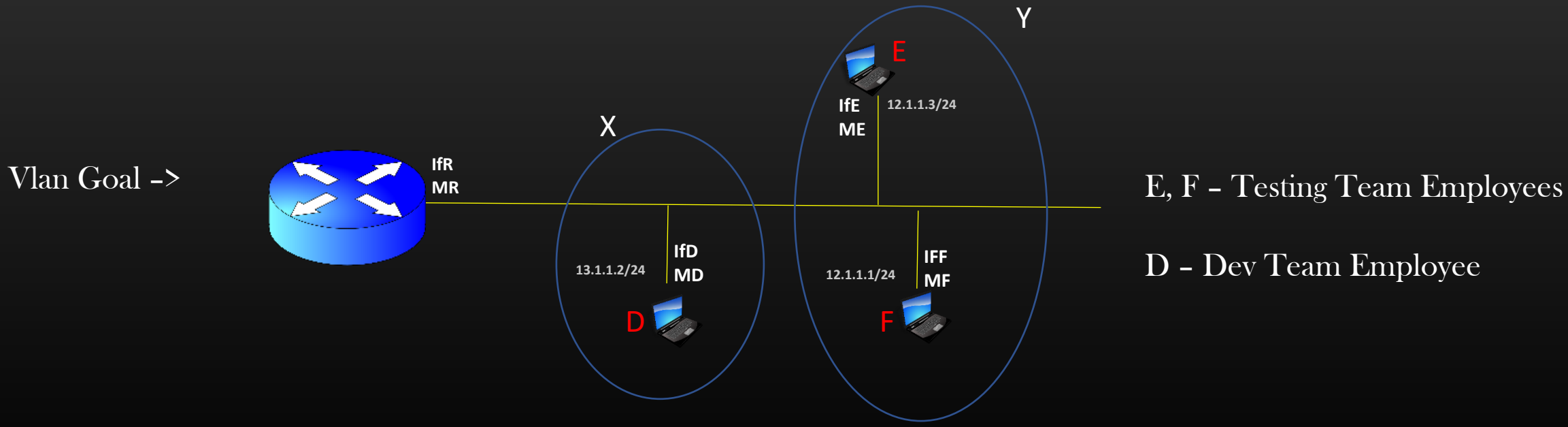
For an Employee to stay in Subnet 1.1.1.x/24, he has to connect himself to cable 1 only, and hence Need to go to Floor 4 only !
Moreover, eating occupying physical interfaces of L3 router !

Problem with LANs - Security

If they all are connected to same physical LAN, they would have equal access to company resources !
Company would not want to give the same permissions and privileges to Visitors/Interns
So, Company would want to separate out Regular employees with interns/visitors in different subnets
and can impose access limitations on subnet in which interns/Visitors are present

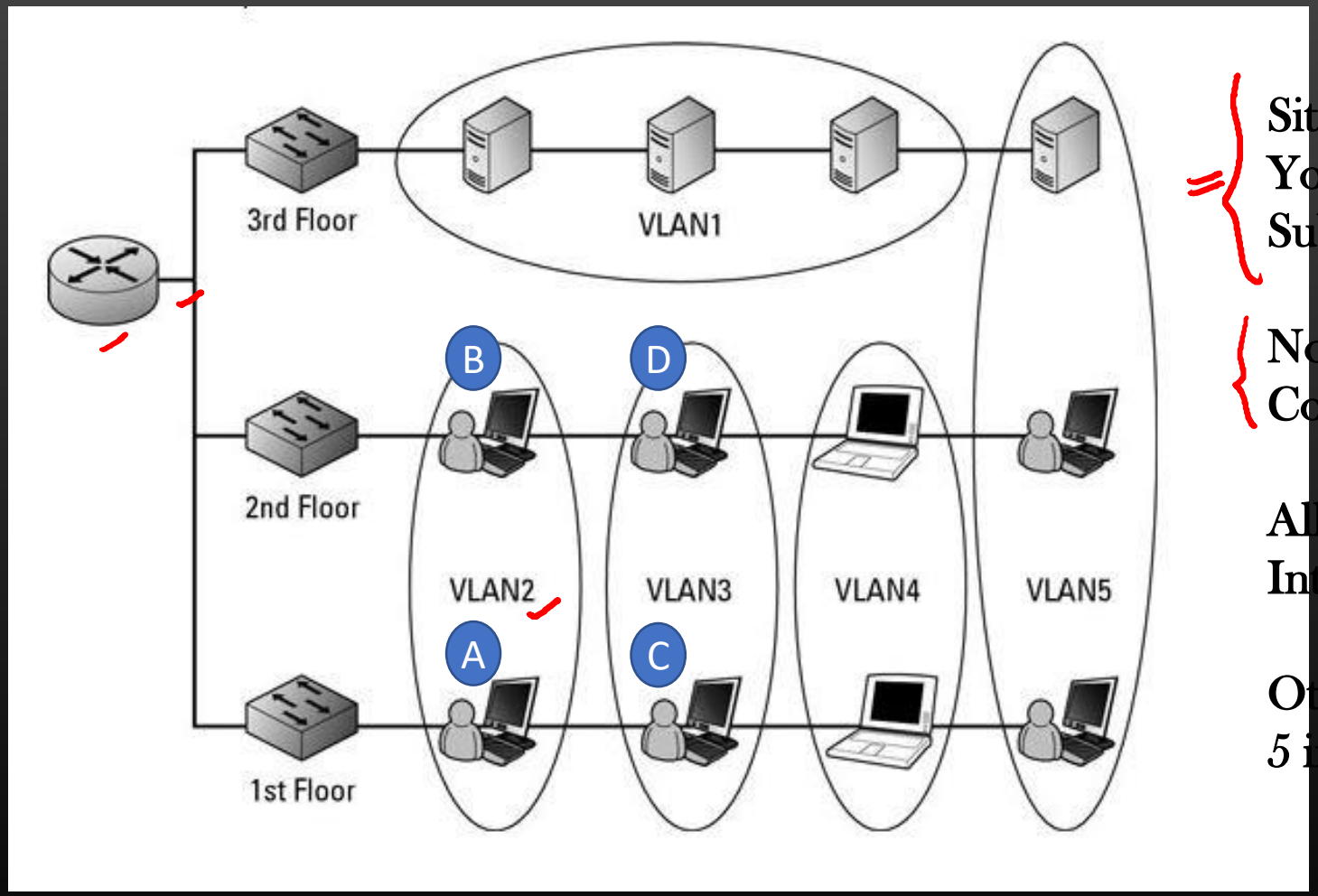
VLANs help Remove the positional constraint / Immobility

- Vlans allows different hosts which are connected to same network (physical wire) to reside in different subnets
- Vlan will allow host D to be in subnet X, while E and F could reside in a subnet Y While all hosts still sit behind the same L3 router Interface !
- Vlan allows logical Grouping of hosts by virtue of their functions rather than by virtue of their position



So, your friend sitting by your side could be connected to different subnet altogether

VLANs help Remove the positional constraint / Immobility



Sit Wherever you want in the building
You can always connect to the
Subnet you want !

No need to go to floor no X only to
Connect to subnet S

All subnets sits behind the same Router
Interface, Wow !!

Otherwise you would have consumed
5 interfaces of a L3 router

A testing team engineer can sit right by the side of Dev Engineer while both stay connected to their respective Subnets

VLANs

1. We have Just seen a Major problem which Vlans can solve – The problem of Immobility & Security

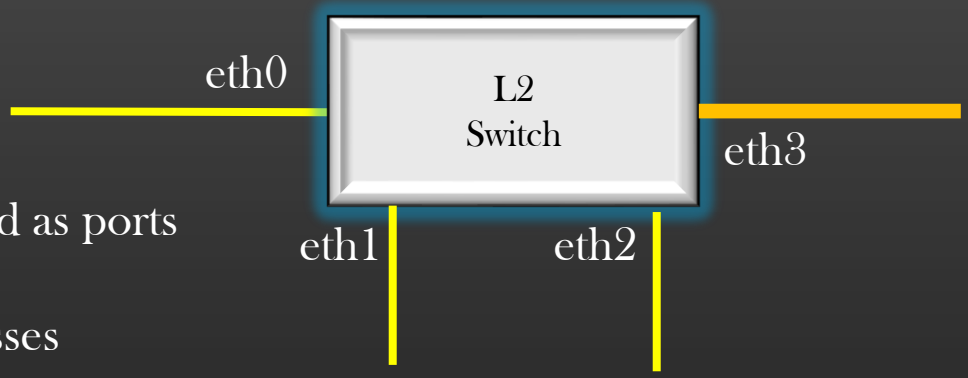
2. Vlans is a Layer 2 concept, and is implemented by L2 device called Switch

3. Let us now understand, How can we create Vlans using Switch and see what other problems it can solve, Yes there are more benefits of Vlans, but hold on !

4. By Now you must have got some Idea about Vlans, more we will understand when we shall see Vlans in action

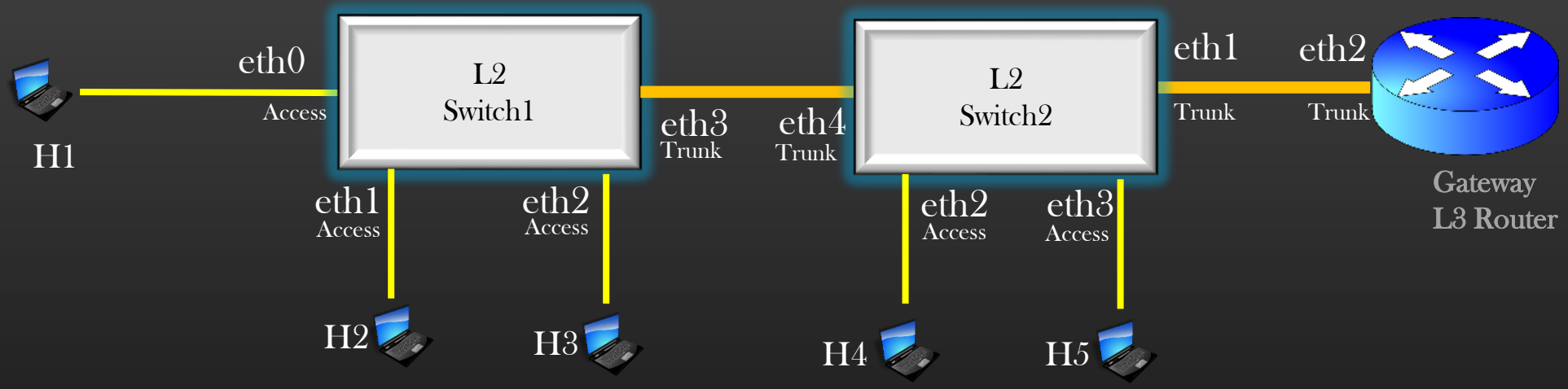
5. Lets begin with Vlan Concepts . . . Pls fasten your Seat-Belts !

Lets Build the pre-requisite knowledge first



- A Sample Switch is shown in the Diagram
- Switch has interfaces as shown. Interfaces of L2 devices are usually termed as ports
- Switch is a L2 device, it works only with MAC addresses, forget IP addresses
- Interfaces of a switch are not configured with IP addresses, like L3 router's Interfaces
- Interfaces of a switch can operate in one of two modes
  - Access mode (Yellow interfaces)
  - Trunk Mode (Orange interfaces)
- It is the responsibility of the Admin to configure the interfaces of the switch to operate in one of two modes
  - Guidelines
    - Access mode interfaces are connected to hosts
    - Trunk mode interfaces are connected to gateway L3 router or another L2 switch

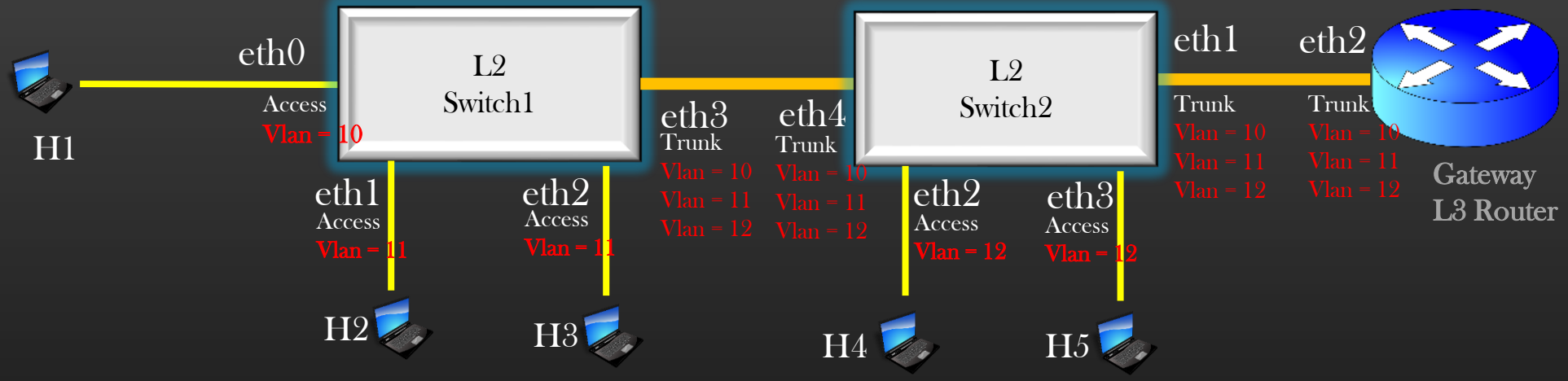
VLANs – Access and Trunk ports



Going forward . . .

- An interface of a switch can be configured to operate in a vlan(s)
- It is done by assigning a Vlan ID to the interface
- Access ports can be configured to operate in exactly one vlan
- Trunk ports can be configured to operate in more than 1 vlans

VLANs – Access and Trunk ports



Going forward . . .

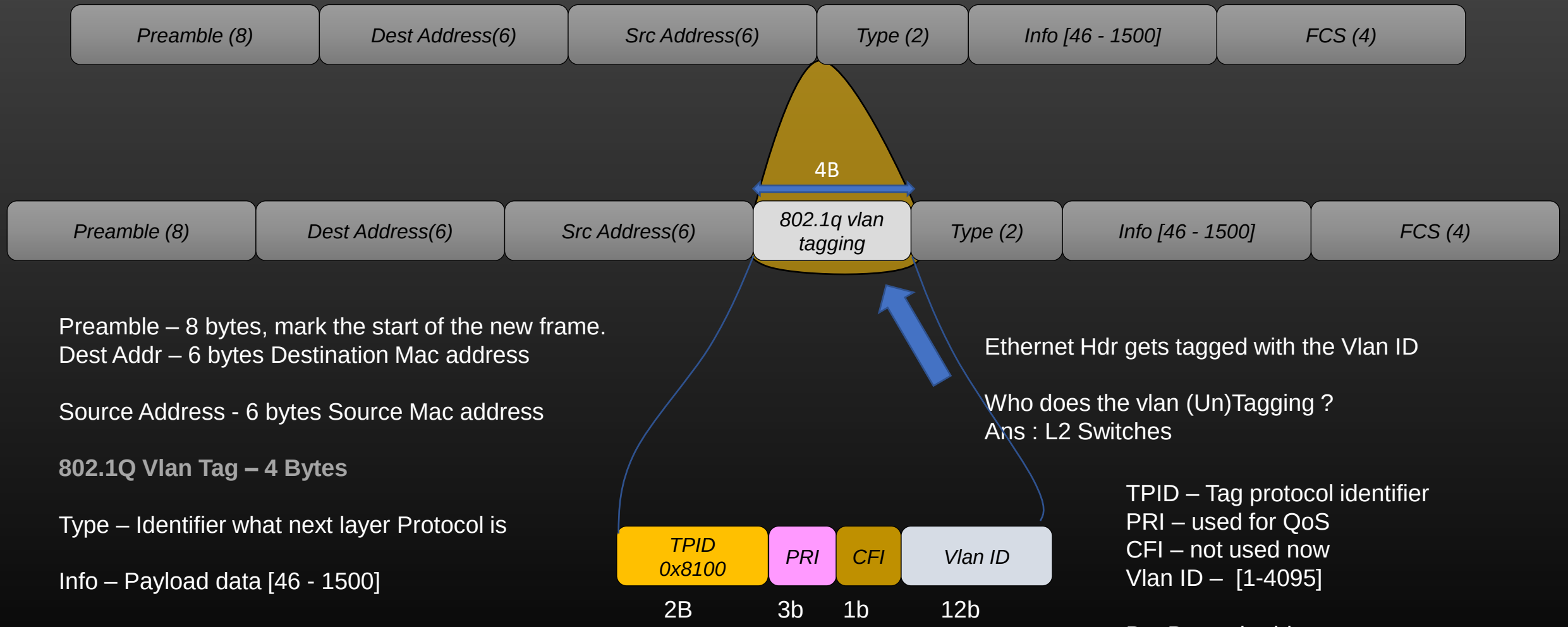
- An interface of a switch can be configured to operate in a vlan(s)
- It is done by assigning a Vlan ID to the interface
- Access ports can be configured to operate in exactly one vlan
- Trunk ports can be configured to operate in more than 1 vlans

ACCESS MODE = Exactly 1 VLAN

TRUNK MODE = More than 1 VLAN

Vlan tagging

Vlan Tagged Ethernet Header format



Preamble – 8 bytes, mark the start of the new frame.
Dest Addr – 6 bytes Destination Mac address

Source Address - 6 bytes Source Mac address

802.1Q Vlan Tag – 4 Bytes

Type – Identifier what next layer Protocol is

Info – Payload data [46 - 1500]

FCS – checksum
When frame gets tagged/untagged, FCS is recalculated

Ethernet Hdr gets tagged with the Vlan ID

Who does the vlan (Un)Tagging ?
Ans : L2 Switches

TPID – Tag protocol identifier
PRI – used for QoS
CFI – not used now
Vlan ID – [1-4095]

B – Bytes, b - bits

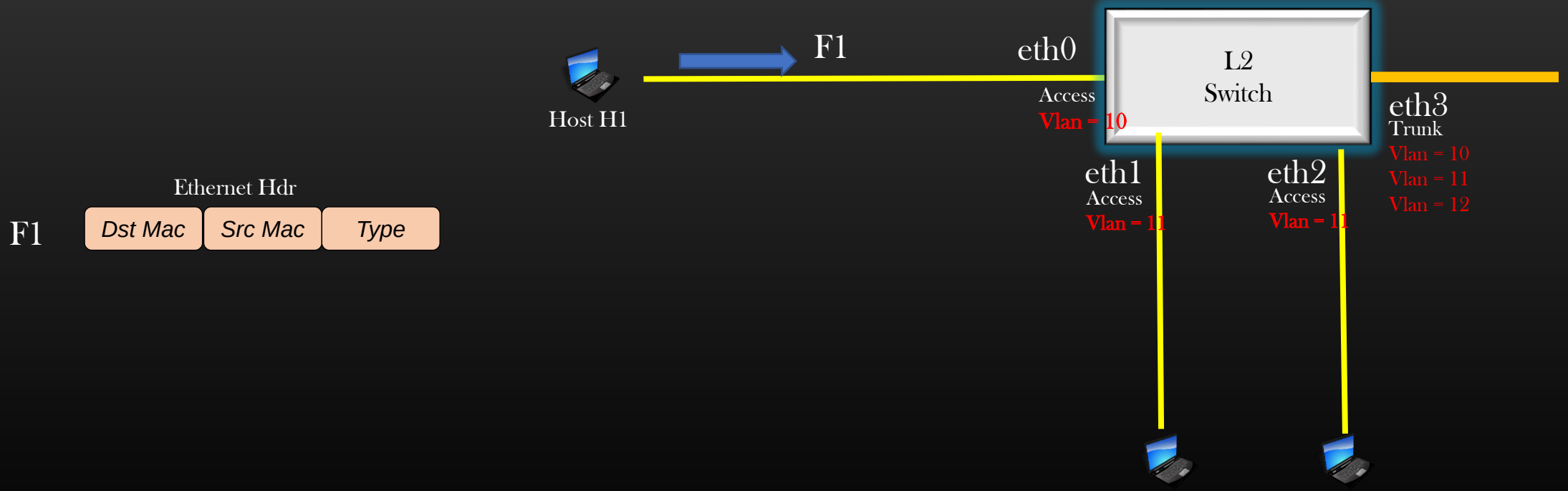
Only ethernet header is tagged or untagged when frame moves across switch boundaries, no change in any other hdr of the frame

5 Rules of VLAN TAGGING

Rules of Vlan Tagging

Rule 1

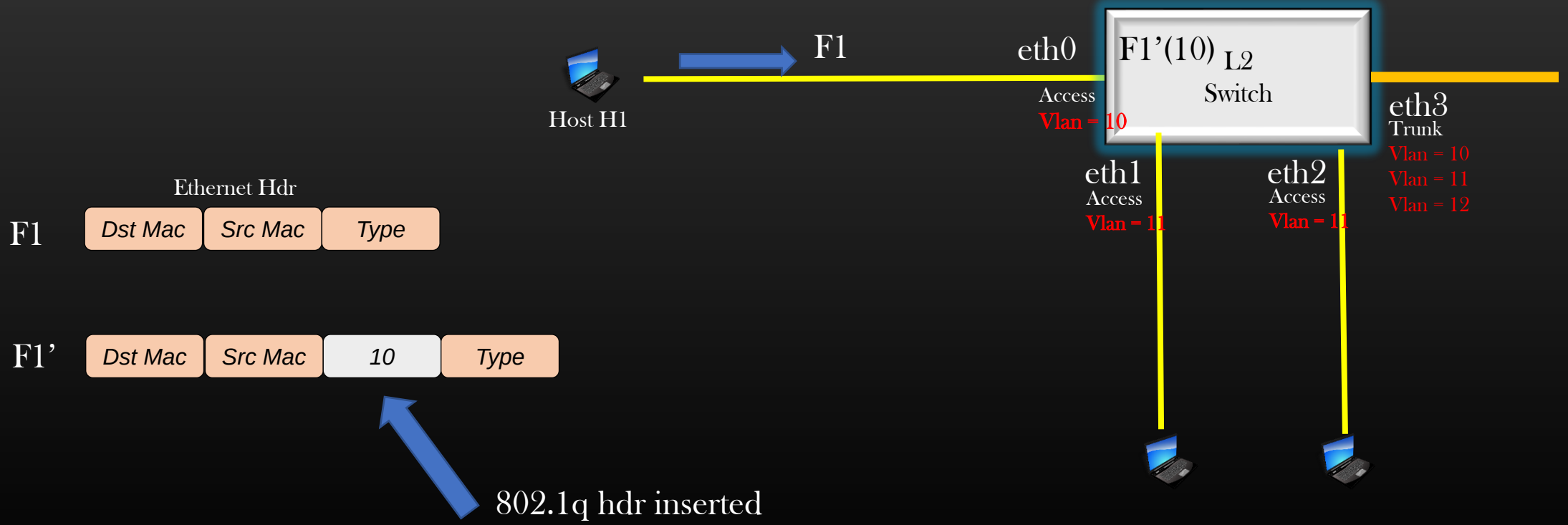
- *Hosts are Vlan unaware*
  - They neither generates Vlan tagged packet nor should they receive vlan tagged packets
  - Eg, Any frame generated by H1 (Or any other Hosts) is always untagged



Rules of Vlan Tagging

Rule 2

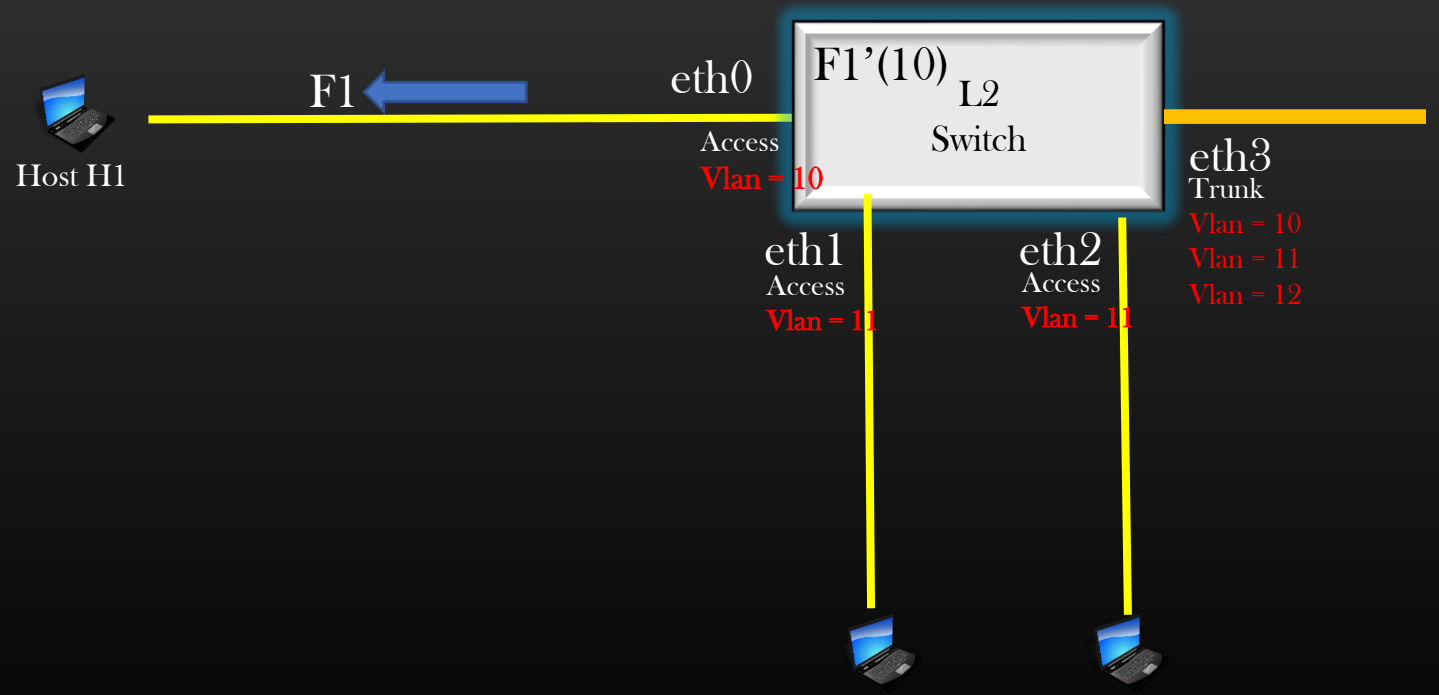
- *When Untagged frame enters the switch from interface operating in Access mode, they get tagged with corresponding vlan id*
  - Untagged Frame F1 generated by host H1 enters L2 Switch from interface eth0
  - F1 Frame get tagged with vlan 10, resulted fame is F1'

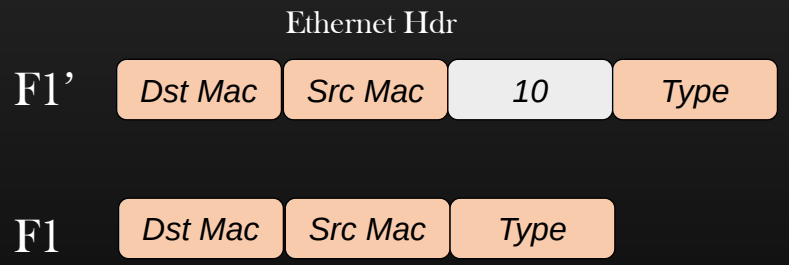


Rules of Vlan Tagging

Rule 3

- *When tagged frame Leaves the switch from interface operating in Access mode, they get untagged with corresponding vlan id*
  - Frame F1' Tagged with vlan id 10 becomes untagged with it leaves L2 Switch from interface eth0, resulted Frame is F1
  - Frame Tagged with vlan id != 10 is not allowed to exit out of interface eth0 of L2 Switch because of mismatch vlan id

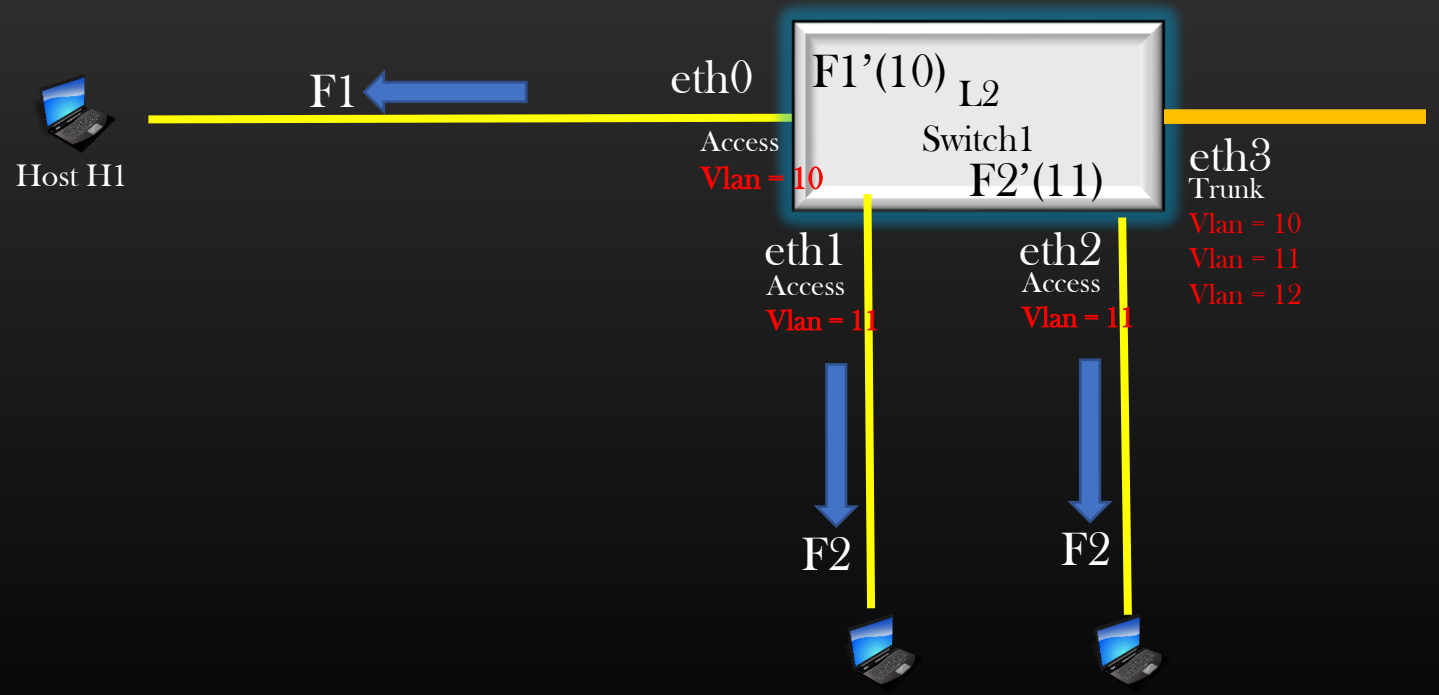


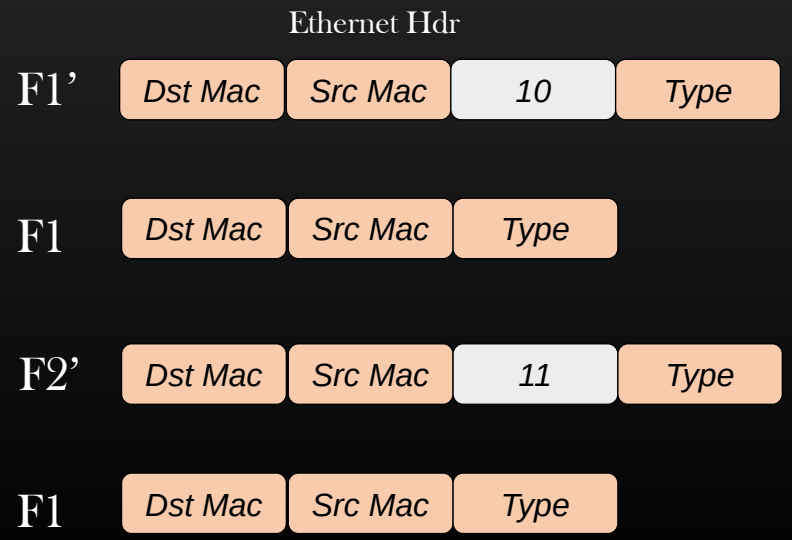


Rules of Vlan Tagging

Rule 4

- *Frame tagged with vlan X are allowed to exit out of those interfaces of switch (Access or Trunk) which are configured with same vlan id X*
  - Frame F1' Tagged with vlan id 10 is allowed to exit the L2 switch from interface eth0 and eth3 only
  - Frame F2' Tagged with vlan id 11 is allowed to exit the L2 switch from interface eth1, eth2 and eth3 only

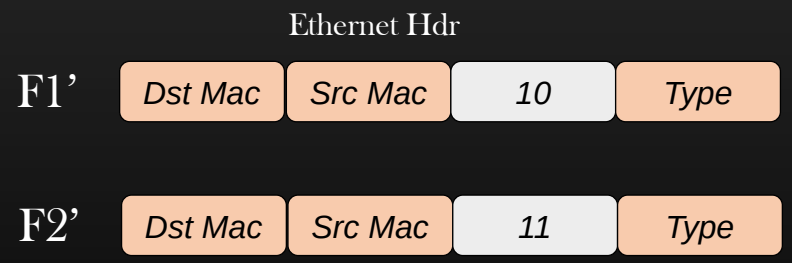


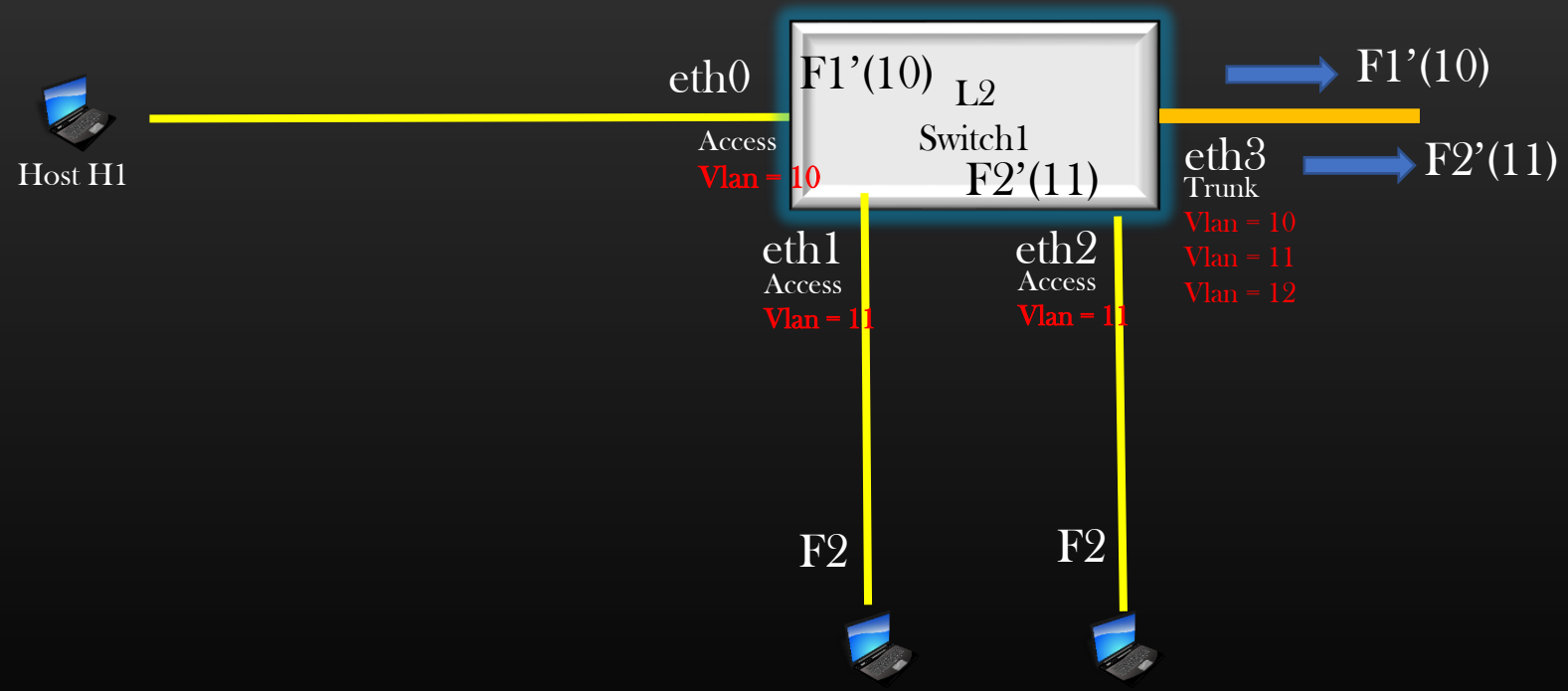


Rules of Vlan Tagging

Rule 5

- *Trunk ports are only pass through ports, they allow the tagged frames to pass through them (exit or enter L2 switch) which are tagged with matching vlan id. Untagged frames are dropped by trunk interfaces*
  - Frame F1' Tagged with vlan id 10 is allowed to exit the L2 switch from trunk interface eth3 without any change in mac hdr
  - Frame F2' Tagged with vlan id 11 is allowed to exit the L2 switch from interface eth3 without any change in mac hdr
  - Frame F3' Tagged with vlan id 15 is not allowed to exit the L2 switch from interface eth3





Mac address table of vlan enabled L2 switch

Dst Mac (Key)	Outgoing interface
MA	eth0
MB	eth1

Not a vlan enabled Switch
L2 switch inspect only the dst mac address In the ethernet hdr of the frame to take a Forwarding decision

Vlan Id (Key)	Dst Mac (Key)	Outgoing interface
10	MA	eth0
11	MB	eth1

Vlan enabled Switch
L2 switch inspect 801.1q vlan hdr + dst mac address to take a Forwarding decision

Mac address table of vlan enabled L2 switch

- L2 switch does mac learning

1. Whenever the untagged frame enters the L2 switch from interface i operating in Access mode and interface I is configured in vlan X,

2. Whenever the tagged frame with vlan Y enters the L2 switch from interface J operating in Trunk mode and interface J is configured with atleast vlan Y

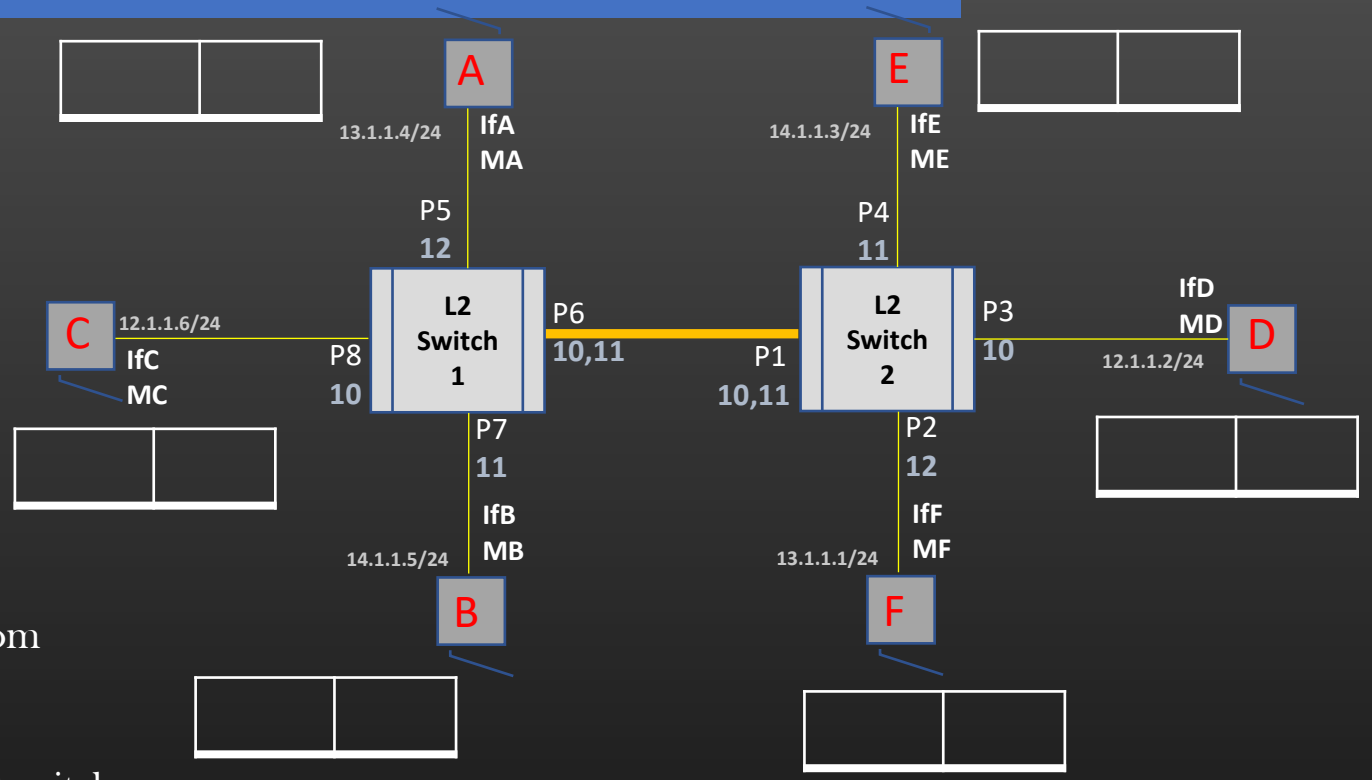
3. If the Learned entry already exists, L2 switches refreshes the entry (restart the expiry timer)

Vlan Id (Key)	Dst Mac (Key)	Outgoing interface
X	Src MAC	I
Y	Src MAC	J

Remember, Mac learning is done using Src mac of the frame, not dst mac address , Whereas L2 switch forwards the frame based on Dst mac

Example – Vlan based Routing

- Lets see end to end example here
- C wants to communicate with D, and Network is supported by two L2 switches
- Tables used – ARP table for devices, and Mac table for L2 switches, initially all tables are empty
- Also Notice that, IP address assigned to host devices are in compliance to subnet scheme – All devices in same subnet (same Vlan) should derive same network ID from their ip addresses
- I assume , you have already gone through Detailed study of L2 switches in L2 routing module of this course



Vlan ID	Mac address	Outgoing port no

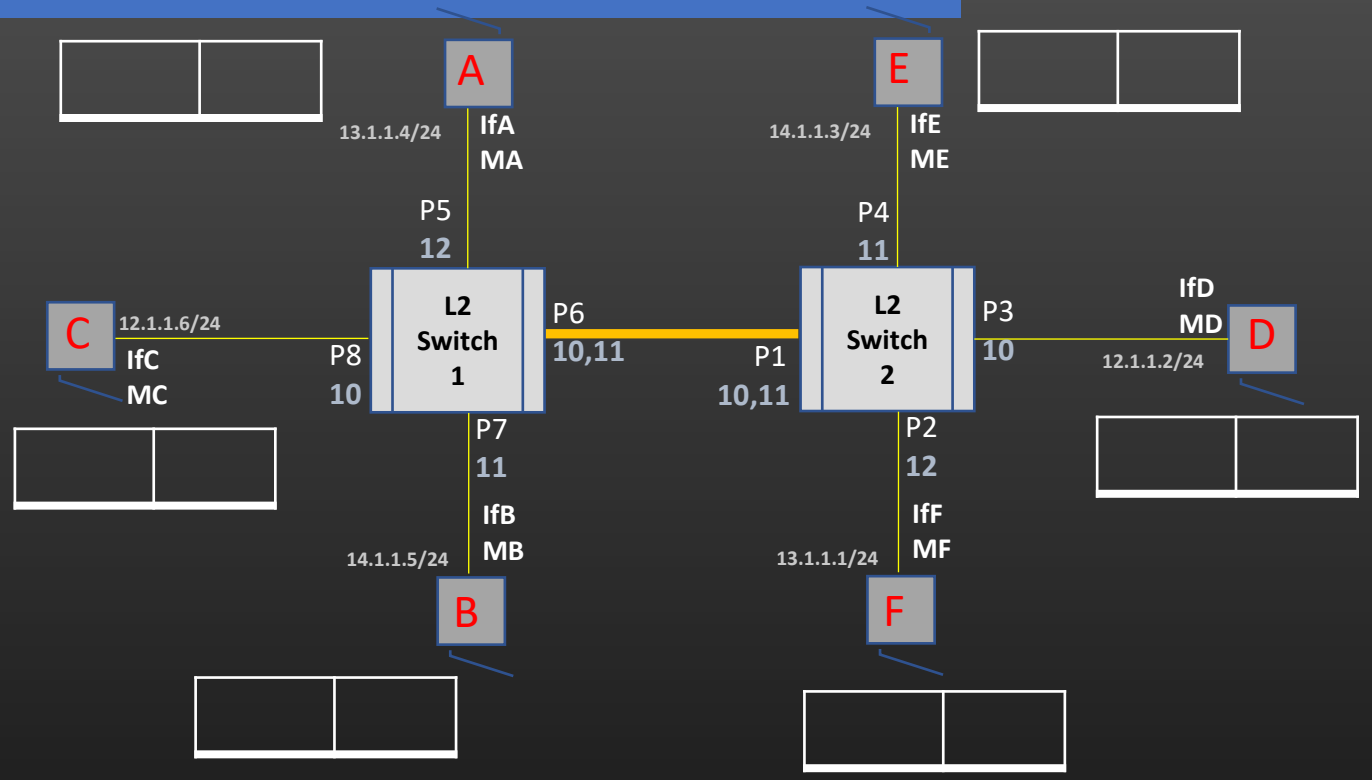
MAC table 1

Vlan ID	Mac address	Outgoing port no

MAC table 2

Example – Vlan based Routing

1. C wants to send Data to D, C knows D's ip address = 12.1.1.2
2. From D's ip address, C figure out that D is in its local subnet
3. C needs D's MAC, C looks up its ARP cache for D's ip = 12.1.1.2
4. C do not have D's MAC, therefore C launches ARP Broadcast msg out of interface ifC (which is in same subnet – 12.1.1.0/24)
   ARP Broadcast msg :
   src mac = MC, dst mac = FFFFFFFF , dst ip = 12.1.1.2
5. ARP broadcast msg enters L2 switch 1, get tagged with vlan 10
6. L2 Switch 1 does mac learning : Insert 10, MC, P8
7. L2 switch 1 Broadcast the msg out of all interface tagged with vlan 10
8. L2 switch 1 forwards ARP B msg out of interface P6 only
9. L2 switch 2 receives ARP B msg on interface P1
10. L2 Switch 2 does mac learning : Insert 10, MC, P1
11. L2 switch 2 Broadcast the msg out of all interface tagged with vlan 10
12. L2 switch 2 forwards ARP B msg out of interface P3 only
13. When ARP B msg exit L2 switch 2 out of interface P3, ARP B msg become untagged



Vlan ID	Mac address	Outgoing port no
10	MC	p8

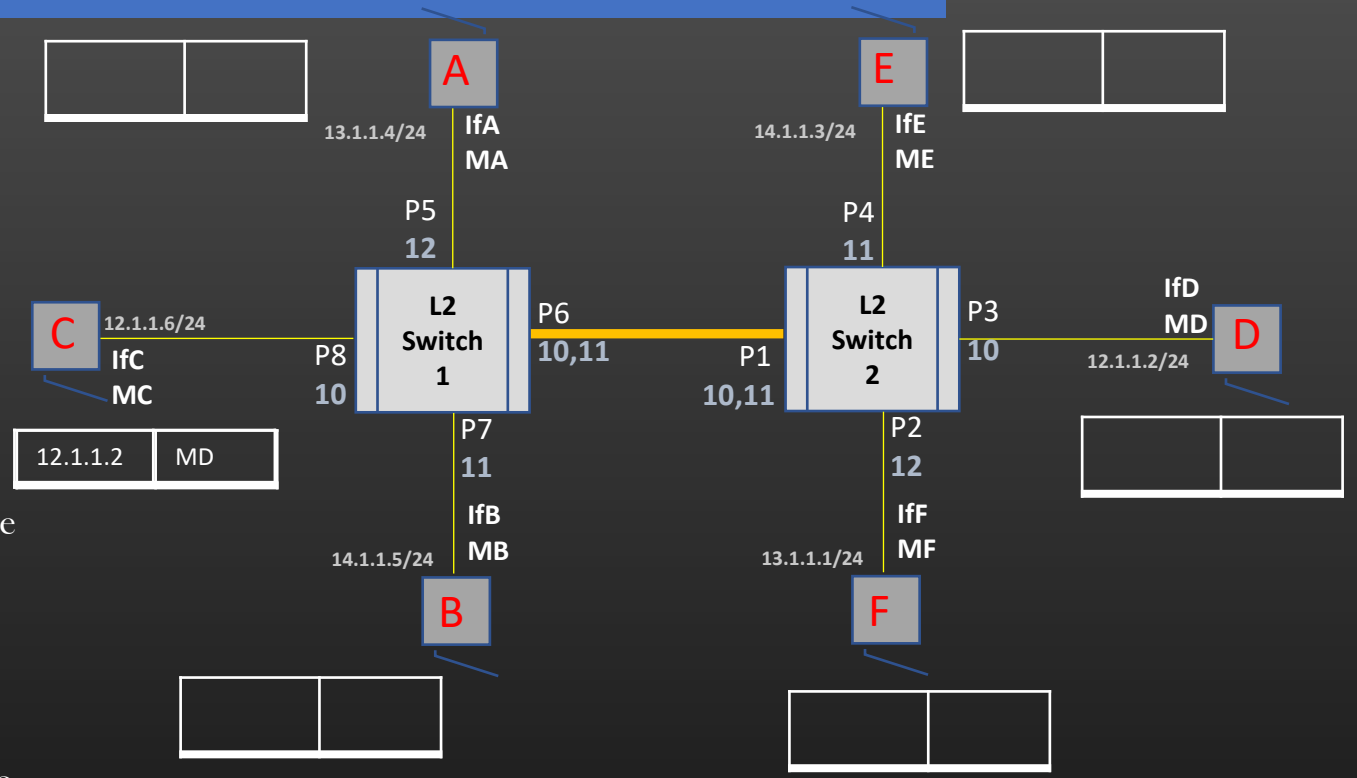
MAC table 1

Vlan ID	Mac address	Outgoing port no
10	MC	P1

MAC table 2

Example – Vlan based Routing

14. D receives ARP B. msg and prepare ARP reply
    ARP Reply : src mac = MD, dst mac = MC, dst ip = 12.1.1.6
15. L2 switch 2 receives ARP reply on interface p3, tag it with vlan id 10
16. L2 switch 2 does MAC learning : insert 10, MD, P3
17. L2 switch 2 checks its Mac table for entry : 10, MC
18. L2 switch 2 forwards the frame out of interface P1 only
19. L2 switch 1 receives ARP reply on interface P6
20. L2 switch 1 does MAC learning : insert 10, MD, P6
21. L2 switch 1 checks its Mac table for entry : 10, MC
22. L2 switch 1 forwards the frame out of interface P8 only, untags the frame
23. C receives ARP reply and update its ARP cache :
    Insert 12.1.1.2, MD
    (ARP resolution Complete)
    (Now Actual Data transfer)
24. C now knows D's MAC, so C prepares the frame now
    dst mac = MD, src mac = MC, src ip = 12.1.1.6, dst ip = 12.1.1.2



Vlan ID	Mac address	Outgoing port no
10	MC	P8
10	MD	P6

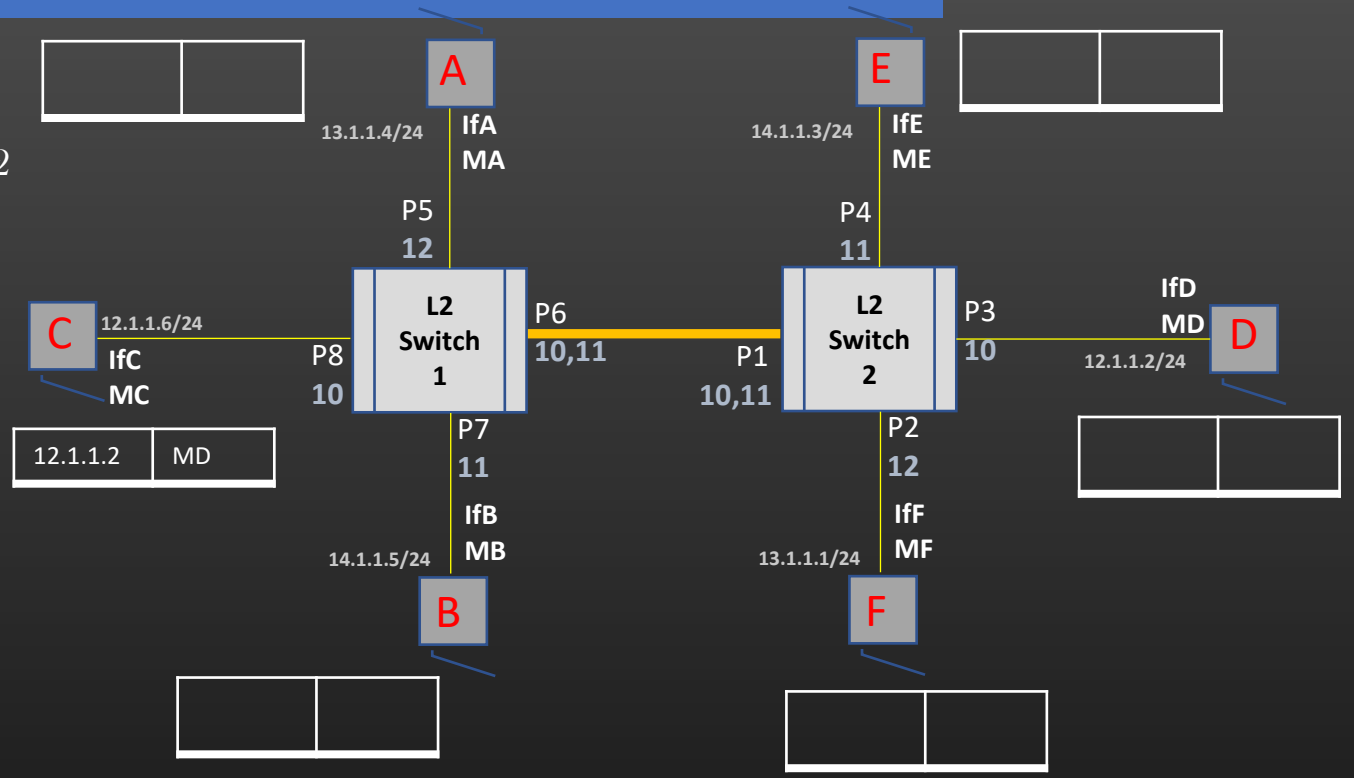
MAC table 1

Vlan ID	Mac address	Outgoing port no
10	MC	P1
10	MD	P3

MAC table 2

Example – Vlan based Routing

24. C now knows D's MAC, so C prepares the frame now
    dst mac = MD, src mac = MC, src ip = 12.1.1.6, dst ip = 12.1.1.2
25. L2 switch 1 recvs packet and tag it with vlan 10
26. L2 switch 1 does mac learning (refresh the entry)
27. L2 switch 1 checks its Mac table for entry : 10, MC
28. L2 switch 1 forwards the frame out of interface P6
29. L2 switch 2 recvs the frame on interface P1
30. L2 switch 2 does mac learning (refresh the entry)
31. L2 switch 2 check its mac table for entry : 10, MD
32. L2 switch 2 forwards the packet out of interface P3, frame gets untagged
33. D received the data
34. Complete !



Vlan ID	Mac address	Outgoing port no
10	MC	P8
10	MD	P6

MAC table 1

Vlan ID	Mac address	Outgoing port no
10	MC	P1
10	MD	P3

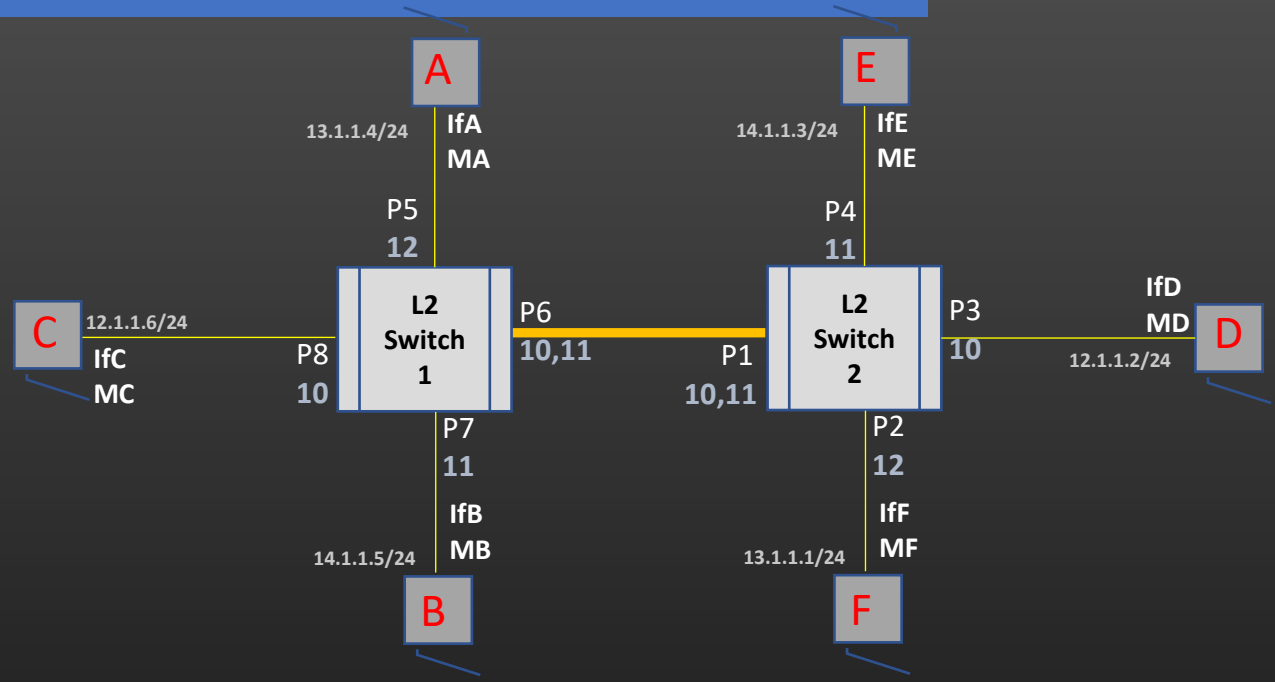
MAC table 2

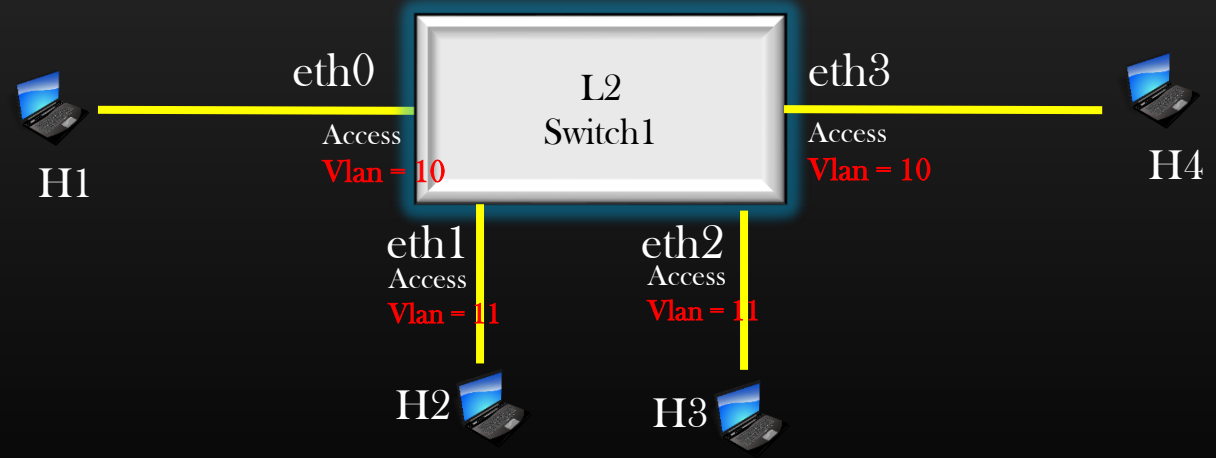
Vlan Benefits

Points to Note :

1. Hosts A,E,B and F are completely unaware of any communication taking place between C and D – *Segmentation*

Split one physical L2 switch into multiple logical switches





Physical Topology

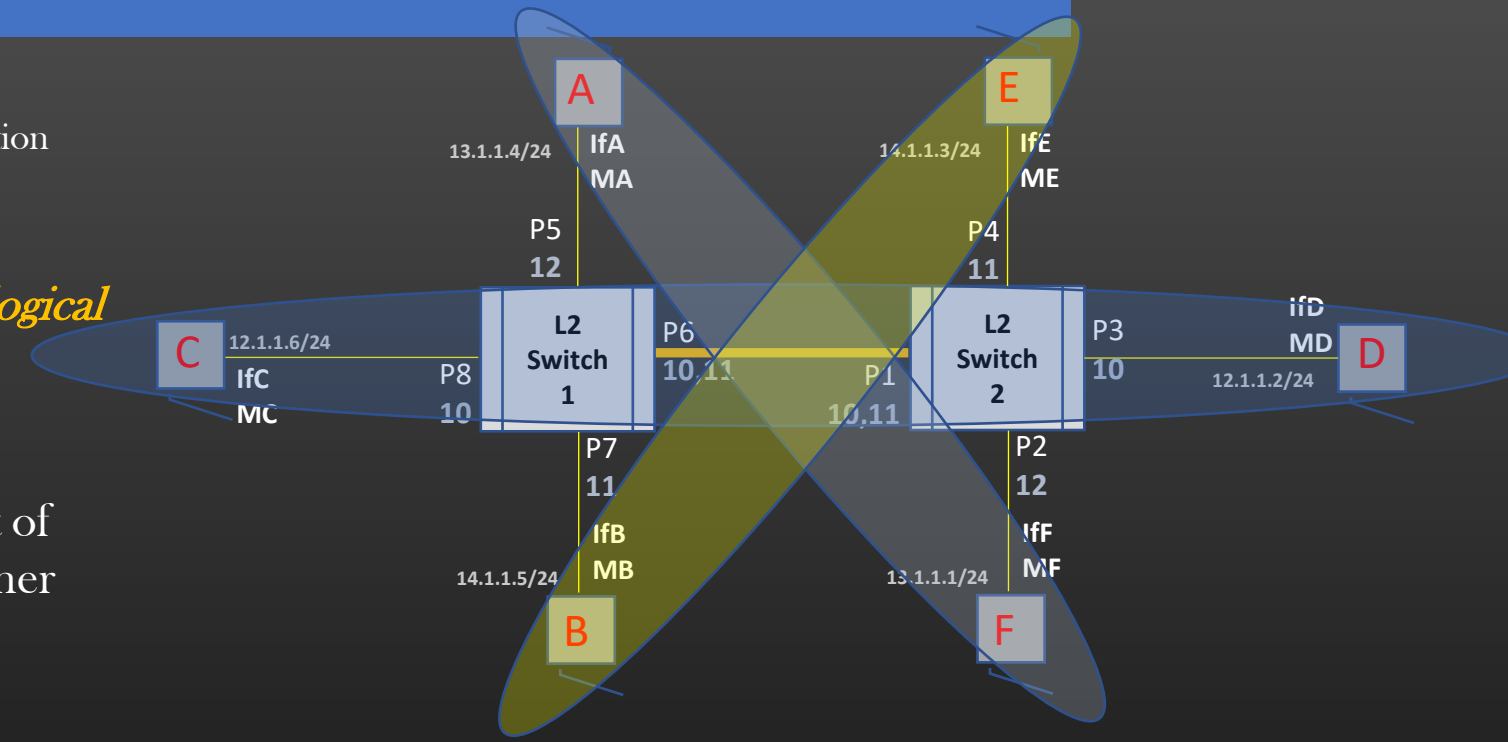
Other Vlans Benefits

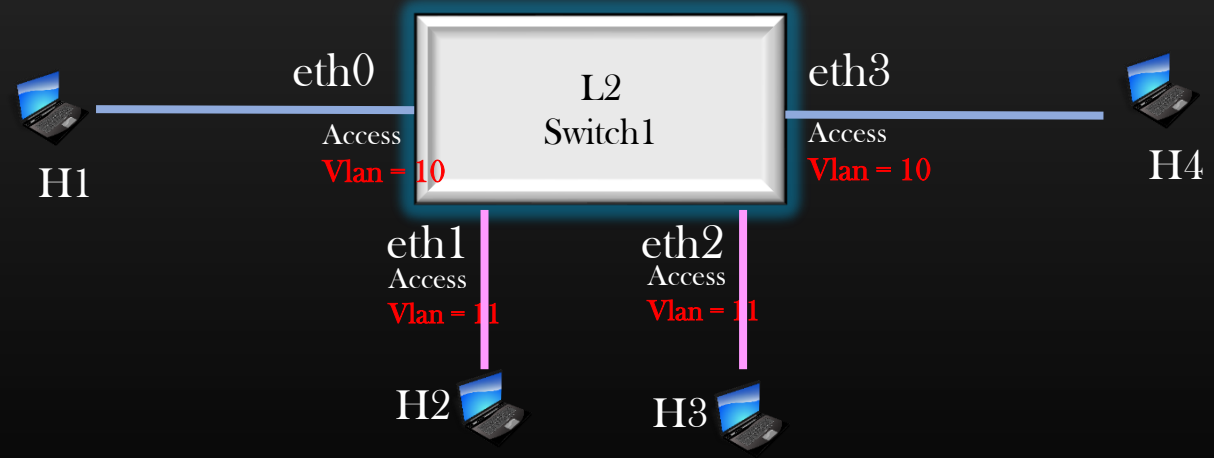
Vlan Benefits

1. Hosts A,E,B and F are completely unaware of any communication taking place between C and D – *Segmentation*

Split one physical L2 switch into multiple logical switches

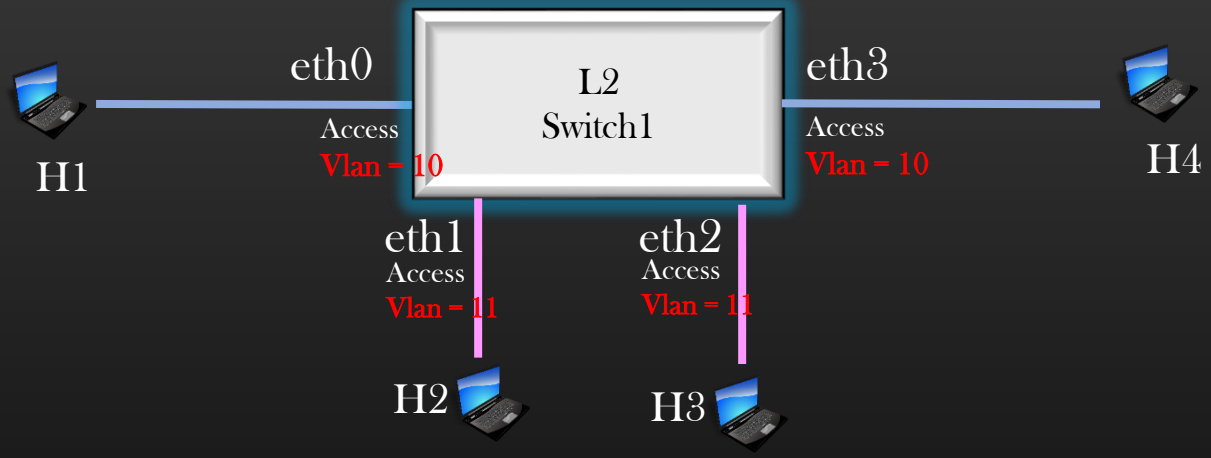
You would want the Finance and Research Dept of your company to work in isolation With each other !



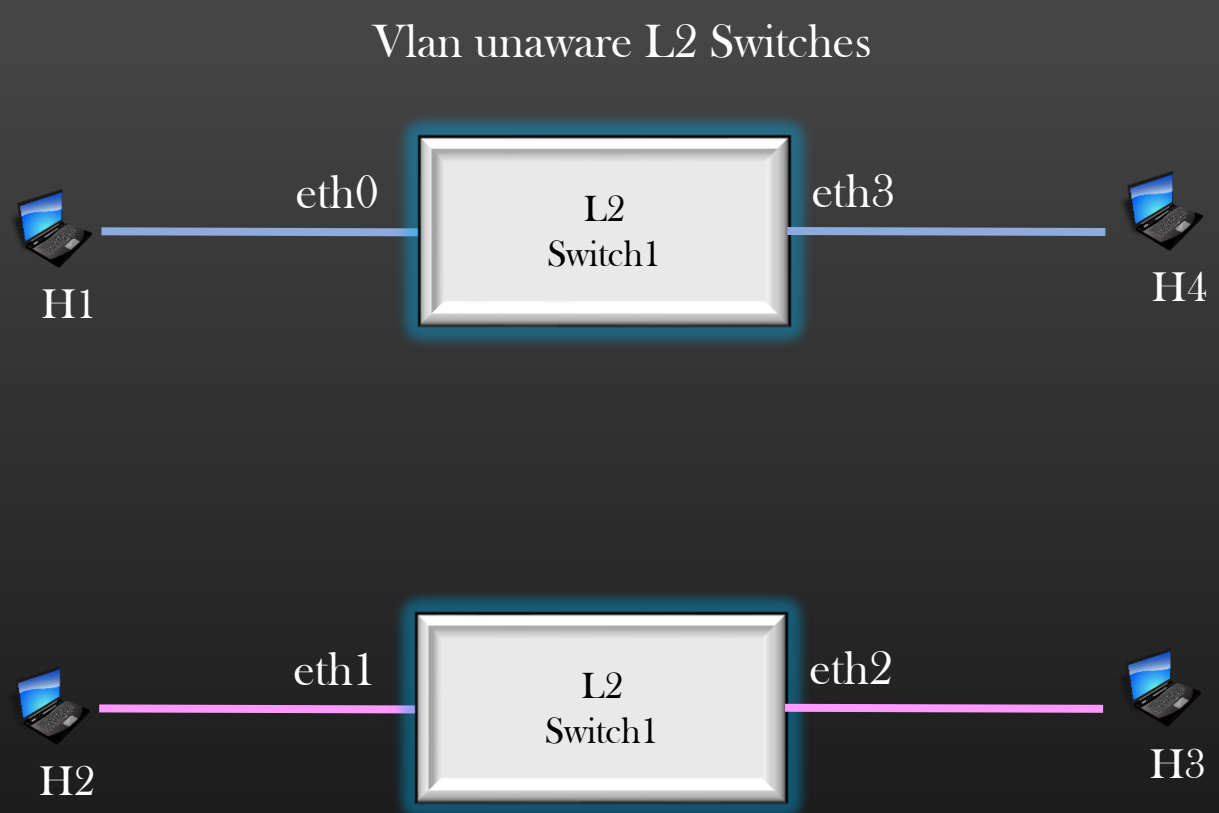


Logical Topology

Vlan Benefits



Logical Topology

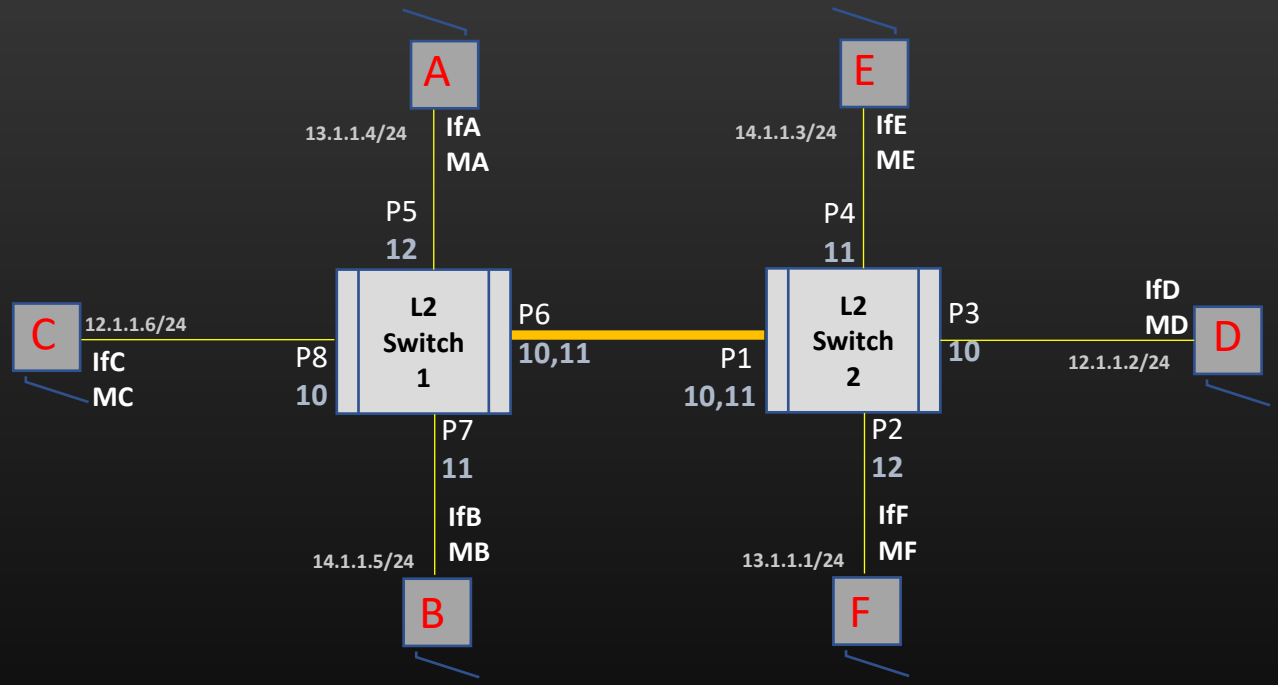


In this example, Vlans Split one physical L2 switch into 2 logical switches

Vlan Benefits

2. ARP broadcast msg is received only by hosts which are present in same subnet (vlan) as Source Host (C) - *Even more Granular solution to Thrashing !*

- In Normal Vlan unaware switches, Switches used to flood ARP B msg on all ports
- In Vlan Aware Switches, Switches Floods ARP B msg on all ports which are operating in same vlan

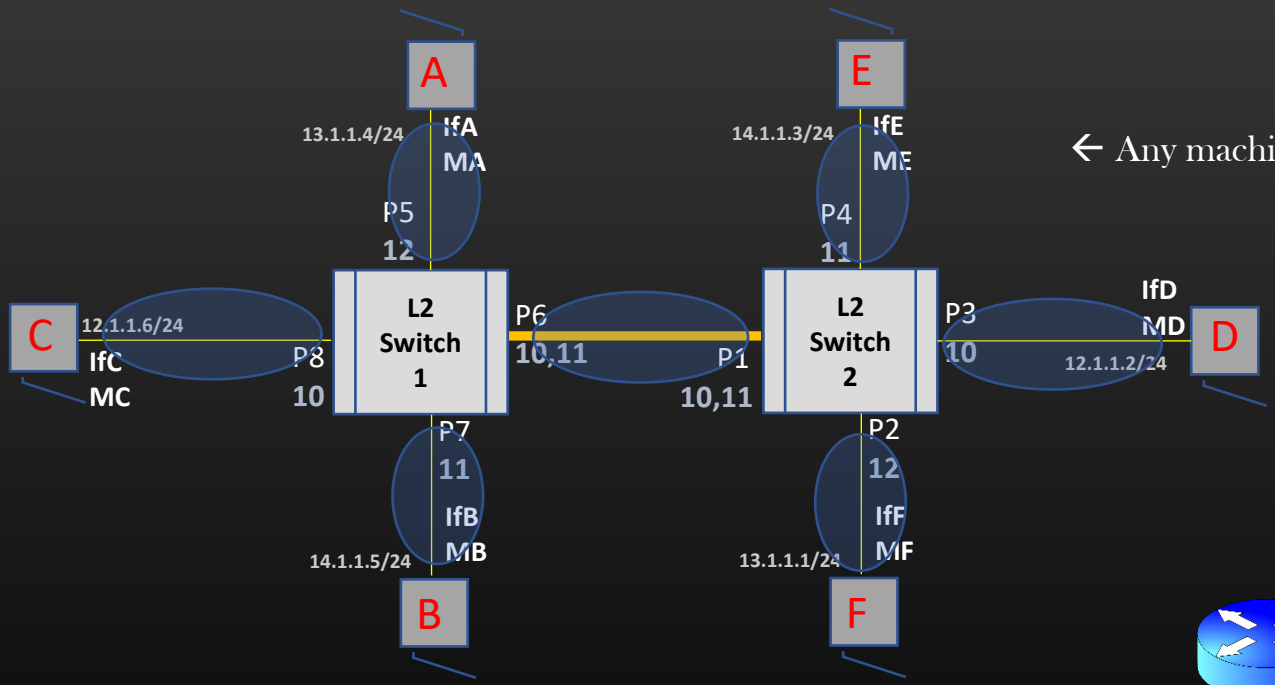


- Here, C shall always issue ARP B request msg for Host machine present in vlan 10 only, then why to bother host machines A, B, E and F !

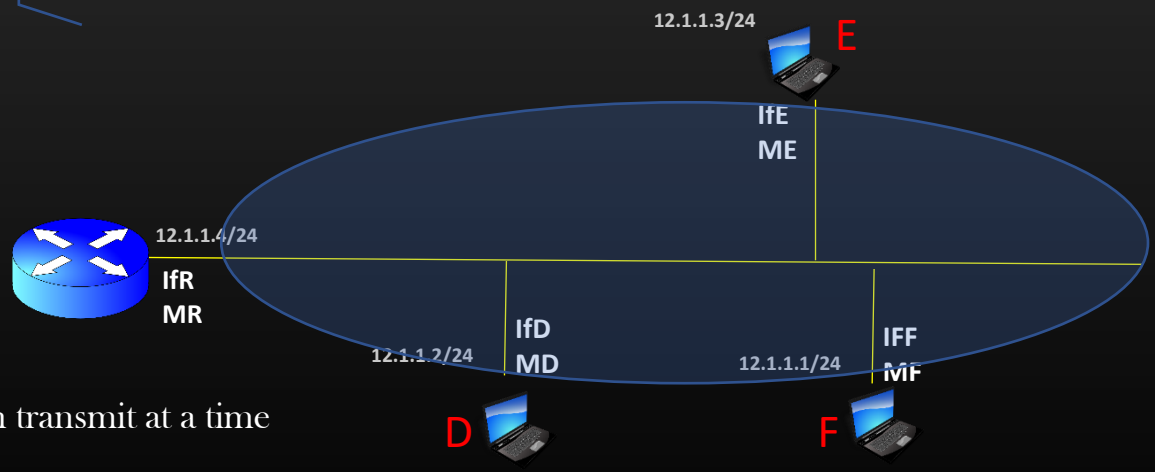
Vlan Benefits - deprecated

3. Communication between hosts in one vlan is completely isolated from hosts which are present in different vlans

- Security, Reduced Collision Domain



← Any machine can transmit any time !



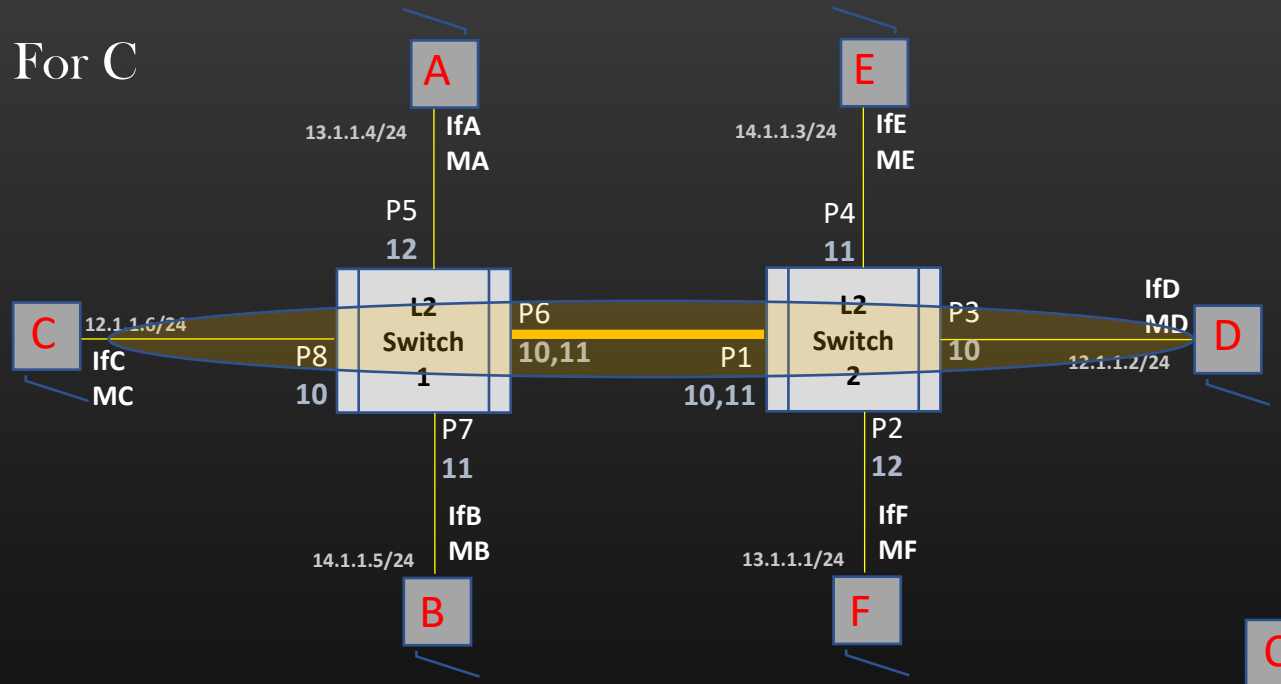
→ One of E,D, Router or F can transmit at a time

Vlan Benefits

3. Restricted flooding of ARP Broadcast message

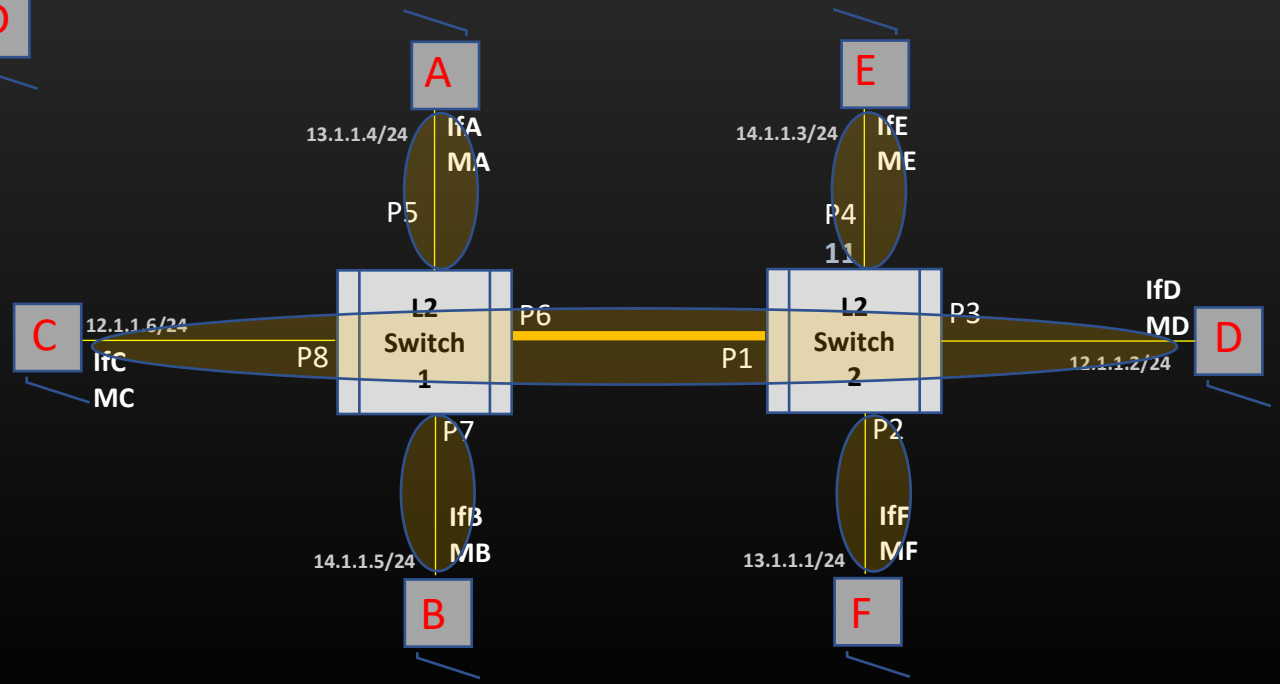
- Security, Reduced Broadcast Domain

For C



Smaller BD !!

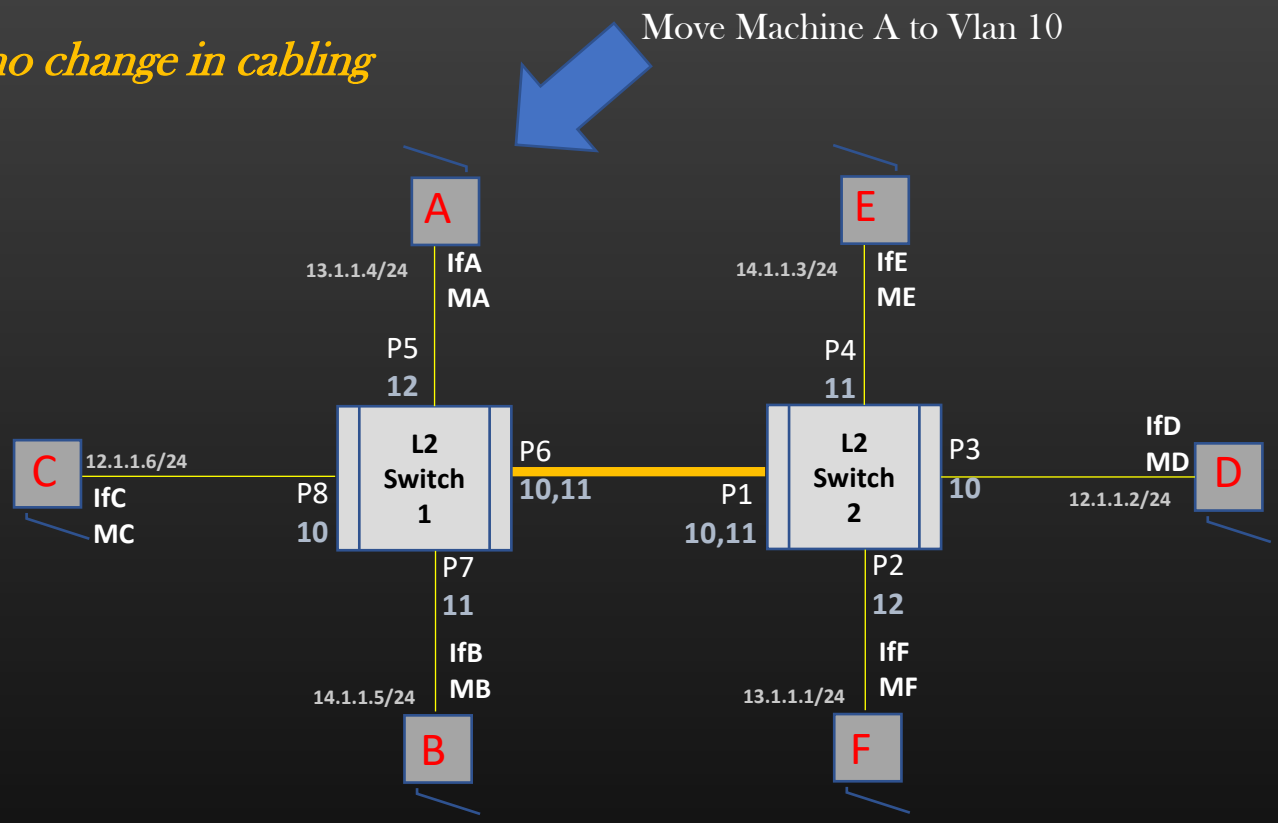
No VLANs !!
One Big BD



Vlan Benefits

4. To move the host A from vlan 12 to vlan 10, simply change the P5 port configuration and configure it under vlan 10. Change IP of A from 13.1.1.4/24 to 12.1.1.x/24

   - *Flexibility, no change in cabling*



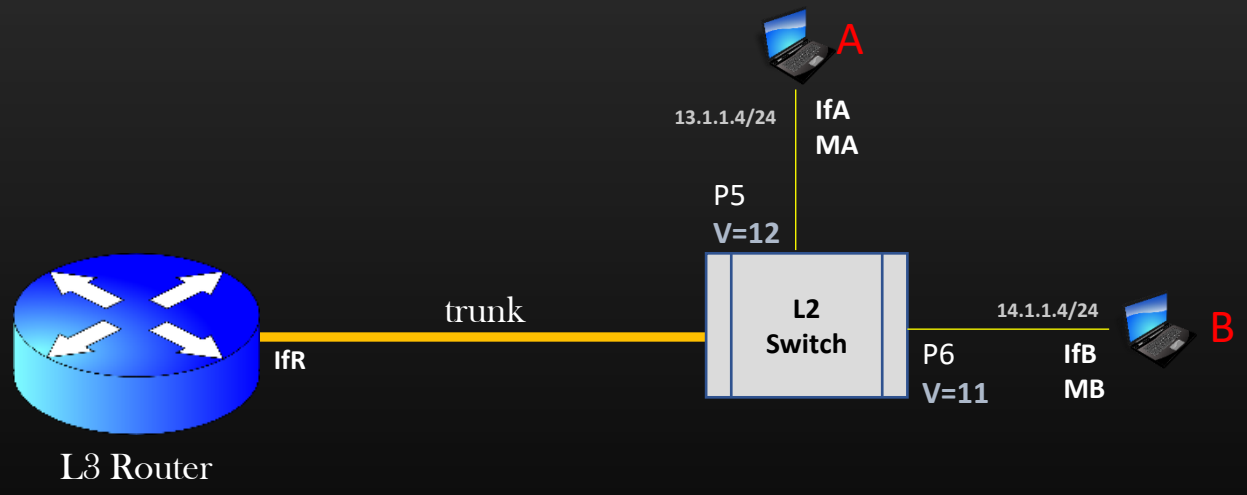
But how would Host C communicate with E ?
How to communicate Inter vlan ?

VLANS

Virtual Local Area Network

Thank you

Router -Vlan Routing



How Router will
Deliver the traffic to A ?

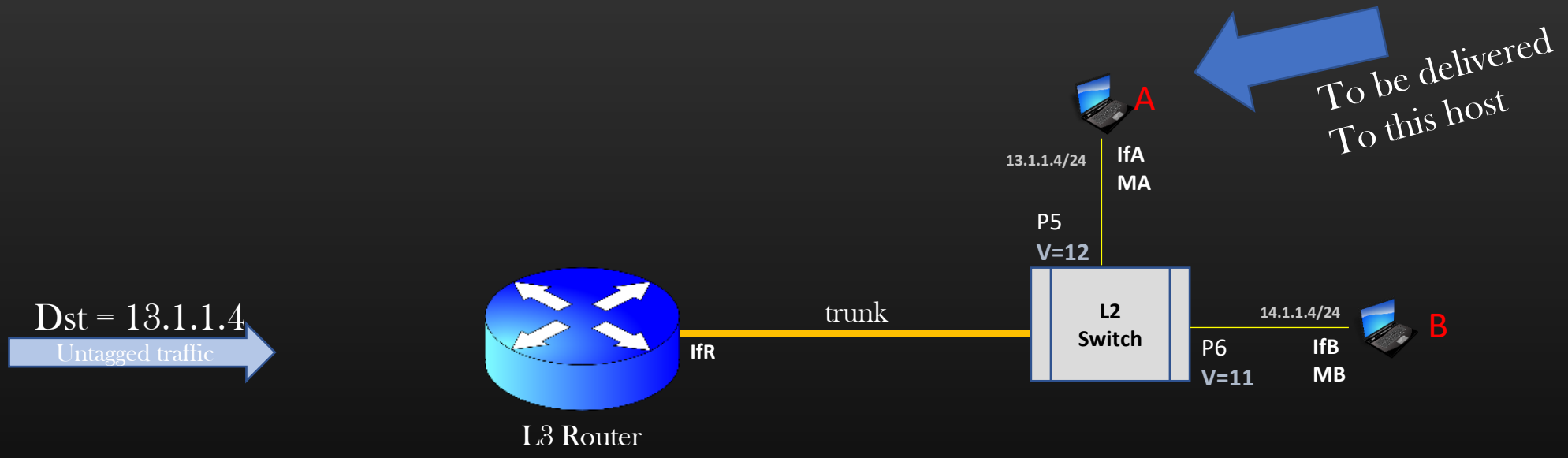
Router Vlan Routing

- Inter Vlan Routing allows Hosts machines present in different VLANs communicate with each other
- Vlans = Subnets, So stick to the basics
- When two communicating Machines present in different subnets wants to communicate, L3 router is always required
- L3 routers are the bridges which makes inter subnet communication possible
- In this Module, We shall understand how L3 router makes the Inter Vlan routing possible
- We shall learn the new concept without actually learning anything new !!
- You will understand what I Mean shortly !

Router Vlan Routing

- Vlans = Subnets, So stick to the basics

- In this Module, We shall understand how L3 router deliver a traffic received from outside (internet) to the host machine present in one of the VLAN



Router Possession

- The L3 router hosts three different subnets
  1. 11.1.1.0/24
  2. 10.1.1.0/24
  3. 12.1.1.0/24

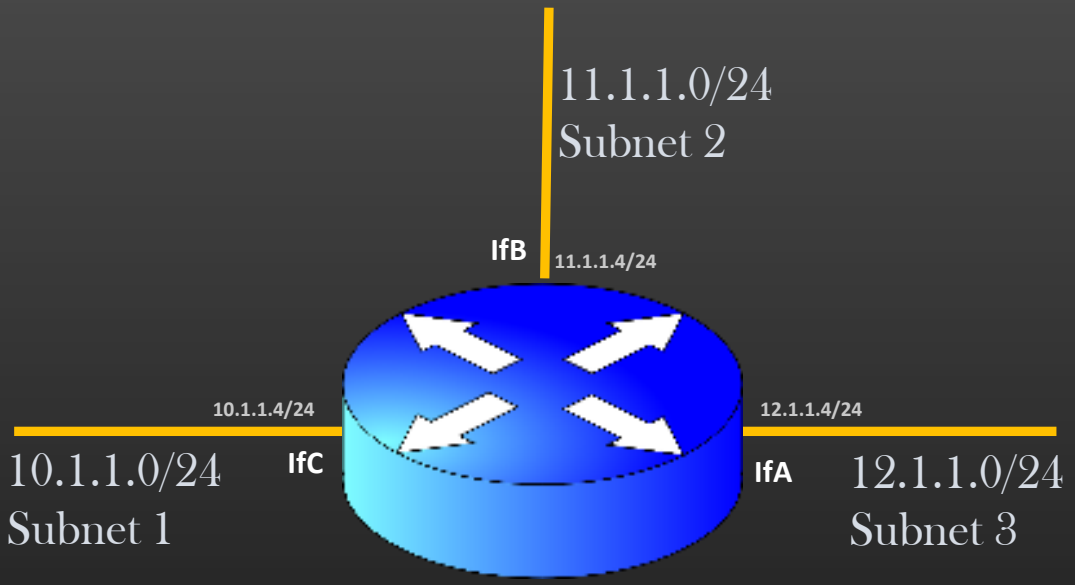
- In other words, we say the L3 router is in possession of these three subnets

- It means, this router should receive the traffic for Destination ip address = member of any subnets possessed by the router

- Now Let us assume, that this router has only three physical hardware interfaces, remember ip-address/mask are configured on interfaces only

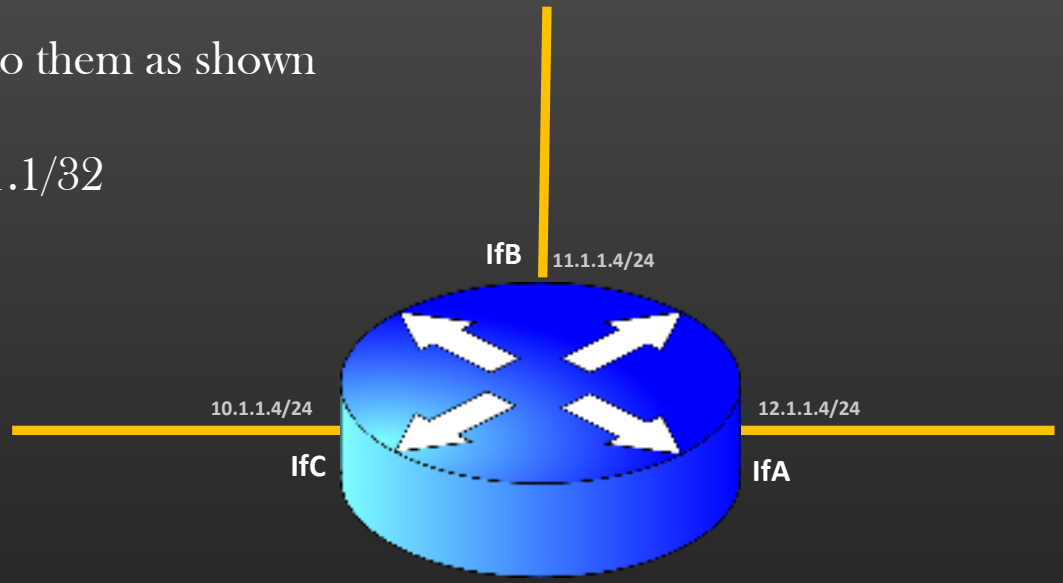
- Given the above limitation, what will you do to make this router posses more number of subnets ?

  Solution :
  - Create logical interfaces and assign ip/mask to them
  - These logical interfaces are given special name as Switch Virtual interfaces (SVI)
  - SVIs are not physical hardware, but just software based interfaces , like loopback interfaces only
  - So, create as many SVIs as you want !



Router Possession

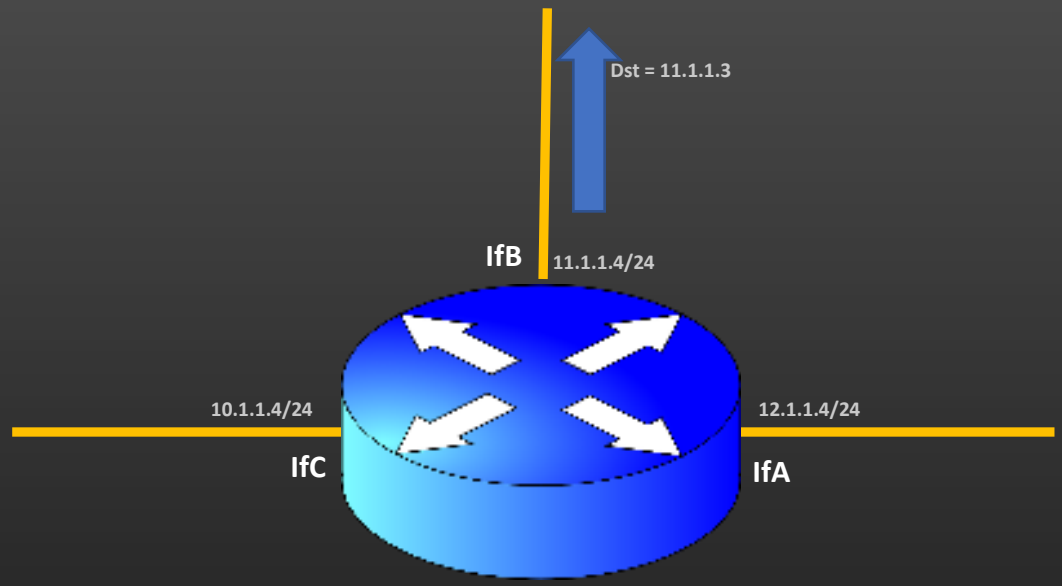
- Let us create SVI10, SVI20, SVI30 on Router and assign ip/mask to them as shown
- We have also created one loopback interface Lo1 with ip – 100.1.1.1/32
- Now router is said to be in possession of following subnets :
  1. 11.1.1.0/24
  2. 10.1.1.0/24
  3. 12.1.1.0/24
  4. 50.1.1.0/24
  5. 60.1.1.0/24
  6. 70.1.1.0/24
  7. 100.1.1.1/32



SVI 10 - 50.1.1.1/24
SVI 20 - 60.1.1.1/24
SVI 30 - 70.1.1.1/24
Lo1 – 100.1.1.1/32

- So, using the concept of SVIs, you can make a L3 router is possession of as many subnets as you want without attaching real physical interface/hardware
- Now this router must receive the traffic with destination IP address = member any subnet possessed By L3 router
- So, In addition to real physical subnets, this router would receive traffic destined for ip address 50.1.1.x, 60.1.1.x, 70.1.1.x Or 100.1.1.1 (self)

Router Possession



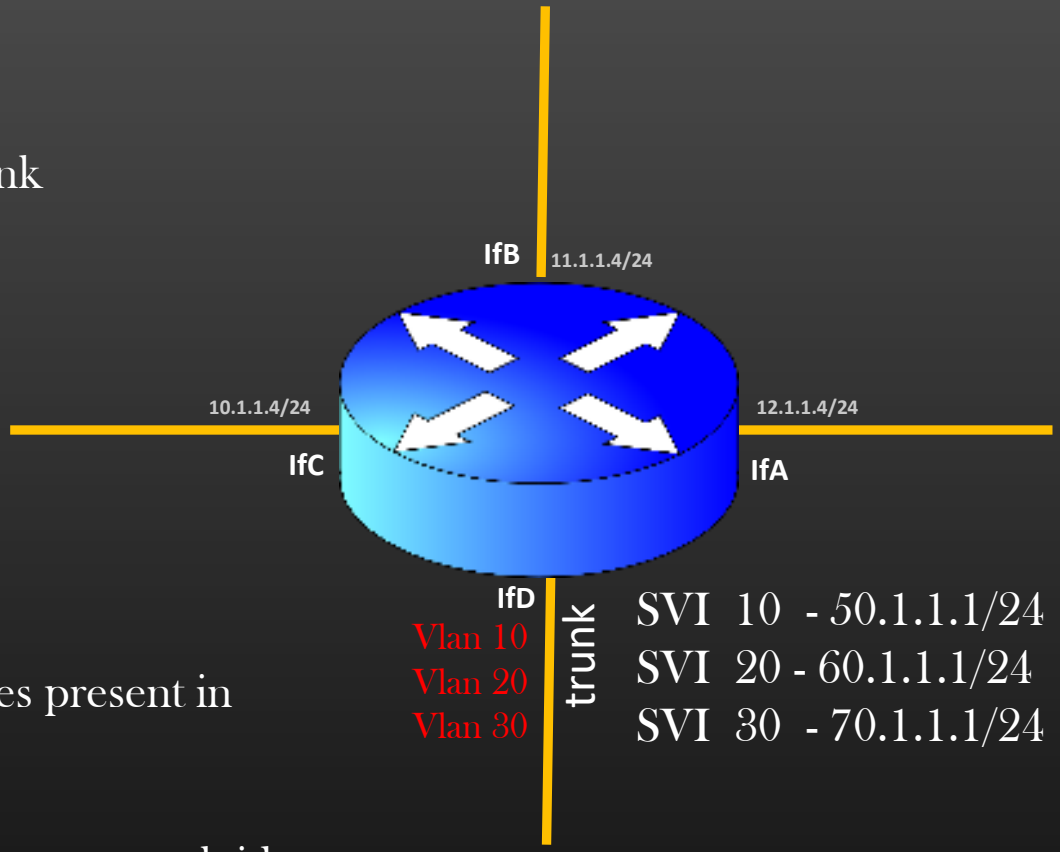
- When this router receives the traffic for Dst = 11.1.1.3, router shall forward the traffic out of interface ifB via L2 routing
- When this router receives traffic for Dst = 100.1.1.1 OR 11.1.1.4 Or 12.1.1.4 Or 10.1.1.4, router itself will consume the traffic (Exact match)
- But what will router do if it receives the traffic for Dst = 50.1.1.x , 60.1.1.x, 70.1.1.x where x != 1
- Router has to forward the traffic to machine present in directly connected local subnets
- But where are those subnets ?

SVI 10 - 50.1.1.1/24
SVI 20 - 60.1.1.1/24
SVI 30 - 70.1.1.1/24
Lo1 – 100.1.1.1/32

Router forwarding traffic on Vlans

Configuration

- Take one physical interface of a router, say ifD, and configure it as Trunk
- Configure ifD to operate in Vlan 10, 20 and 30
- Bind SVI interfaces to corresponding Vlans, one – to – one mapping
- By Binding the SVI – VLAN together, Vlans borrow the network-id of SVIs
- For example, VLAN 10 network id will be 50.1.1.0/24. All host machines present in VLAN 10 must be configured with ip address 50.1.1.x/24
- Remind basics – Every subnet must have network id , Vlans must also have network id since they are subnet after all



Binding :

VLAN 10 ←→ SVI 10

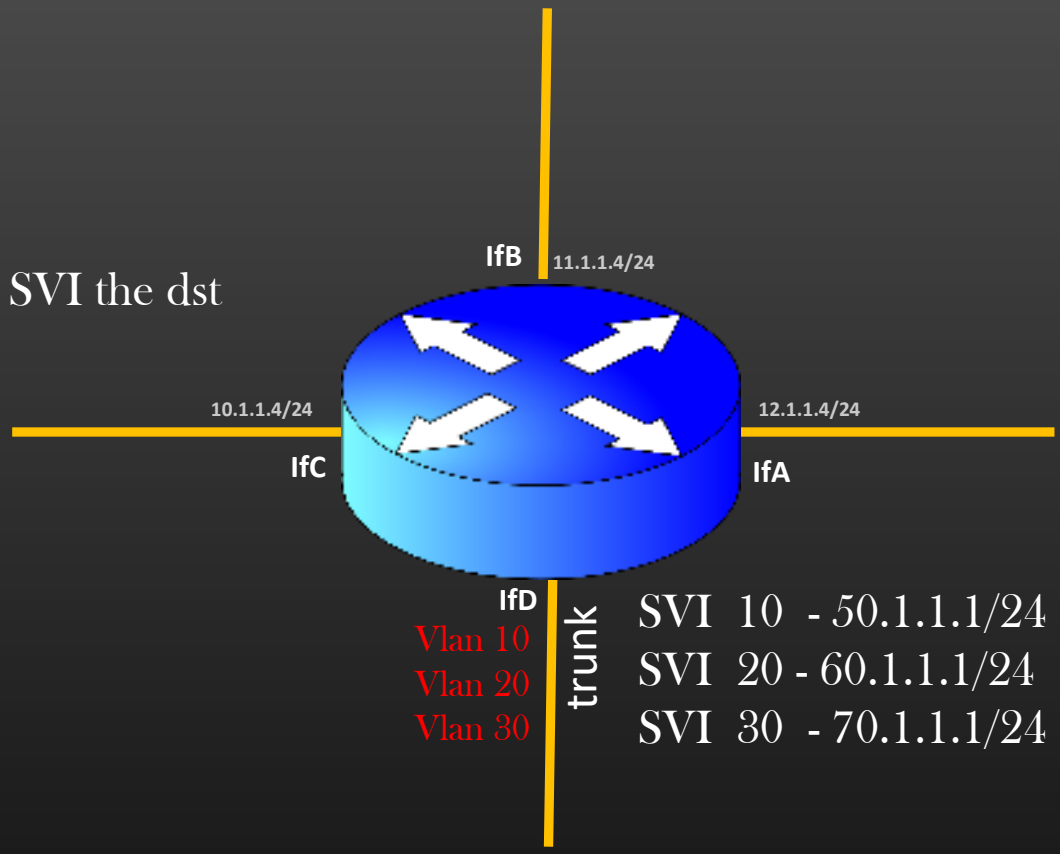
VLAN 20 ←→ SVI 20

VLAN 30 ←→ SVI 30

Router forwarding traffic on Vlans

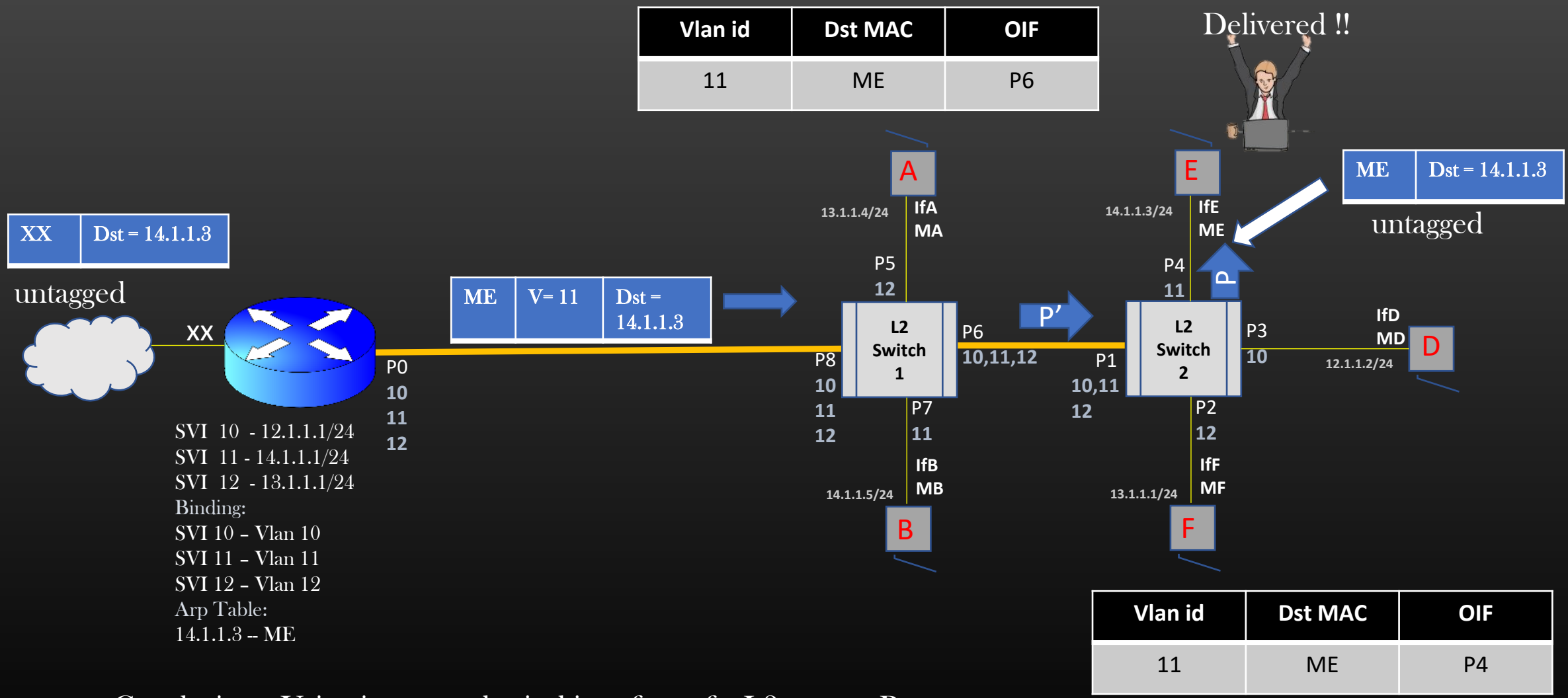
Steps for Router – Vlan Routing

- Step1 : if Router receives a traffic for Dst = 60.1.1.10, it checks in which SVI the dst ip address is a member of ,
  in this case it is SVI 20 (60.1.1.10 lies in subnet 60.1.1.0/24)
- Step 2 : Router then checks the Vlan bind to SVI 20 , which is Vlan 20
- Step 3 : Router tags the packet with vlan id 20
- Step 4 : Router forwards the packet out of all local physical interfaces which are operating in Vlan 20, in this case interface ifD only
- Now let us take an example to understand the concept . . . !



Binding :
Vlan 10 ←→ SVI 10
Vlan 20 ←→ SVI 20
VLAN 30 ←→ SVI 30

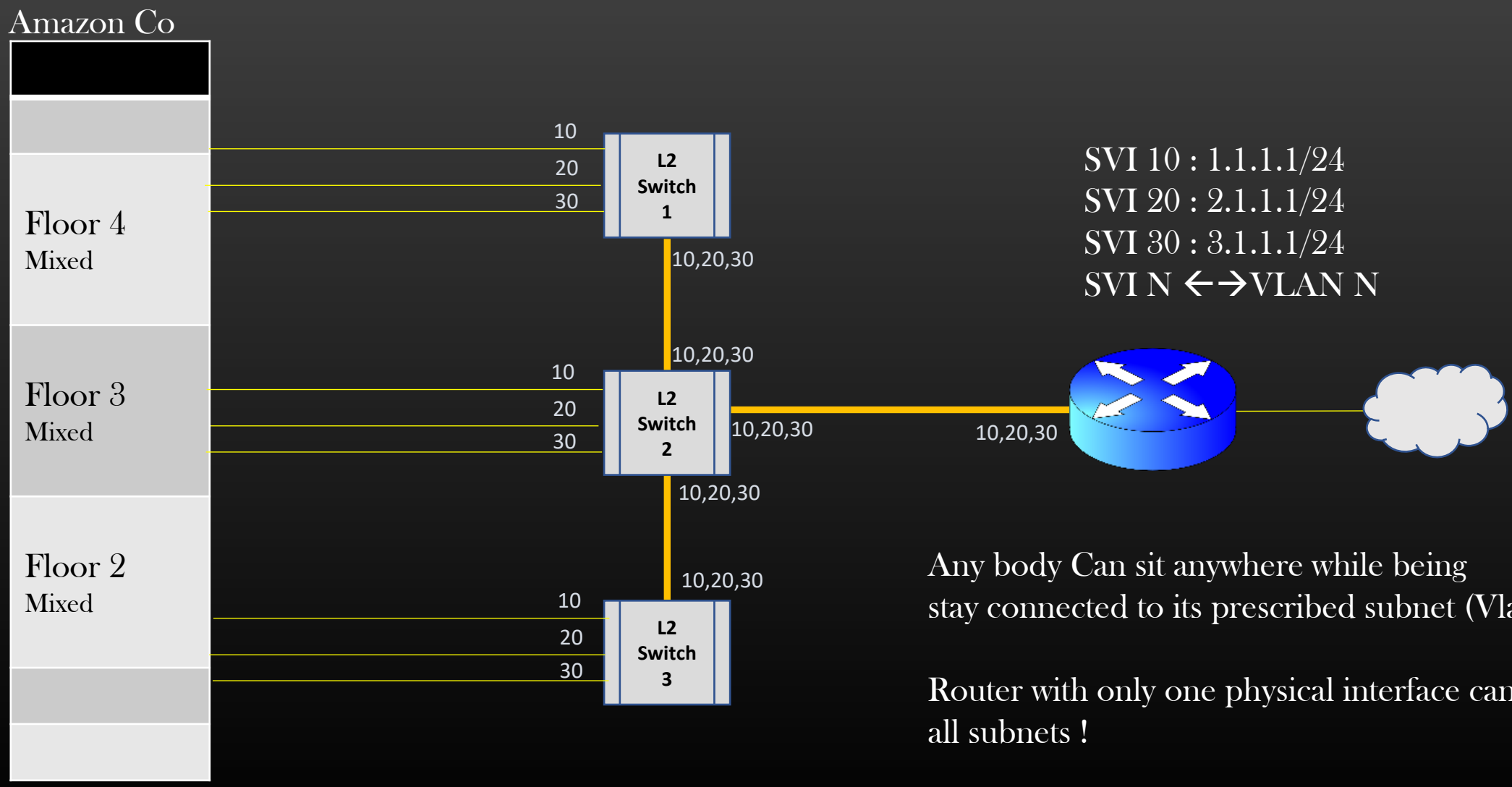
Router forwarding traffic on Vlans



Conclusion : Using just one physical interface of a L3 router, Router can Service multiple subnets using Vlans !!

Router forwarding traffic on Vlans

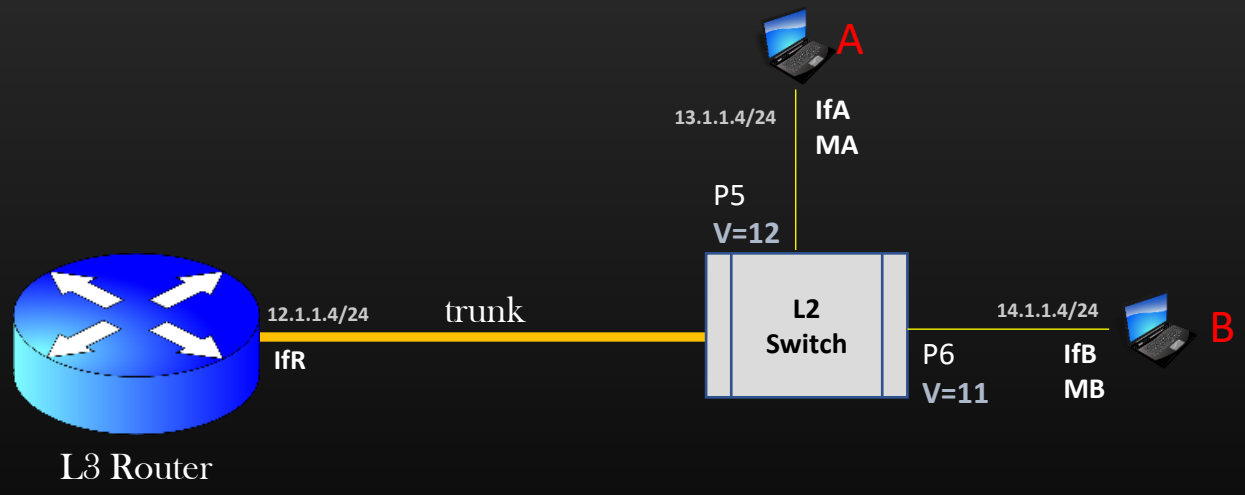
Vlans has resolved the Amazon's Problem of Immobility !



Any body Can sit anywhere while being stay connected to its prescribed subnet (Vlan) !

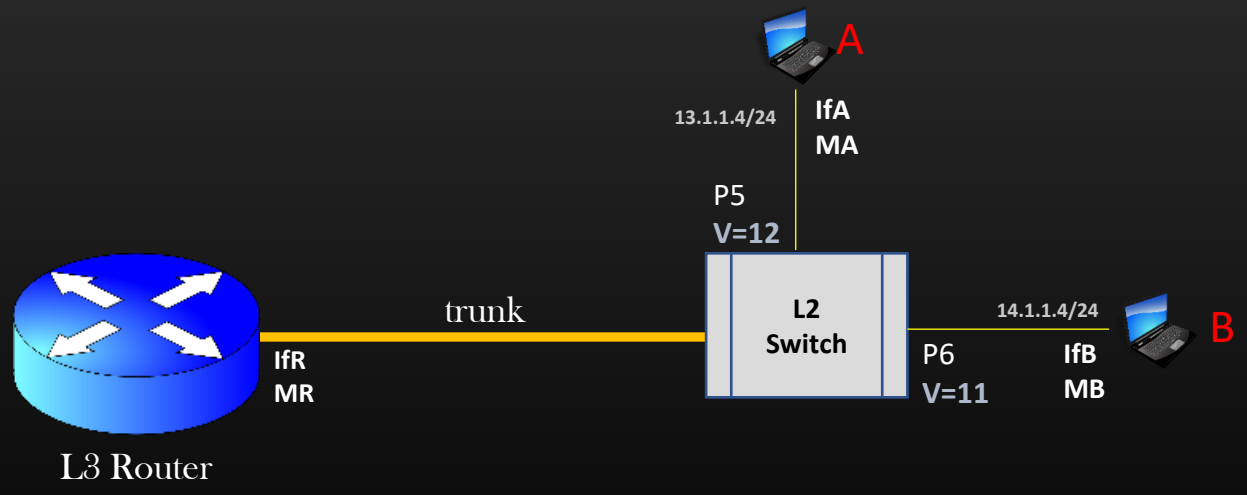
Router with only one physical interface can service all subnets !

Router -Vlan Routing



Thank you

Inter Vlan Routing



Problem Statement :
How Host A will talk to Host B ?

Inter Vlan Routing

Traditional L3 Routing Recap :

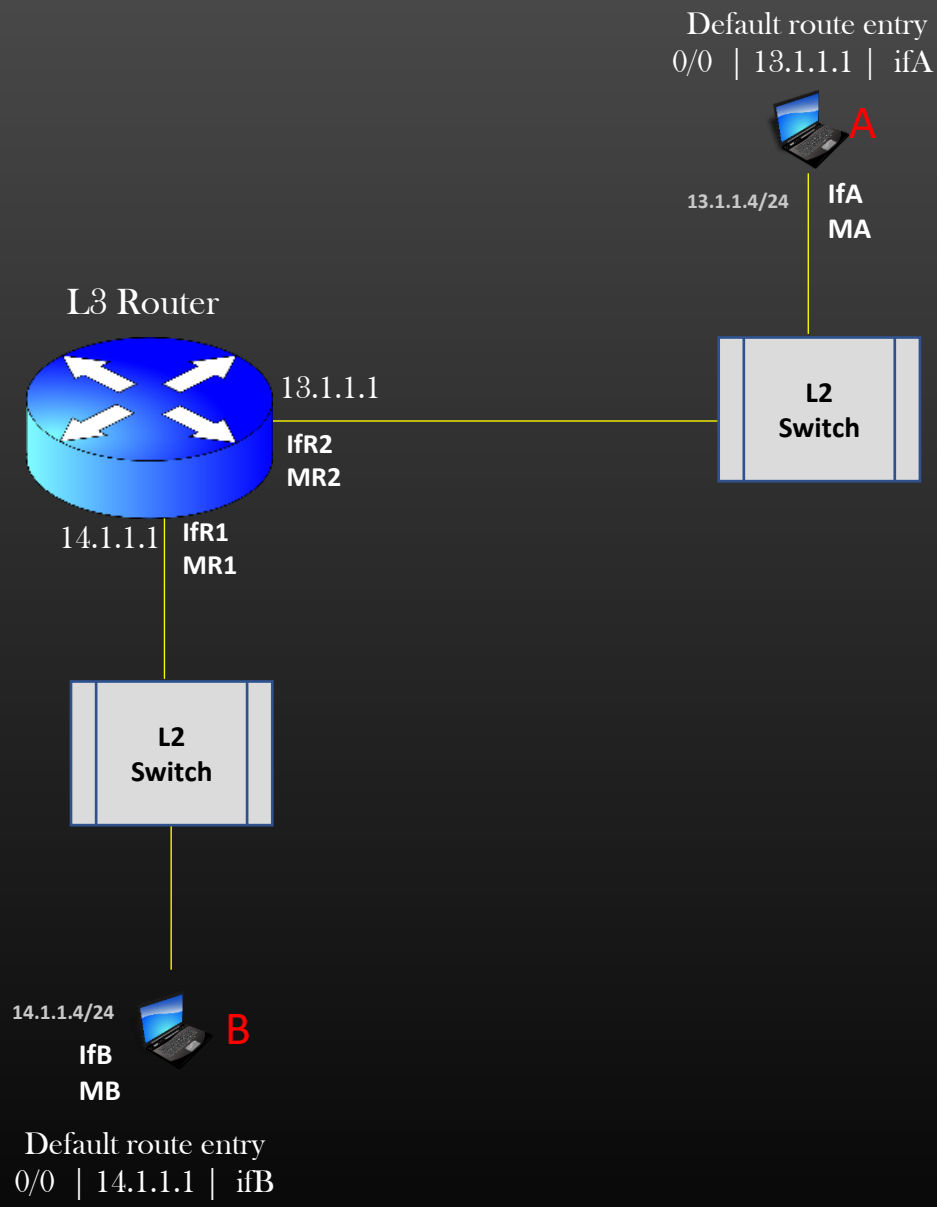
Whenever a host machine A needs to communicate with host machine B present in a remote subnet,
A sends frame to its gateway router using default entry In it's Routing table

Eg : if A wants to send a frame destined to 14.1.1.4, A has to send a frame to L3 router

To do so, Host machine A must know the MAC address for default gateway Ip i.e., it must have ARP entry 13.1.1.1 ←→MR2 In its ARP cache. Once A has this ARP mapping, A can send a Frame to L3 gateway router using dst mac = MR2 in ethernet hdr

We have discussed all this in traditional L3 routing already !

Go and Revise L3 Routing before proceeding forward !!☹



Inter Vlan Routing

- Similarly in inter Vlan routing , We need to know two things first :

1. Default Route

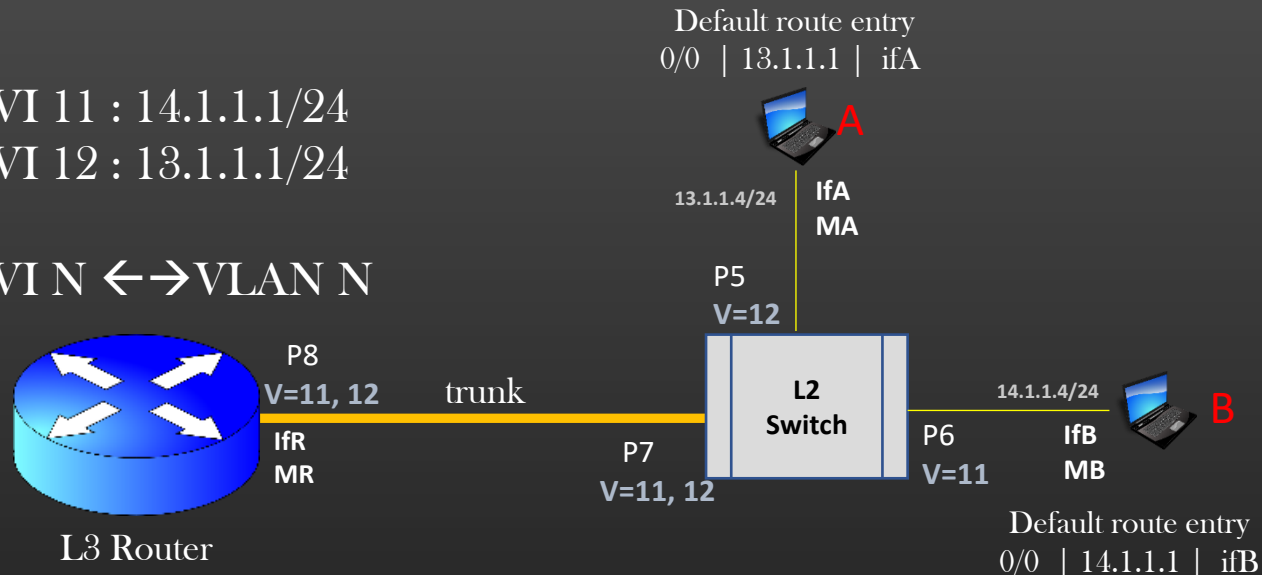
2. ARP Resolution for Default gateway ip

Inter Vlan Routing

Default Route

- To support inter-vlan routing,
  Every host machine in the network
  is configured with a *default route*

- Thus, default route of host machine =
  Ip address of SVI interface which is bind to VLAN
  on which host machine is present



- Thus, A's default route = 13.1.1.1 because A is in vlan 12 , and vlan 12 binds with SVI 12
  B's default route = 14.1.1.1 because B is in vlan 11 , and vlan 11 binds with SVI 11
  default route on host A's routing table is installed as : *0.0.0.0/0 | 13.1.1.1 | ifA*

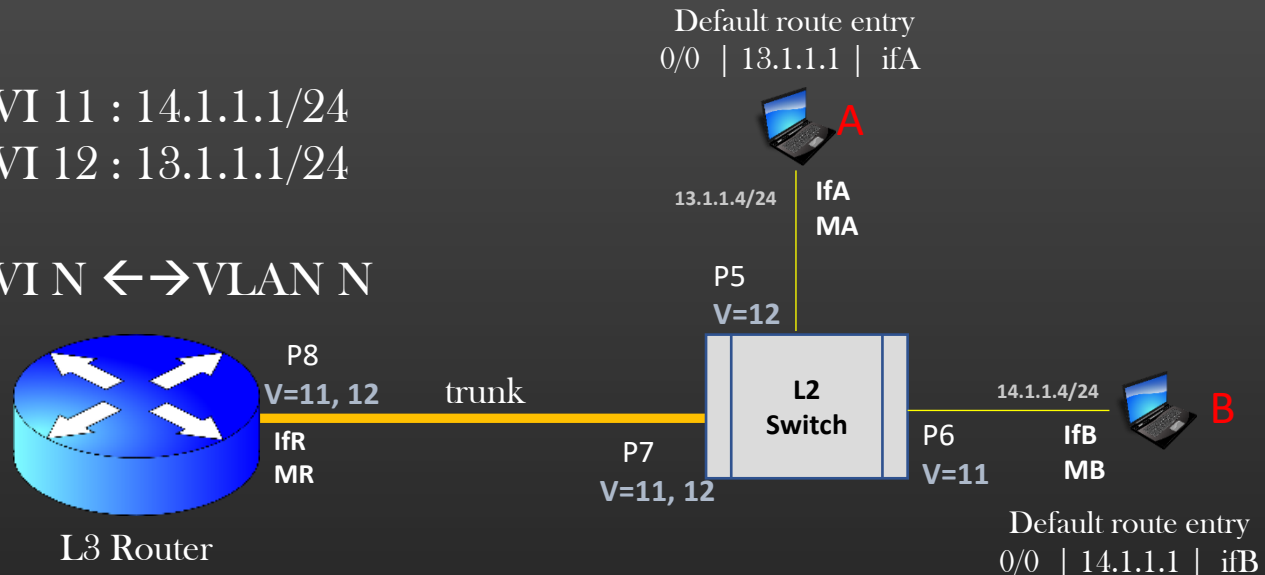
- Whenever the host needs to send a frame to dst machine which is outside its own subnet
  (vlan), host machine use a default route

Inter Vlan Routing

ARP Resolution for Default Routes

SVI 11 : 14.1.1.1/24
SVI 12 : 13.1.1.1/24

SVI N ←→VLAN N

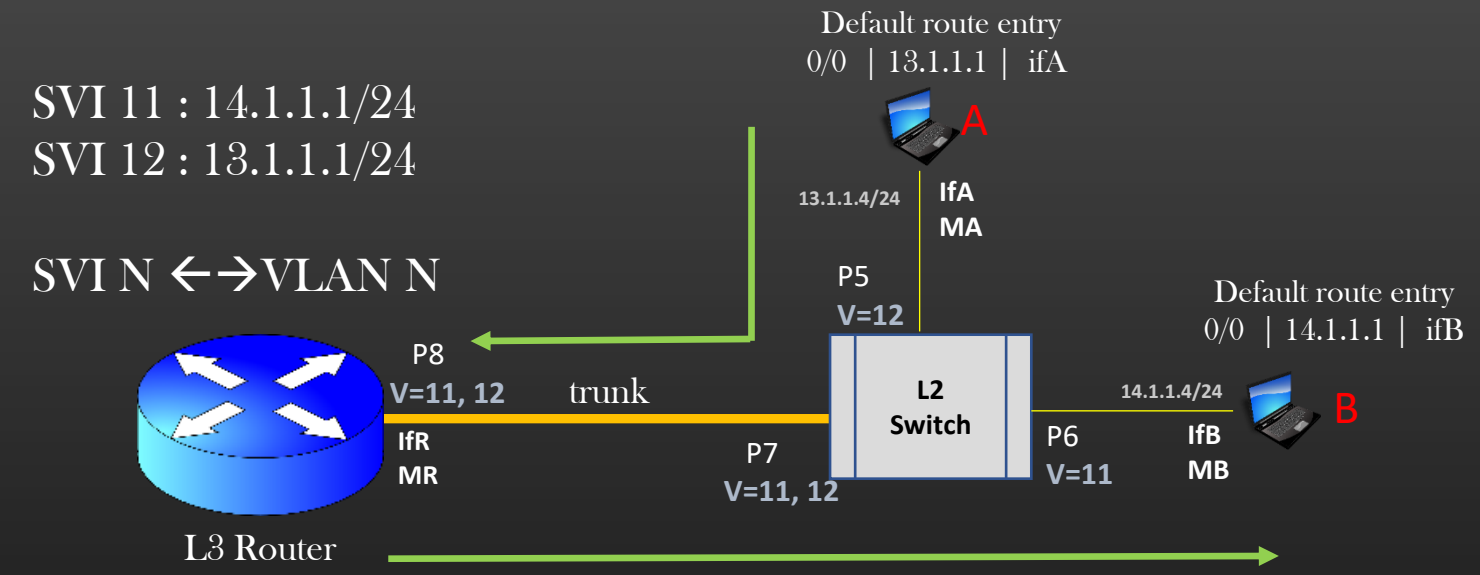


- Whose MAC address L3 router would return if it receives ARP B msg for ARP resolution of IP = IP address of SVI Interface?
- For example, Host Machine A issues ARP B msg to know MAC for default ip = 13.1.1.1
- 13.1.1.1 is the IP of SVI 12 , but SVI 12 is a logical interface, then whose MAC should L3 router must return in ARP reply ?
- Ans : MAC of physical interface on which ARP B msg is received
  In this case, MAC = MR will be returned in ARP reply
- Thus, L3 router returns MAC = MR in ARP reply for IP address = SVI's ip addresses (14.1.1.1 Or 13.1.1.1) since these are the ip address of one of Router's local interfaces (however logical interface) of L3 router

Inter Vlan Routing

Inter Vlan Routing Steps

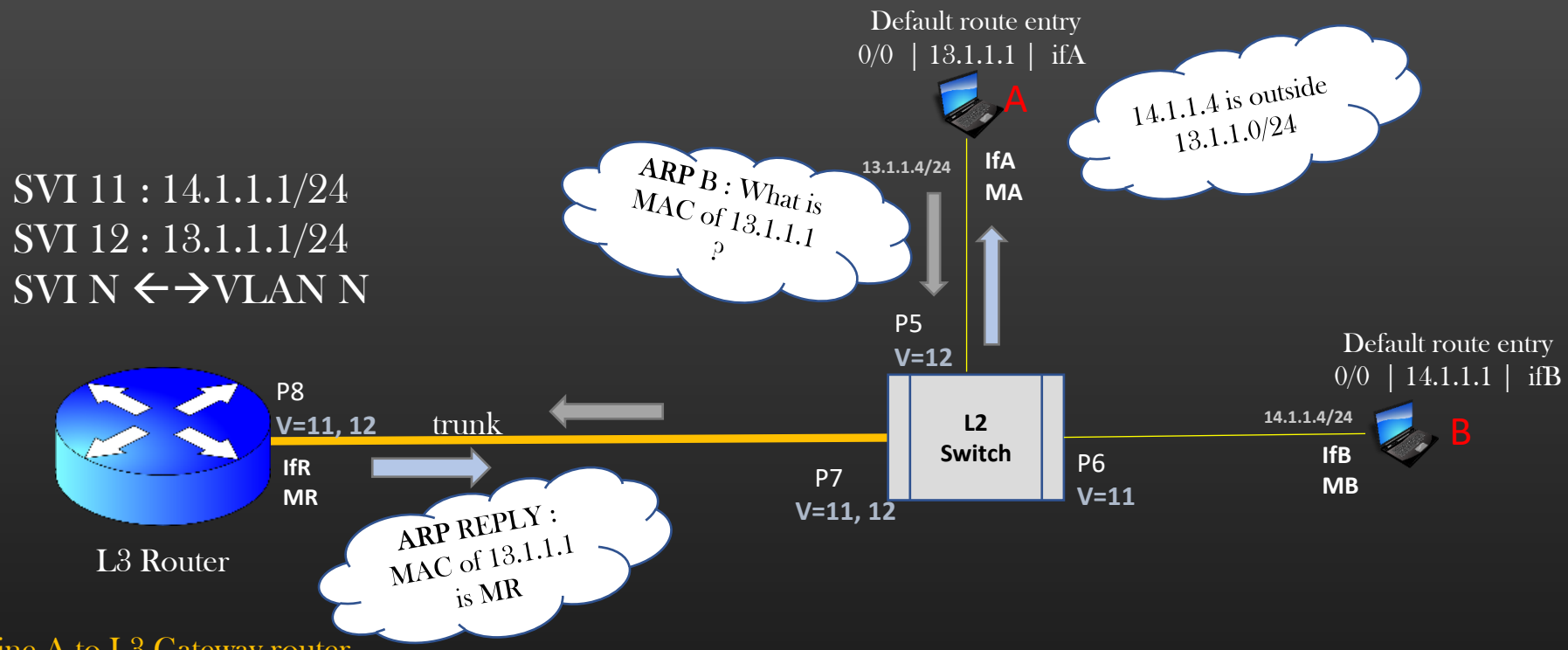
- Now, you have all the knowledge to do Inter vlan routing
- Suppose Host machine A wants to send data to host machine B whose ip address is 14.1.1.4
- Since, The data needs to go to remote subnet, there Host A (src) needs to send frame to gateway router first
- Gateway router then re-routes the frame to destination subnet



Always Remember, A frame/Packet cannot be transmitted from one subnet to another without L3 Router Involved !

Inter Vlan Routing

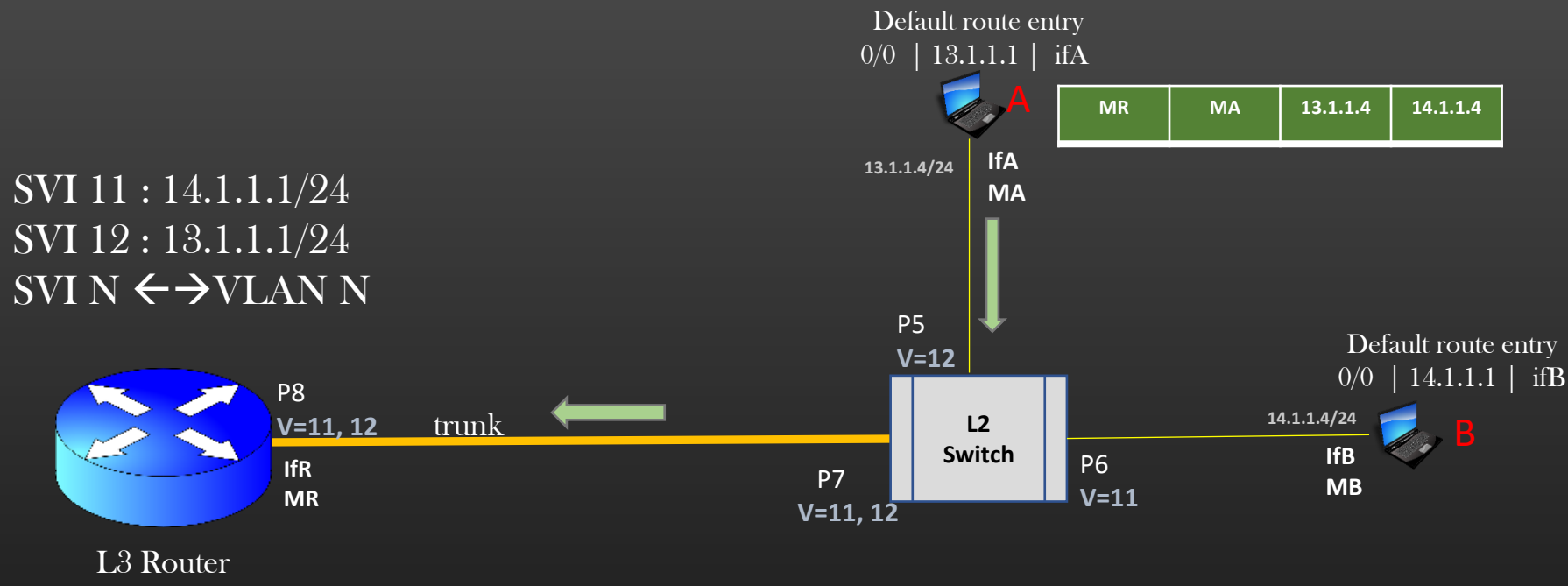
Inter Vlan Routing Steps



- Steps : From SRC Host Machine A to L3 Gateway router
  - Step 1 : A finds the B's ip = 14.1.1.4 belongs to remote subnet
  - Step 2 : A decides to send data using default route which states that *gateway Ip is 13.1.1.1, outgoing interface is ifA*
  - Step 3 : To send Data to gateway Router, A needs Mac of Gw Ip 13.1.1.1
  - Step 4 : A checks its ARP cache to resolve ARP for default gateway ip 13.1.1.1
  - Step 5 : A's ARP cache is empty, A launches ARP B msg out of ifA to know mac for 13.1.1.1
  - Step 6 : L3 routers returns MAC MR in ARP reply to A

Inter Vlan Routing

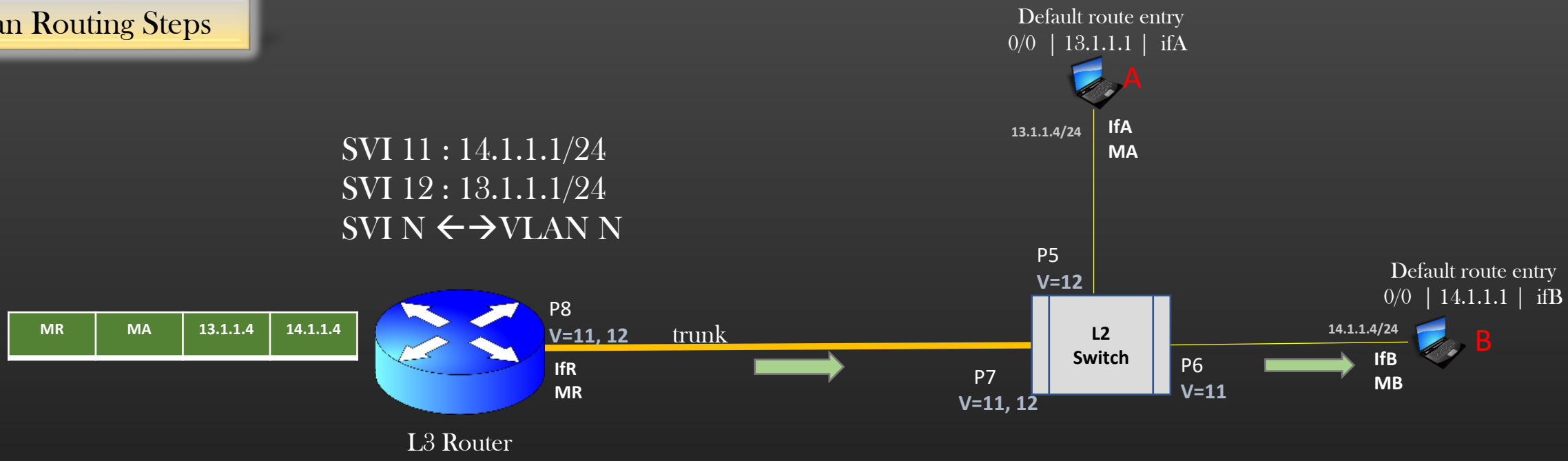
Inter Vlan Routing Steps



- Steps : From SRC Host Machine A to L3 Gateway router
  - Step 1 : A finds the B's ip = 14.1.1.4 belongs to remote subnet
  - Step 2 : A decides to send data using default route which states that *gateway Ip is 13.1.1.1, outgoing interface is ifA*
  - Step 3 : To send Data to gateway Router, A needs Mac of Gw Ip 13.1.1.1
  - Step 4 : A checks its ARP cache to resolve ARP for default gateway ip 13.1.1.1
  - Step 5 : A's ARP cache is empty, A launches ARP B msg out of ifA to know mac for 13.1.1.1
  - Step 6 : L3 routers returns MAC MR in ARP reply to A
  - Step 7 : A prepares the frame : Dst MAC : MR, Src Mac : MA, Src ip : 13.1.1.4, Dst ip : 14.1.1.4
  - Step 8 : This frame is received by L3 router only, and now L3 router needs to forward the frame to Host machine whose IP is 14.1.1.4
  - Step 9 : Now rest of the steps is same as that of – *L3 router – Vlan routing* which we learnt in Previous Module

Inter Vlan Routing

Inter Vlan Routing Steps



- Steps : From Gateway L3 router to DST host machine B
  - Step 10 : The frame received by L3 router is tagged with vlan 12
  - Step 11 : Router checks the dst ip 14.1.1.4 address in frame belongs to SVI 11's network id
  - Step 12 : Router finds the VLAN bound to SVI 11 – in this case VLAN 11
  - Step 13 : The Router interface(s) operating in Vlan 11 is P8 (ifR)
  - Step 14 : Router prepares the frame : Src mac : MR, dst MAC : MB, src ip : 13.1.1.4, dst ip : 14.1.1.4
    (Note : if router do not know the dst MAC MB, then it launches ARP B msg for dst ip = 14.1.1.4 on all interfaces operating in VLAN 11)
  - Step 15 : Router retags the frame from VLAN 12 to VLAN 11
  - Step 16 : Router sends out frame on interface P8
  - Step 17 : Host B receives the untagged frame

Inter Vlan Routing

Summary

1. For Host machines present in different VLANs to communicate, L3 router device is required

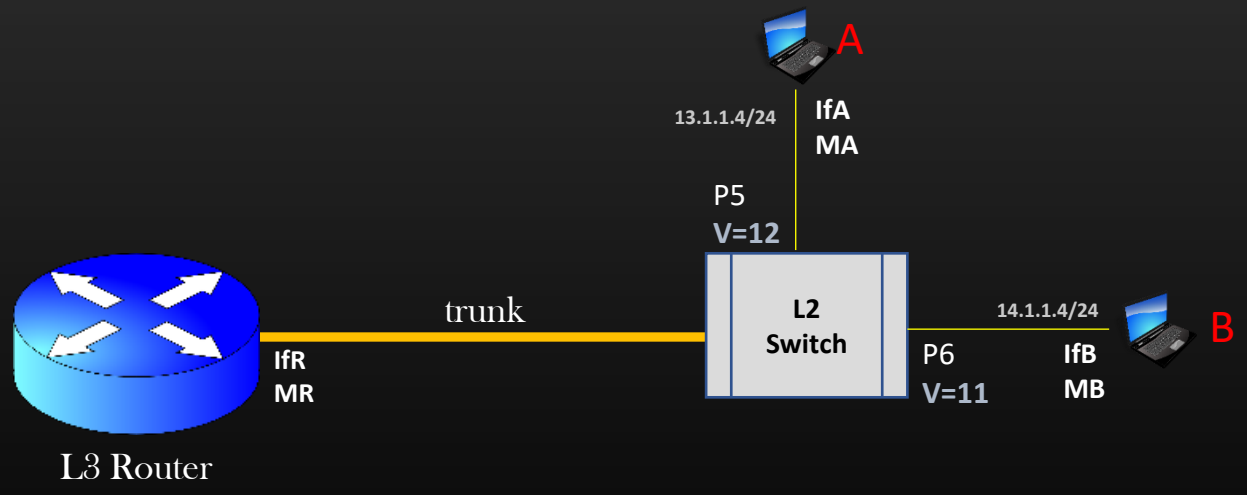
2. Remind basics – L3 router is required for routing the traffic from one subnet to another

3. L3 router retags the frame from Src VLAN to Dst VLAN in inter vlan routing

4. L2 switch functions as usual – Does MAC learning and forwards the frame based on tagged VLAN-id and Dst MAC address value in ethernet hdr of the frame

5. We have learnt three models of Vlan based routing :
   - From one host machine to another host machine in same Vlan
   - From L3 router to host machine present in a Vlan
   - From host machine present in one Vlan to host machine present in another vlan

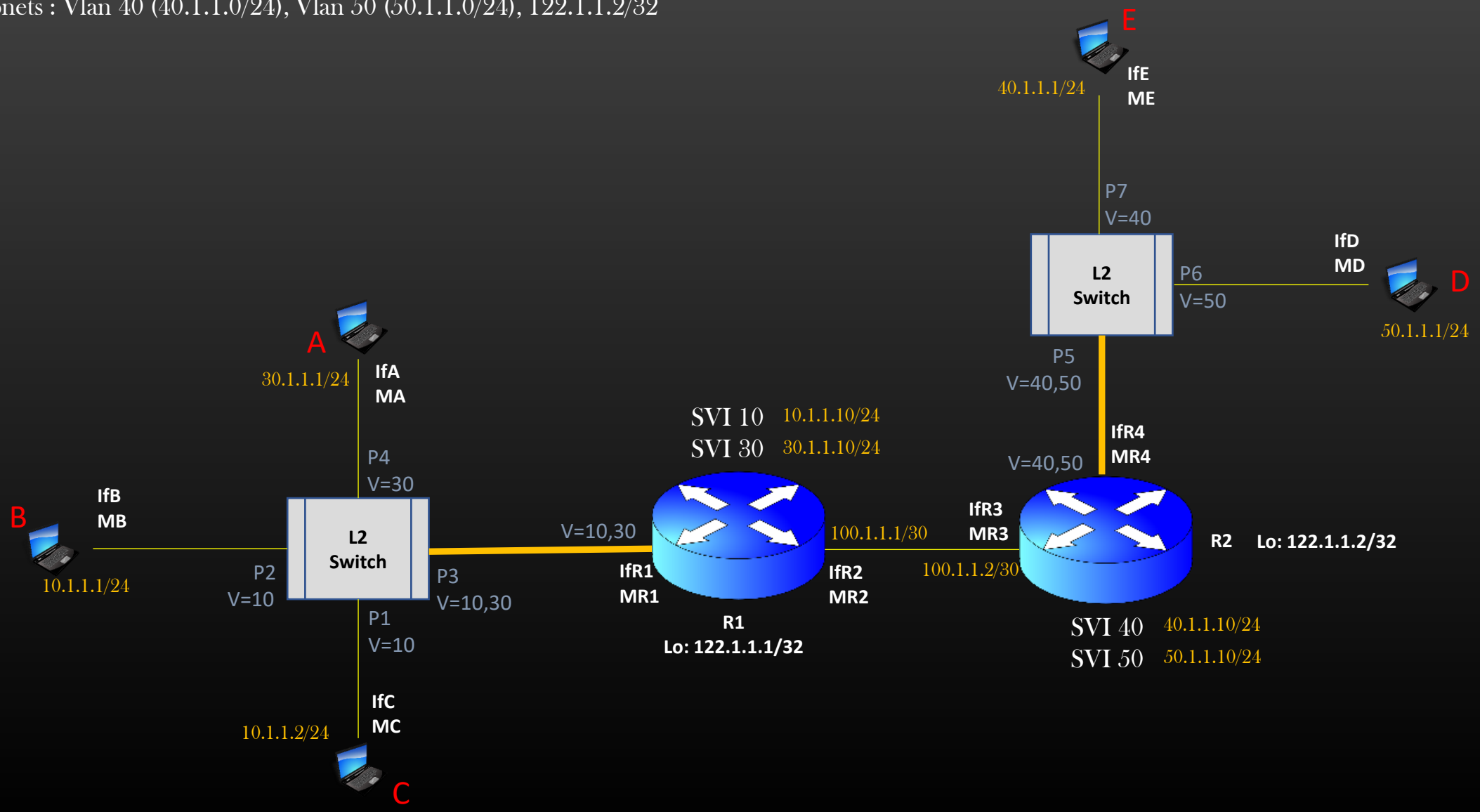
Inter Vlan Routing



Thank You

Exercise

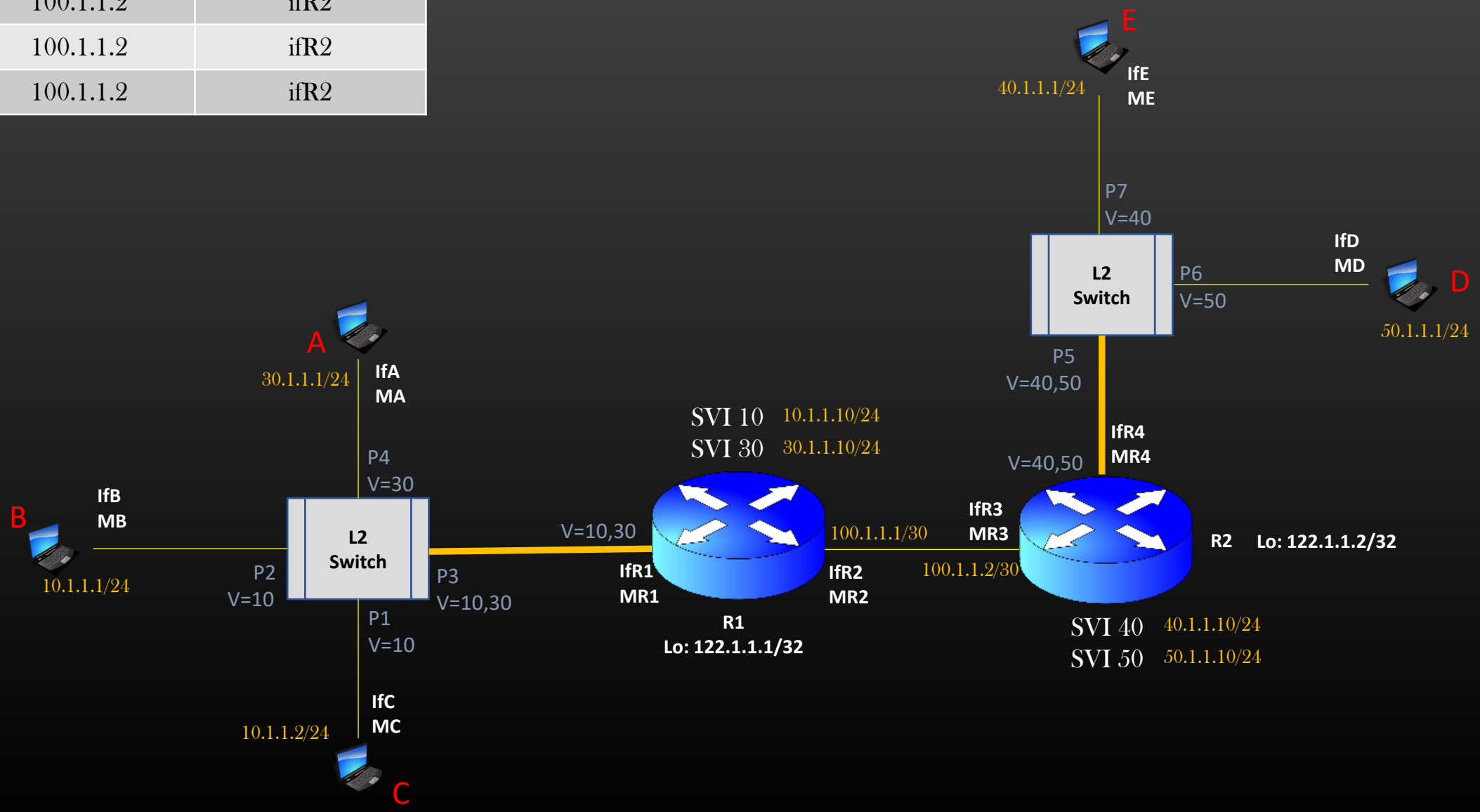
Q. How many Local and Remote subnets does router R1 have ?
Ans : Local Subnets : Vlan 10(10.1.1.0/24), Vlan 30 (30.1.1.0/24),
100.1.1.0/30, 122.1.1.1/32
Remote Subnets : Vlan 40 (40.1.1.0/24), Vlan 50 (50.1.1.0/24), 122.1.1.2/32



Exercise

Q. How should be the Routing table of R1 so that it can forward the Traffic destined to all remote subnets ?

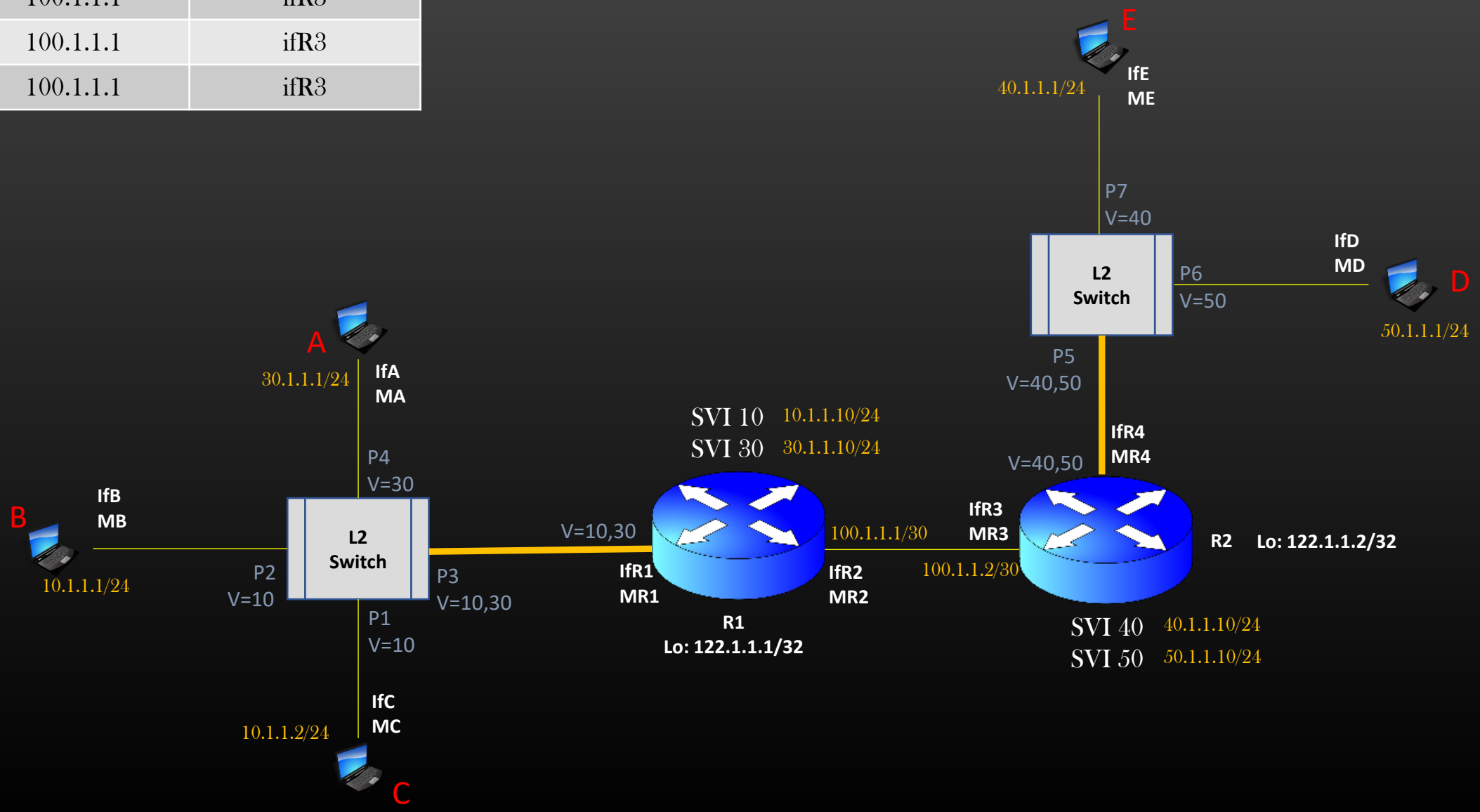
Destination Subnet	Gateway	OIF
40.1.1.0/24	100.1.1.2	ifR2
50.1.1.0/24	100.1.1.2	ifR2
122.1.1.2/32	100.1.1.2	ifR2



Exercise

Q. How should be the Routing table of R2 so that it can forward the Traffic destined to all remote subnets ?

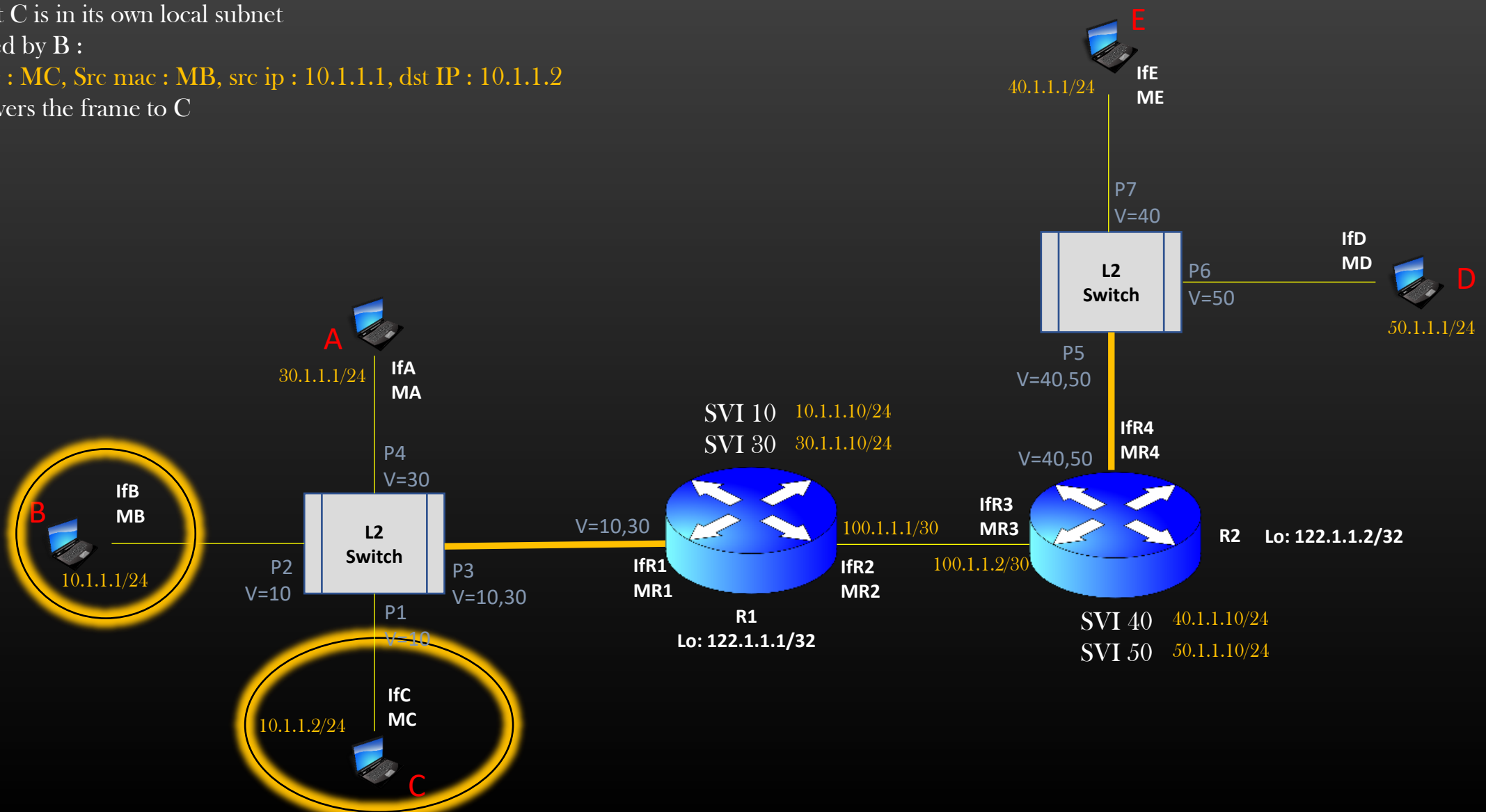
Destination Subnet	Gateway	OIF
10.1.1.0/24	100.1.1.1	ifR3
30.1.1.0/24	100.1.1.1	ifR3
122.1.1.1/32	100.1.1.1	ifR3



Exercise

Q. How B and C will communicate ?
Ans : Apply Concepts for Communication between hosts present within same VLAN

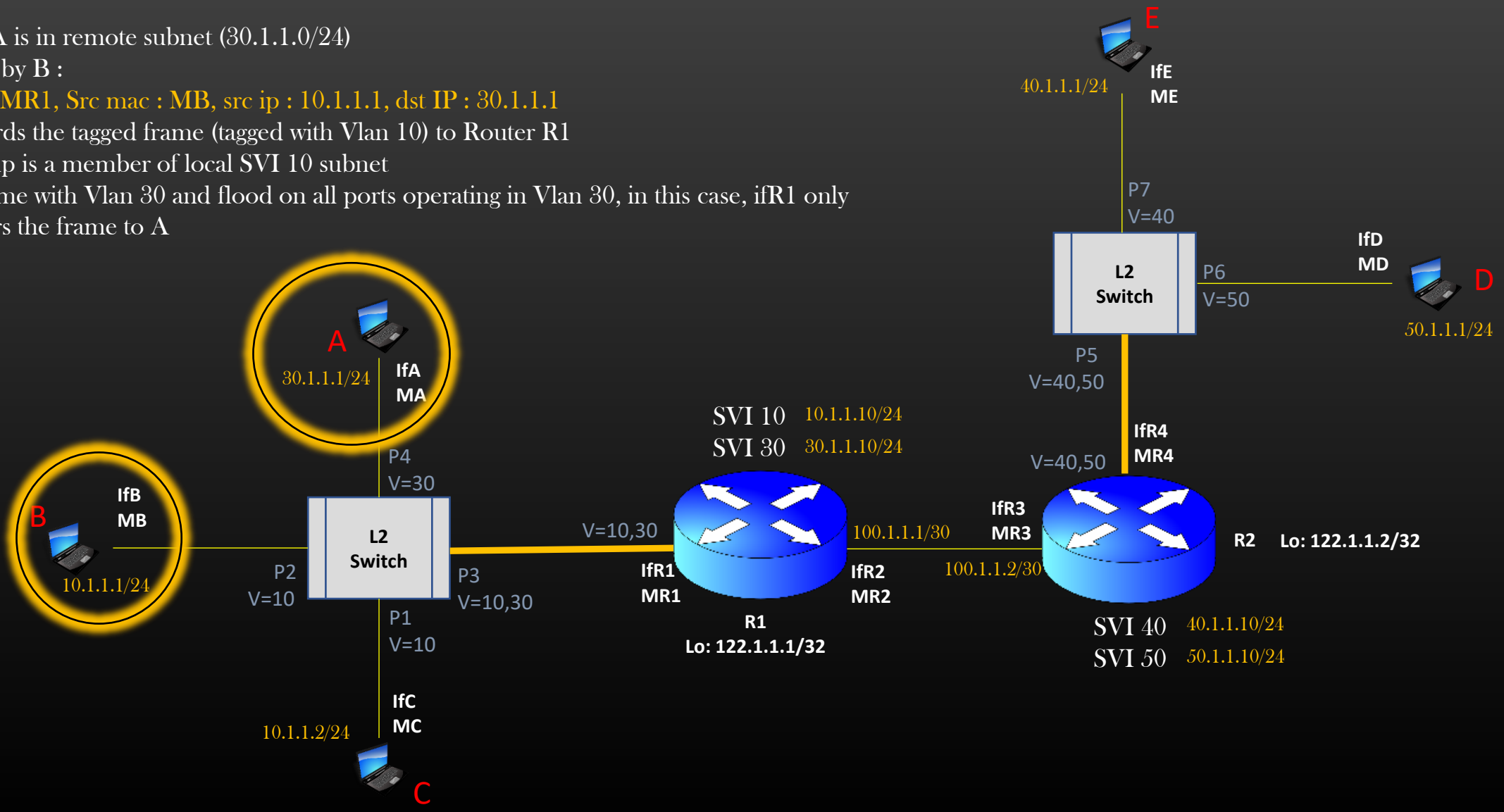
- B finds that dst C is in its own local subnet
- Frame prepared by B :
  - Dst mac : MC, Src mac : MB, src ip : 10.1.1.1, dst IP : 10.1.1.2
- L2 Switch delivers the frame to C



Exercise

Q. How B and A will communicate ?
Ans : Apply Concepts for Communication between hosts present in different Vlans (Inter Vlan Routing)

- B finds that dst A is in remote subnet (30.1.1.0/24)
- Frame prepared by B :
  - Dst mac : MR1, Src mac : MB, src ip : 10.1.1.1, dst IP : 30.1.1.1
- L2 Switch forwards the tagged frame (tagged with Vlan 10) to Router R1
- R1 finds the dst ip is a member of local SVI 10 subnet
- R1 retags the frame with Vlan 30 and flood on all ports operating in Vlan 30, in this case, ifR1 only
- L2 switch delivers the frame to A

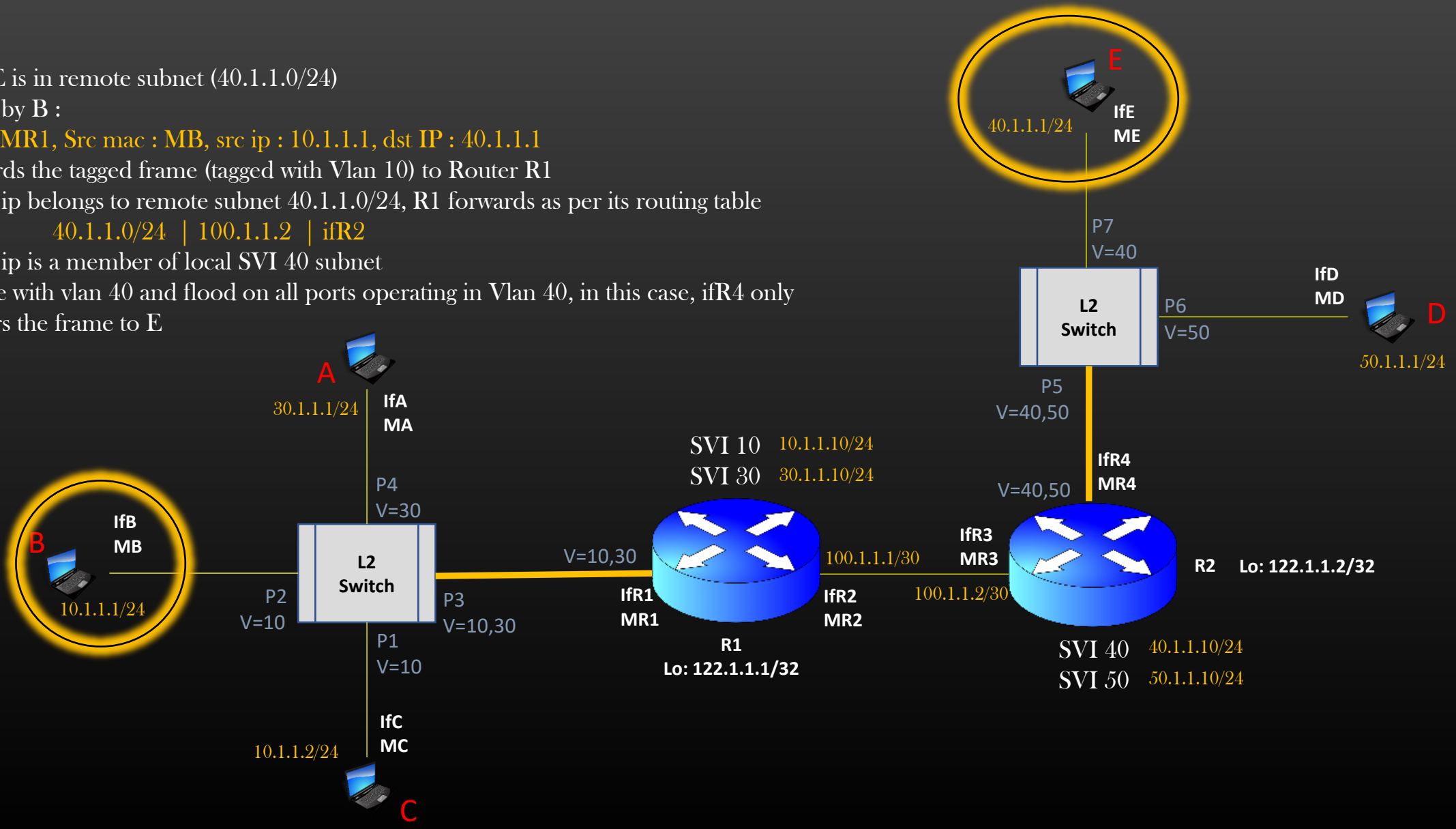


Exercise

Q. How B and E will communicate ?
Ans : L3 Routing coupled with Router to Vlan Routing

- B finds that dst E is in remote subnet (40.1.1.0/24)
- Frame prepared by B :
  - Dst mac : MR1, Src mac : MB, src ip : 10.1.1.1, dst IP : 40.1.1.1
- L2 Switch forwards the tagged frame (tagged with Vlan 10) to Router R1
- R1 finds that dst ip belongs to remote subnet 40.1.1.0/24, R1 forwards as per its routing table

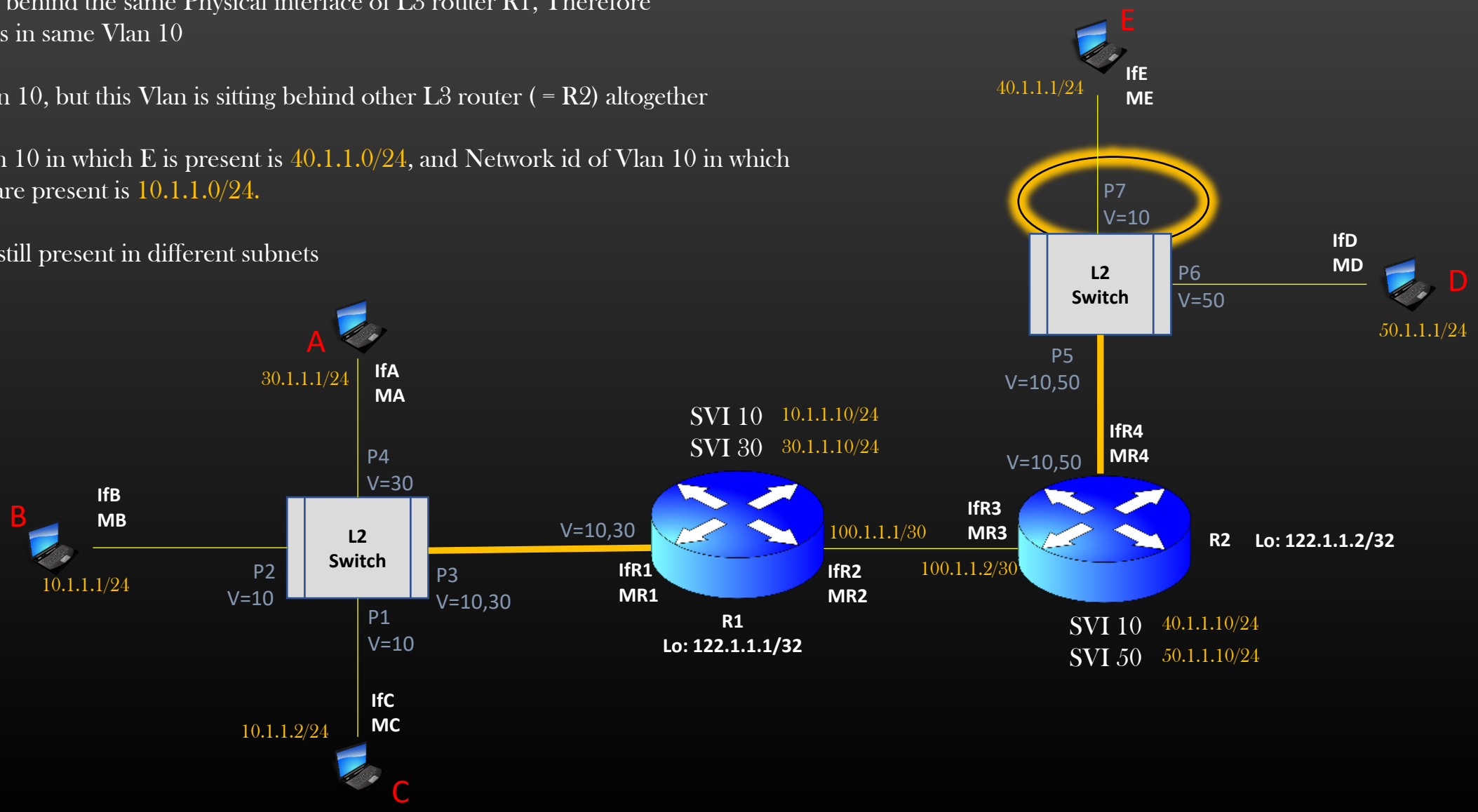
  40.1.1.0/24 | 100.1.1.2 | ifR2
- R2 finds that dst ip is a member of local SVI 40 subnet
- R2 tags the frame with vlan 40 and flood on all ports operating in Vlan 40, in this case, ifR4 only
- L2 switch delivers the frame to E



Exercise

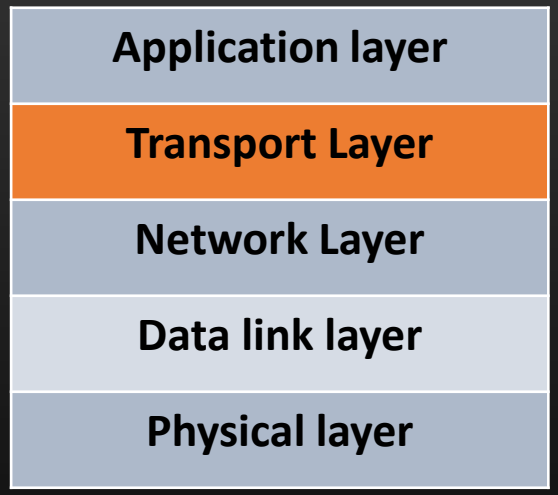
Q. If port P7 of L2 switch to which E is connected is changed to Vlan 10, Do E, B and C are in same Vlan 10 ?
Ans : No

- B and C are sitting behind the same Physical interface of L3 router R1, Therefore
  B and C is in same Vlan 10
- E is present in Vlan 10, but this Vlan is sitting behind other L3 router (= R2) altogether
- Network id of Vlan 10 in which E is present is 40.1.1.0/24, and Network id of Vlan 10 in which
  B and C are present is 10.1.1.0/24.
- E and (B + C) are still present in different subnets



Transport Layer

- **Table of Contents**

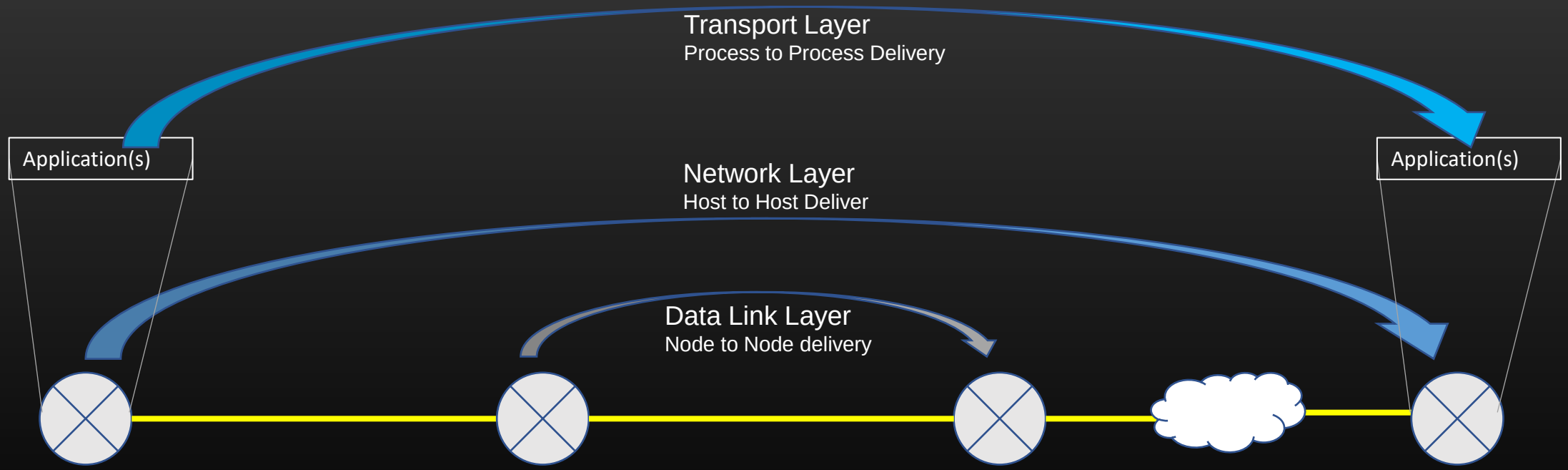


Transport Layer

- **Transport Layer Goals**
  - After the network Layer has verified that packet is not to be forwarded to next hop, and packet is destined to itself, network layer chop off network layer header (IP header) and hands over the remaining packet to upper transport layer
  - To achieve its functions, Transport Layer has 3 famous standard protocols :
    - TCP (Transmission control protocol)
    - UDP (User datagram protocol)
    - SCTP (Stream Control Transmission Protocol)
  - Transport Layer Function is:
    - Identify the correct process/application to deliver the packet (TCP, UDP)
    - Congestion control in the network (TCP only)
    - Reliable delivery of the packets to application (without error) (TCP only)

Transport Layer

- Responsible for process to process delivery
- Not aware of layer above it or below it (This is true for every layer in TCP/IP stack)



Transport Layer

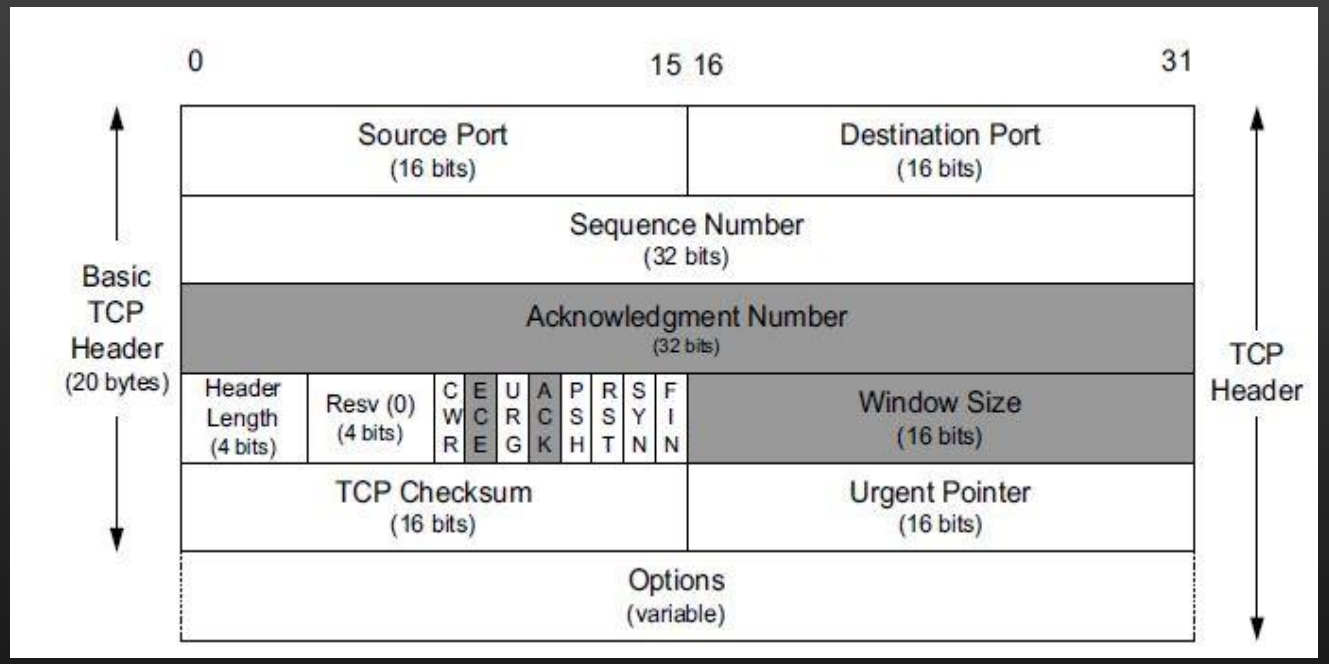
- Two standard and Famous Transport layer Protocols are :
  - UDP (User Datagram Protocol)
  - TCP (Transmission control protocol)

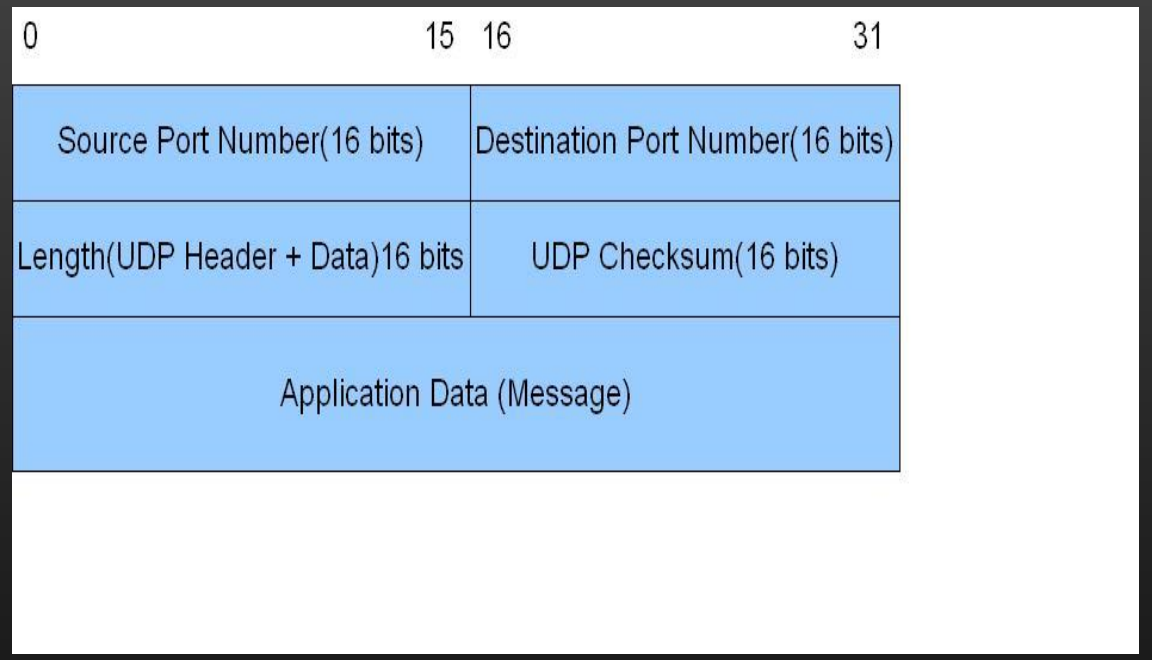
TCP	UDP
If the application needs reliability	Appln do not needs reliability
Slower and complex service	Simpler and fast service
Connection-oriented and reliable	Connectionless and unreliable
Can recover lost packets, detect malformed corrupted packets, react to congestion in the network etc	No such mechanism, a packet lost or corrupted is gone forever
Eg : Sending Email Doc/file	Eg : Audio/Video streaming

Consider TCP and UDP as Twins, both helps achieve the same goal – Data delivery from source PROCESS to destination PROCESS but in a different way.

- TCP is a complex protocol, We shall cover the detailed discussion on internal functioning of TCP protocol in a separate course

Transport Layer – TCP and UDP Headers





- Both Header contains two fields – Source port number and Destination port number
- These port numbers are used by the TCP/UDP protocol to carry out communication between two end processes.
- We shall limit our discussion to port numbers only – Oher fields in the headers (esp TCP) are used for congestion control and error control and should be self-studied from textbooks

Transport Layer – Headers stacking

- A Header is small chunk of leading message appended in the front which contains meta information about payload data it carries
- Without headers, the payload data is unreadable or junk.
- For example : How would you know that network layer header which follows the Ethernet header is Actually IP hdr or IPV6 hdr or any other network layer protocol (Network layer hdr is payload for Ethernet hdr)
- How would you know that protocol used in Transport layer is UDP Or TCP ?
- Thus, the Header H always encodes some information which tells what lies ahead
- Lets see in the next slide how does all this fit in one picture

Transport Layer – Headers stacking

Ethernet Hdr

IP Hdr (protocol #0800)

TCP Hdr (protocol # 6)

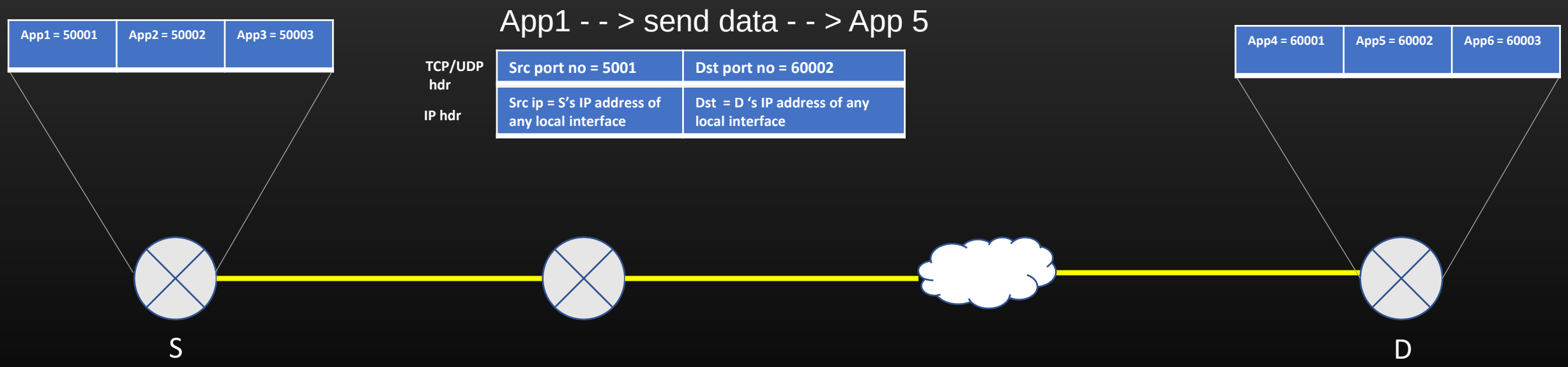
UDP Hdr (protocol #17)

User defined Hdr (protocol # X)

Headers stack

Transport Layer – port number

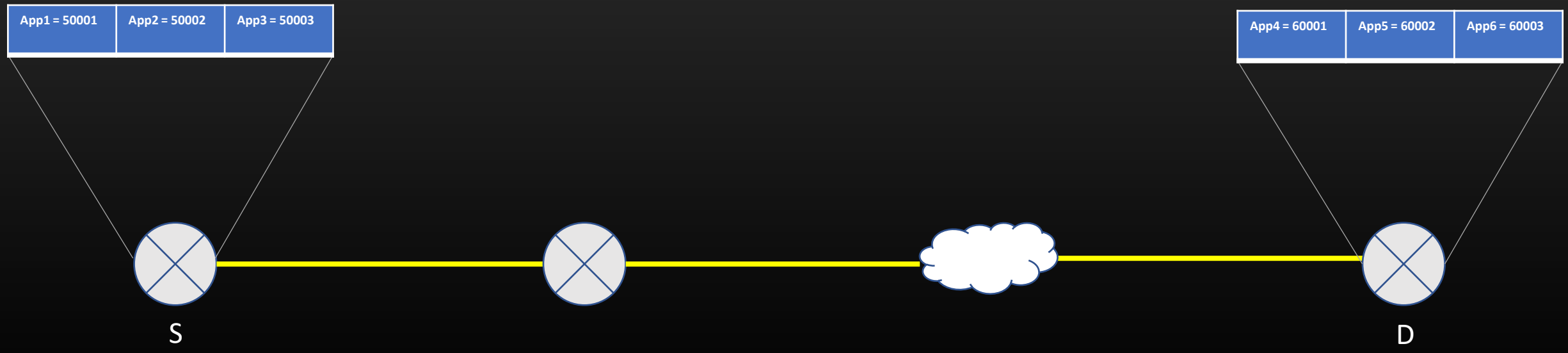
- Port numbers are integer values between 0 – 65,535
- Port numbers helps operating system of receiving machine to identify the process to deliver the packet to
- A process running on the machine, if register a port number, say 60001 for example, all packets which have 60001 as destination port number in their TCP/UDP header will be delivered to this process by the Transport layer



- Like Src and Dest ip addresses in IP hdr, Src port and destination port number in TCP/UDP hdr never changes.
- Only src and dst mac address in Ethernet hdr changes hop by hop

Transport Layer – port number

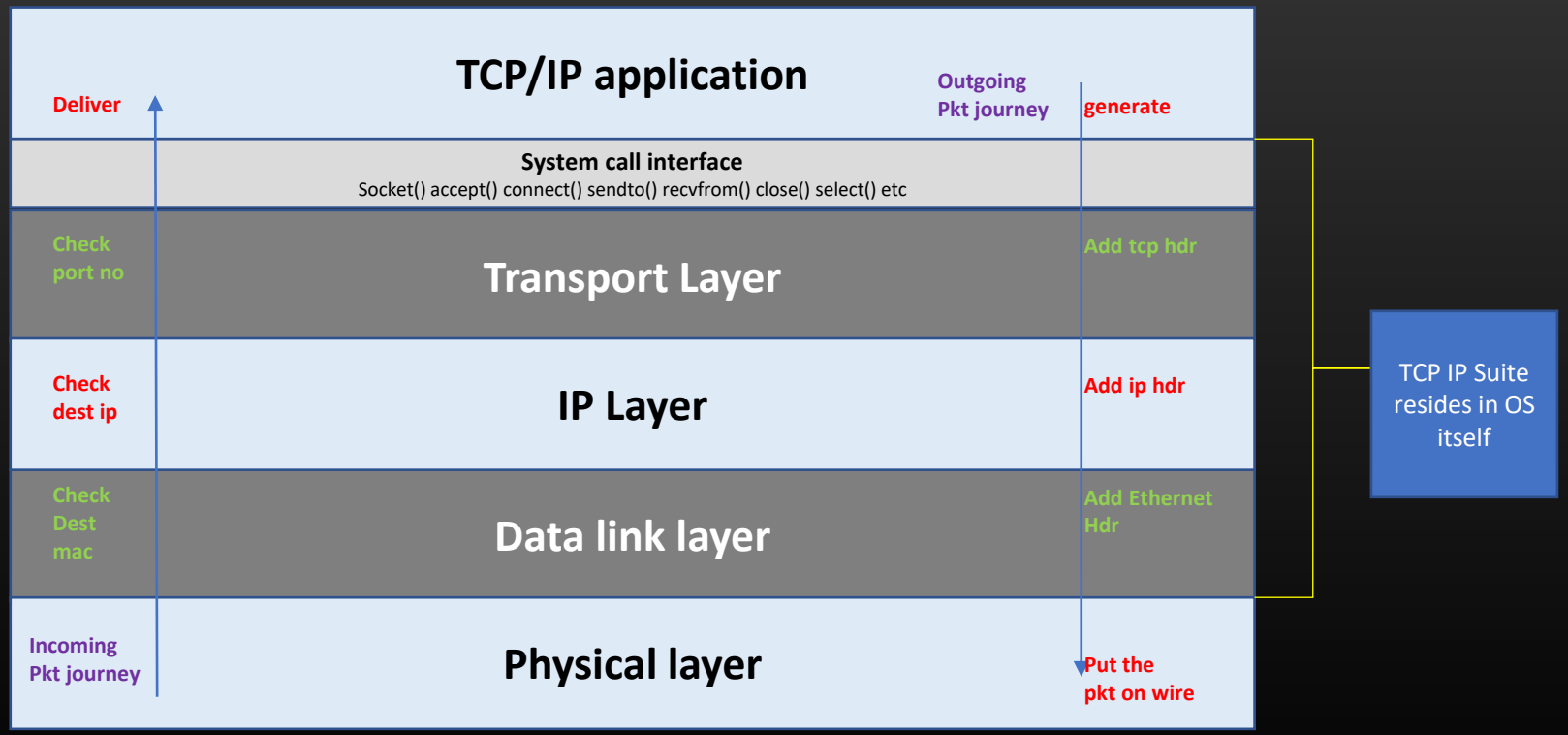
- So, if Process App2 running on machine S wants to send Data to any Process App5 running on machine D which could be present anywhere in the world, App2 needs following pieces of information :
  - 1. App5 ‘s hosting machine ip address (of any local interface)
  - Port no on which App5 is listening/registered
  - Rest of the information required to deliver the packet is gathered by Operating System of sending machines/intermediate machines such as – TCP seq no, src and dst mac addresses etc.



Transport Layer – System Call interface

System Calls are APIs exposed by the OS to User space applications. Applns interact and invoke/Use the OS Services through system calls.

Eg : *malloc/free* for memory allocation and deallocation



OSI Model | TCP/IP stack

Remember our old diagram

- Above the transport Layer Sits the system call interface APIs
- These APIs are used to send Or receive packets to Or from the underlying Layer of TCP/IP stack
- These APIs are used for Writing Network applications. Socket programming is all about using these APIs by the application to harness Networking features provided by the underlying OS (TCP/IP stack)

Summary

- We learnt the significance of port numbers. What IP addresses are to Network Layer, is what Port numbers are to Transport Layers
- We learnt about Header stacking. Networking is all about placing one header after the other. We do this all the time.
- Receiving Machine delivers the TCP/UDP packet to an application which is registered for port number specified as dst port number in TCP/UDP hdr of the packet
- Receiving machine encode the src port number received in the UDP/TCP packet as dst port in TCP/UDP reply packet
  (Port number Swapping)

Lecture VDO 8

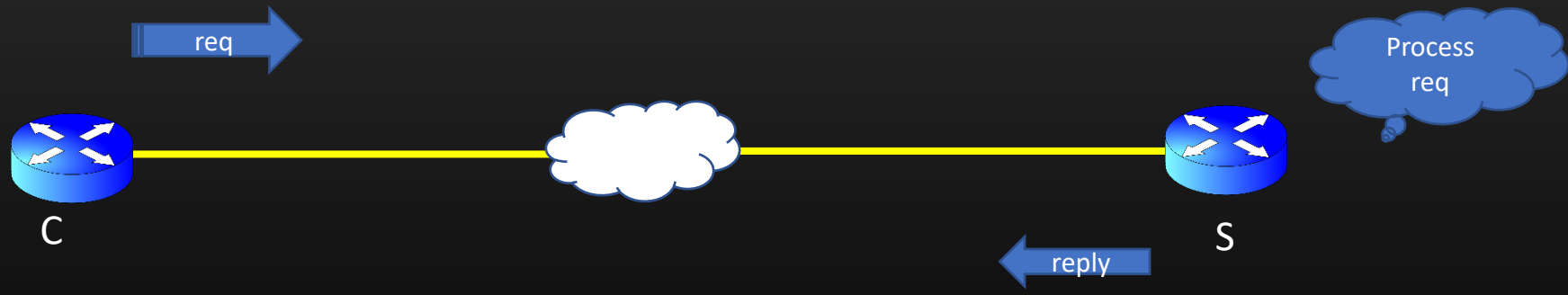
Socket Programming

- **Table of Contents**

Lecture VDO 8

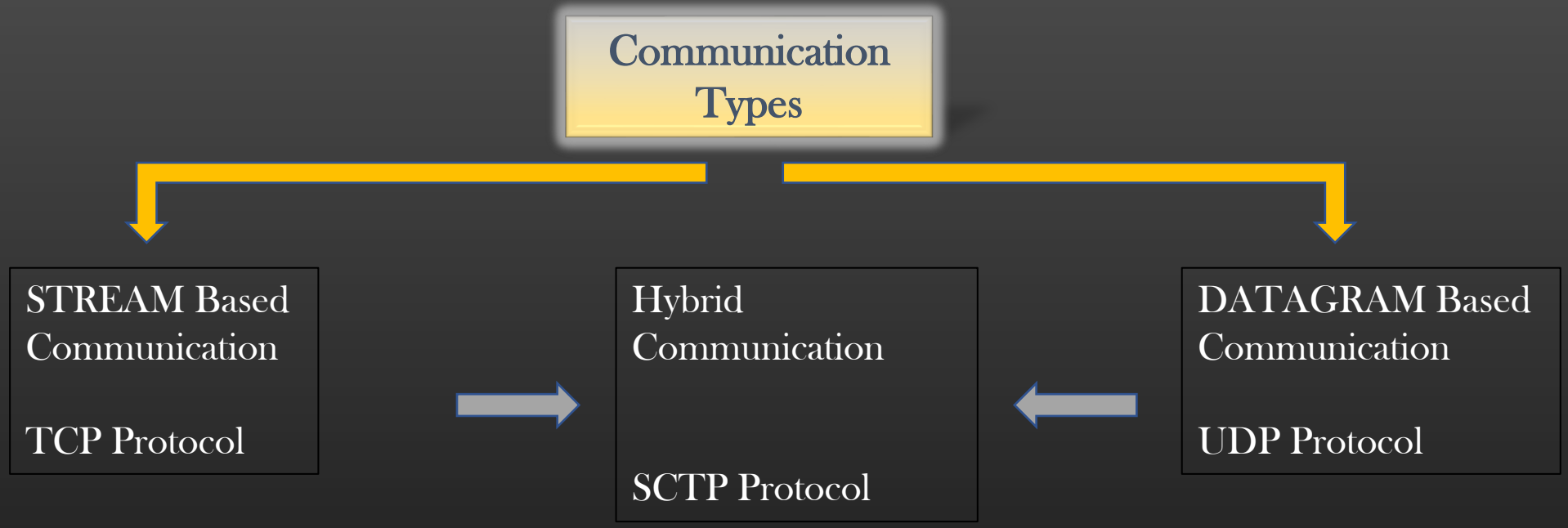
Socket Programming -> Client and Server based Model

- Server is any machine which receives the request, process it and optionally returns back the result
- Client is any machine which initiates the request
- We will quickly write a simple Client Server Hello world model



- Server never initiates the communication

Communication Types



- A Communication between two machines can be broadly classified into STREAM or DATAGRAM based Communication
- Which type of communication to setup – depends on end goals
- Each Communication type has its own pros and cons
- Great Interview Question – you are given a situation and asked which type of communication you would setup and why !!

Communication Types

STREAM based Communication

TCP

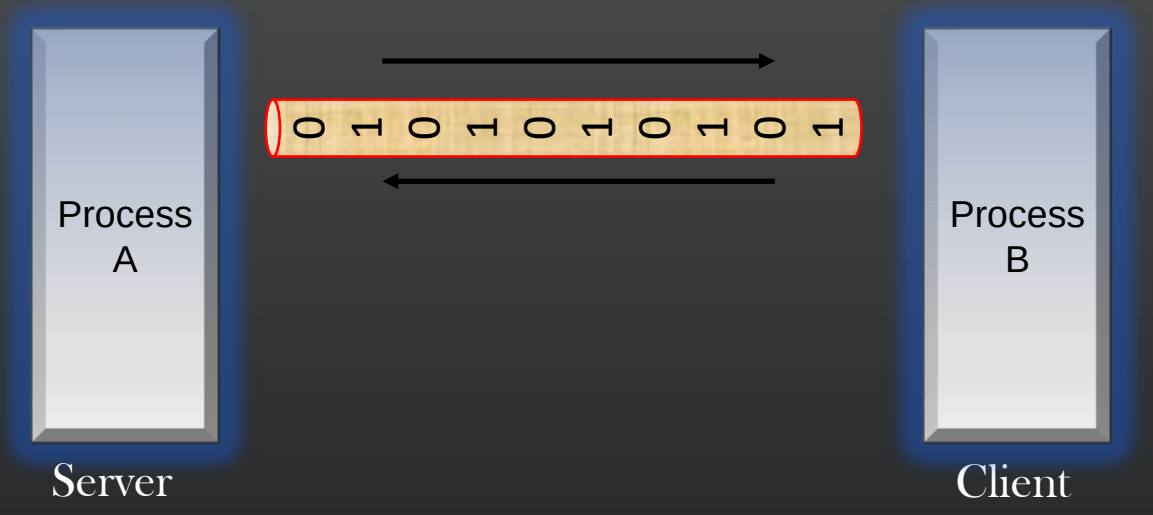
Transmission Control Protocol

&

UDP

User Datagram Protocol

TCP based Communication



1. Connection Oriented : Dedicated connection between A & B
   - Server and Client maintains the internal Data structures to maintain connection session state
   - Communication processes tracks the amount data which is sent, not sent, pending ACK, ACK'ed etc
   - Eg : Downloading software resumes downloading the file from the same state when internet connectivity restores

2. Mutual Agreement : 3-Way handshake necessary before any actual data exchange takes place
   > 3-Way handshake is a mechanism using which the communicating processes mutually agrees to participate in bidirectio connection oriented communication

3. Reliable : Design to be reliable, recoverable from errors, error correction, deal with congestion
   > A lost Data is re-transmitted by sender, so delivery of all data bytes are guaranteed.
   > If Network is congested, Sender can slow down the rate of data transmission
   > If receiver receives any corrupted data, receiver can ask the sender to retransmit

4. STREAM of Bytes : Sender can send continuous stream of bytes of data to recipient, analogous to flow of water through a pipe

TCP based Communication

5. Sequenced Data : Data packets arrives in the same sequence to recipient application in which they were transmitted

6. Use-case : Should be used where receiver can tolerate a time LAG but not packet loss Or if the info is sensitive towards pkt loss

7. Example :

Downloading an Audio Or a Movie Or software *stored on a disk on a server – Little Delay is tolerable*

Emails, Chat messages etc – *Sensitive towards packet loss*

Suggested Readings : To Study internal implementation in Detail how STREAM based communication works, you are recommended to study the internal implementation of TCP/IP protocol

UDP based Communication

1. Connection-less : No Connection establishment between A & B
   - Sending machine do not maintain any connection state Or store the identity of other machine
   - Every data transmission instance is a fresh data transmission

2. Mutual Agreement : Communicating machine triggers data transmission without any prior mutual agreement Or notification

3. Reliable : Not Reliable, not to be used in error prone Networks Or environments
   - > A lost Data is lost forever
   - > no Recovery mechanism to deal with congested network or corrupted data

4. DATAGRAM of Bytes : Data is sent as discrete isolated units of data called Datagrams. Each Datagram is a complete Unit of data by itself

5. Sequence : Datagram can arrive out of sequence on recipient machine

UDP based Communication

6. Use-case : Should be used where recipient can tolerate a packet loss, but not a time lag

7. Example :

LIVE Video/Audio conferences

You can compromise you Audio/Video Quality, but time lag in LIVE sessions is horrible

SCTP based Communication

Communication Types -> Exercise

Scenario	STREAM Or DATAGRAM	Reason
Your finger prints are being collected by scanner and being saved on local Disk	DATAGRAM	No error, no Congestion, No loss. Either of the two could be used
Uploading/downloading Videos on/from Youtube	STREAM	Your Data is going out on open WAN, reliability should be ensured
Downloading ISO image file of Windows 10 from MS Servers	STREAM	Even if one packet is missed, the software may not run at all. Delay in download is tolerable
On Google Hangout or In Team meetings on Skype	DATAGRAM	Time lag is not tolerable, few packet losses are tolerable
Sending Emails	HYBRID	Data loss is not tolerable, Individual units of data is sent
Sending Text messages	HYBRID	Data loss is not tolerable, Individual units of data is sent

Lecture VDO 8

Socket Programming -> Select And Accept System Calls

- Linux OS provide two system calls to enable us write efficient Network Applications
  - Select()
  - Accept()
- In C, it is difficult to write a socket based application without the use of Select and Accept System calls
- From interview point of view – Understanding these two calls is very important
- Let us first grasp the idea of these two APIs. Having understood these two, remaining socket programming APIs would automatically fall in place

Lecture VDO 8

Socket Programming -> Select And Accept System Calls

- High level Linux Socket based Communication Design

- Messages (Or requests) send by the client to the server can be categorized into two types :
  - Connection initiation request messages
    - This msg is used by the client to request the server to establish a dedicated connection.
    - Only after the connection has been established, then only client can send Service request messages to server.

    And

  - Service Request Messages
    - Client can send these msg to server once the connection is fully established.
    - Through these messages, Client requests server to provide a service

- Servers identifies and process both the type of messages very differently

Lecture VDO 8

Socket Programming -> Select And Accept System Calls

- High level Socket Communication Design

Msg type	Client side API	Server side API
Connection initiation request msgs	connect()	accept()
Service request msgs	sendmsg(), sendto()	recvmsg(), recvfrom()

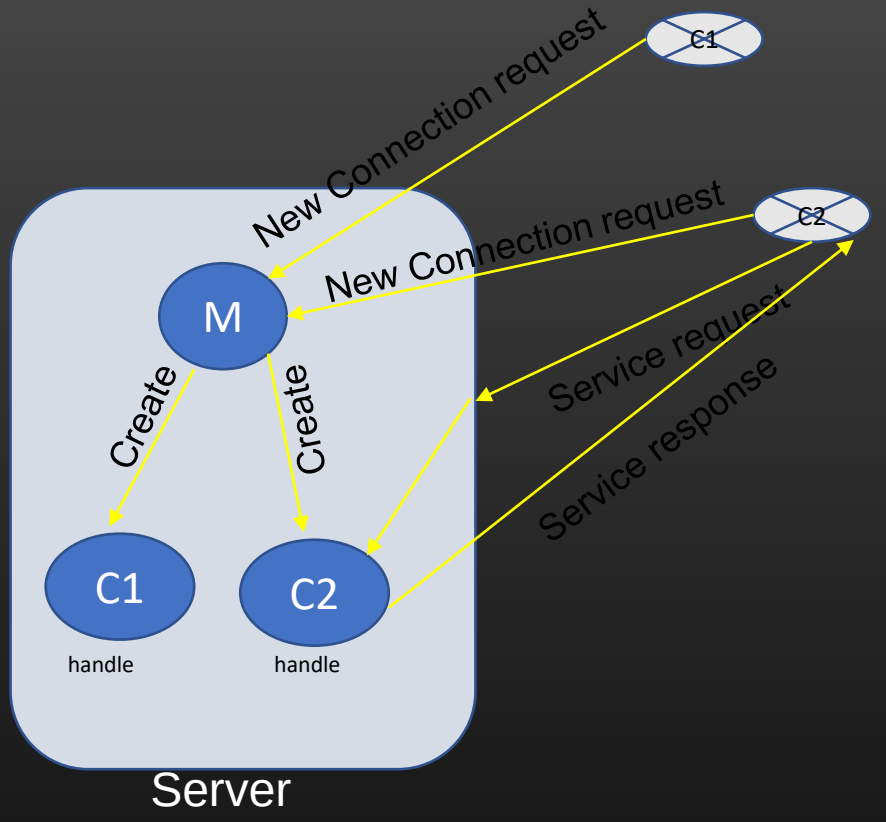
Lecture VDO 8

Socket Programming -> Select And Accept System Calls

- Life Cycle of a Client Server Communication

When Server boots up, it creates a master socket using socket()

= socket()



M is the mother of all Client Handles. M gives birth to all Client handles.

Once Client handles are created for each client, Server carries out Communication with the client using Client handle (and not M).

Server Has to maintain the database of connected client handles

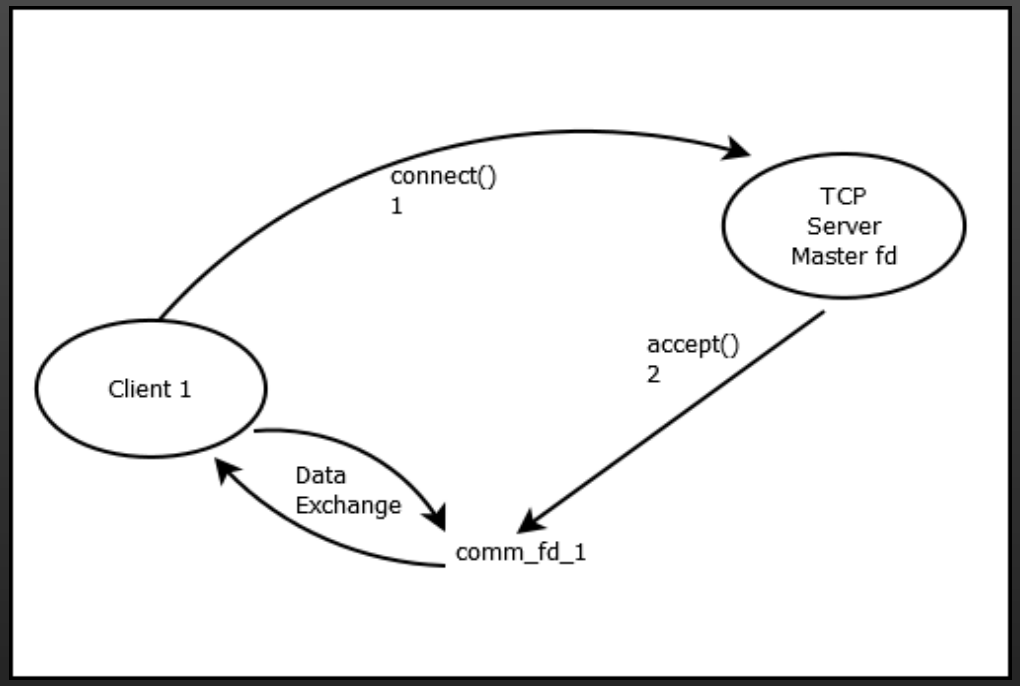
M is only used to create new client handles. M is not used for data exchange with already connected clients.

Accept() is the system call used on server side to create client handles.

In Linux Terminology, handles are called as “file descriptors” which are integer numbers. Client handles are called “”communication file descriptos” and M is called Master socket file descriptor

Lecture VDO 8

Socket Programming -> Accept() System Call

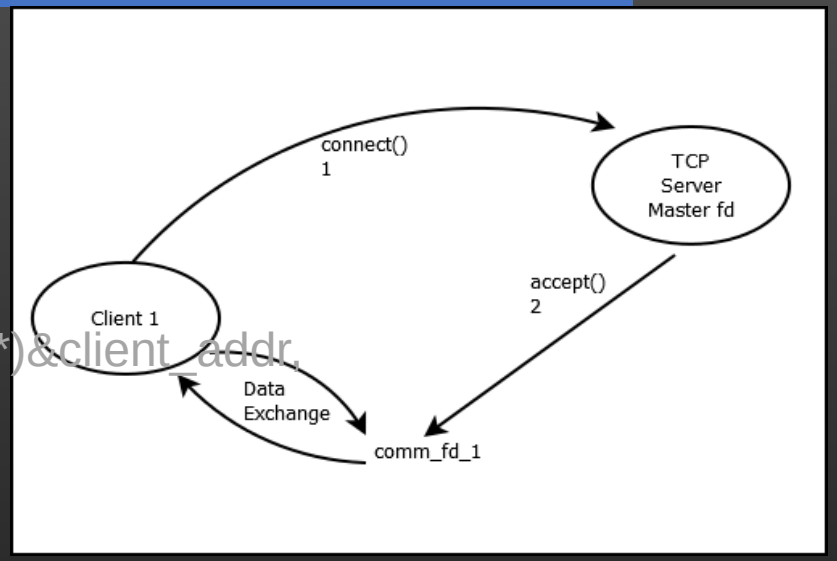


- Accept()
  - Accept() system call is used by the TCP server (not UDP server) to accept the new client connection request
  - Remember, TCP is a connection oriented protocol, meaning – client and server needs to establish dedicated connection first before they can participate in any data exchange activity
  - For example, when you login into facebook, your machine sends connection request to Facebook servers. Facebook servers accept your connection request using accept() system call. Now, you can send data to facebook servers, and facebook servers can update your page with notifications/feeds etc.
  - The sole purpose of Accept() just to establish the connection between two machines (client and server) and carrying out TCP 3-way handshake (refer to your standard books)
  - Accept() is called only on server side, and it returns the handle to the connection with client. Using this handle Server carries out all future communication with the client. This handle is called *Communication file descriptor.*

Lecture VDO 8

Socket Programming -> Accept() System Call

```
int comm_sock_fd =          accept ( master_sock_tcp_fd,
                                     ( struct sockaddr *)&client_addr,
                                       &addr_len)
```



- Accept()
  - Accept() returns a **file descriptor** (comm_socket_fd), which is just an integer, representing as the dedicated connection handle between connected client and server. Server carries out all communication with client using this returned integer value.
    - master_sock_tcp_fd is the fd of the socket which server has opened to listen to new client connection requests using *socket()*
    - client_addr is the structure which contains client info – ip address and tcp port no of client which has just connected
    - addr_len = size of predefined structure - *struct sockaddr* which is just a constant.
    - There are some more APIs – sendto and recvfrom, but there explicit discussion is not worth.
    - You will understand through code walk-through.

Lecture VDO 8

Socket Programming -> Select() System Call

- Select() system calls allows the server machine to MONITOR multiple client connected connections and check which client has send the data to process
  >> Just like the class monitor keep an eye on all the students of the class at the same time

- Select() system call is ***blocking*** System call – meaning, when it is executed, the code execution halts (similar to scanf/getch)

- Select() unblocks when either of two things happen on server side :
  - New connection request from new client arrives
  - Data request from existing connected client arrives

- When select() unblocks, server needs to check whether it's a new connection request Or new data request on existing connection.
  In the later case, server need to find which client has send the data.

- When server starts, the first thing it does is to create a master socket – A socket to detect arrival of new connection request from new client

- C provides a Data structure called *fd_set* which is a collection (or a set) of file descriptors fd_set reafds;

Lecture VDO 8

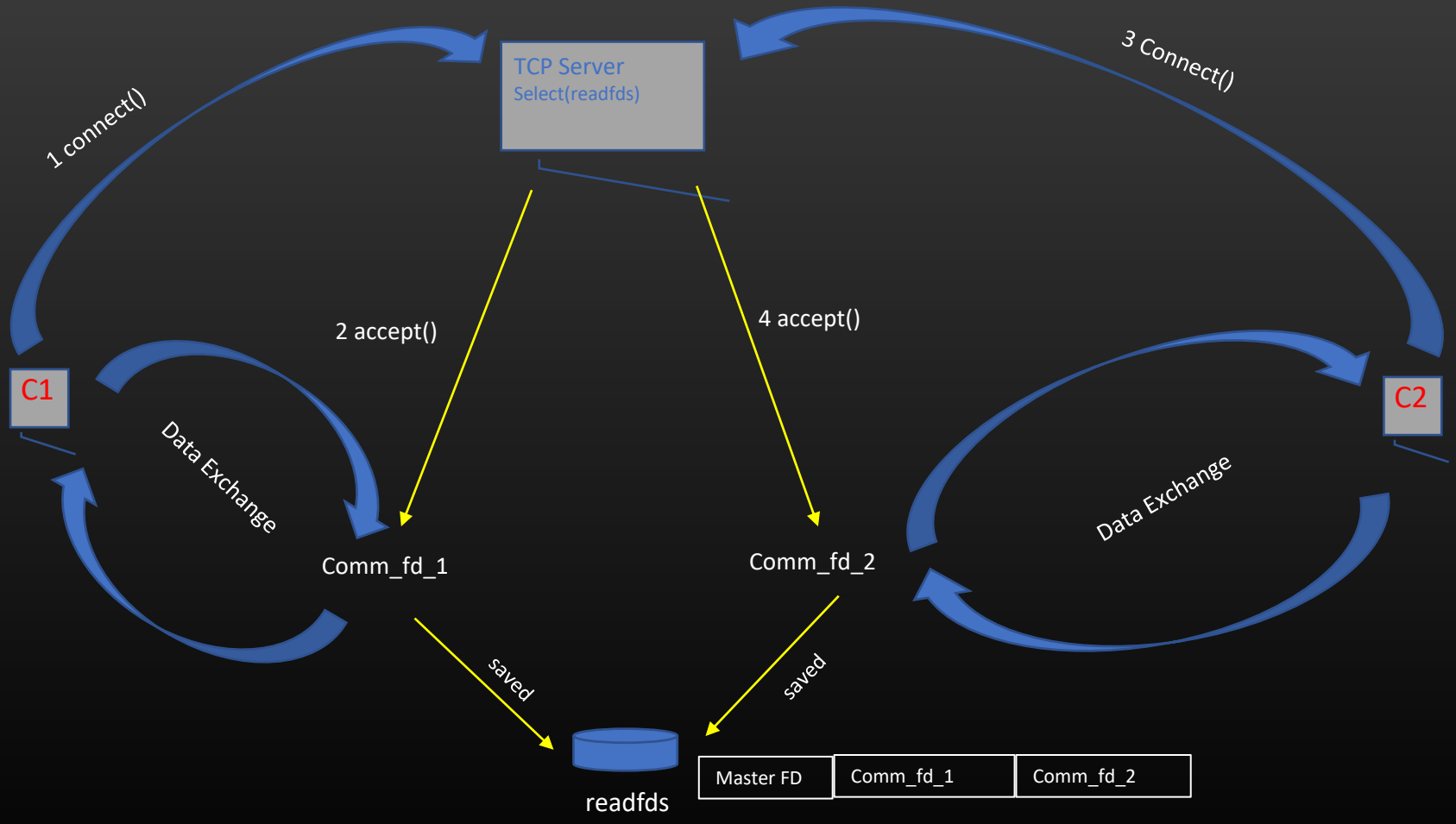
Socket Programming -> fd_set

- C provides a Data structure called *fd_set* which is a collection (or a set) of file descriptors

  fd_set reafds;

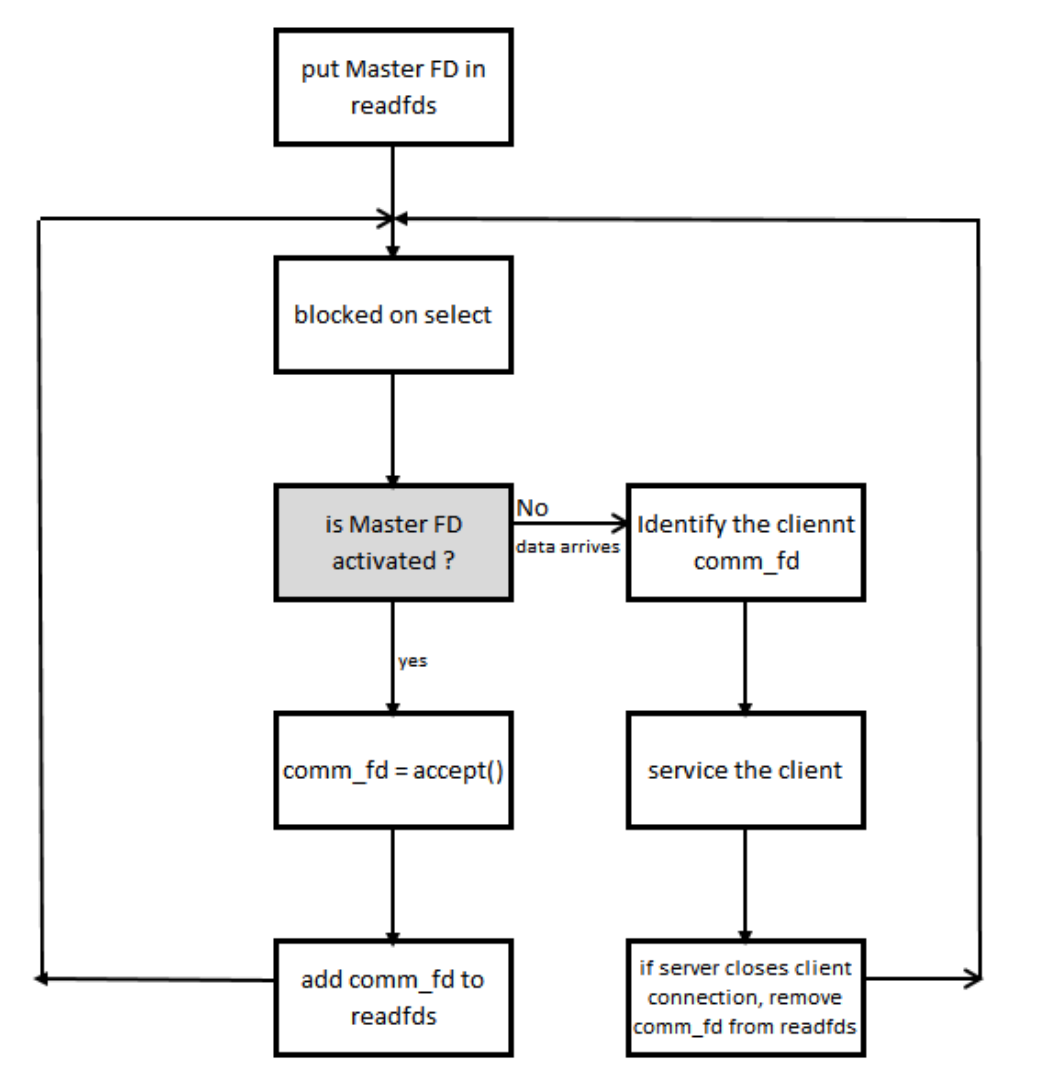
Lecture VDO 8

Socket Programming -> Accept() and Select() System Call in operation



Lecture VDO 8

Socket Programming -> State machine Diagram of a server



Time for a Code walk …

Lecture VDO 8

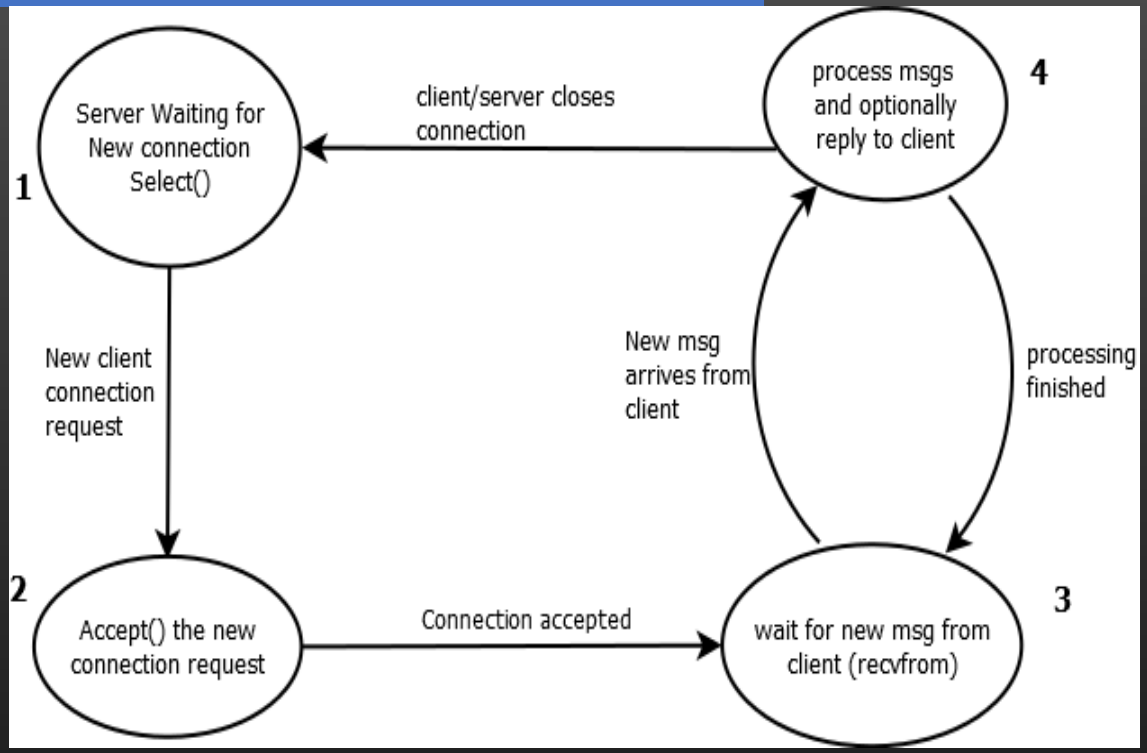
Socket Programming -> Server Implementation

Code Walk

1. Initialize variables
2. Create Master socket
3. Bind
4. Listen
5. Initialize and fill readfds
6. Select
7. Accept the connection
8. service the client requests
9. close the connection
10. Goto 5

Lecture VDO 8

Socket Programming -> Server Implementation



- Observations
  - Server cannot accept other client's request while it is processing the current client (oscillating between stages 3 and 4).
  - Only one client can be served at a time.
  - This kind of design is used where there is only and only one dedicated client of server. Such server processes are used in system soft-wares where there is only one client and one server.
  - This server design cannot handle multiple clients connections
  - Remember, Servers are just another process which accepts some data from another process , nothing special.
  - Server doesn't generate any data, it just replies to the client after processing client's data. Server by themselves never sends msg to client without Client's initiation first.

Lecture VDO 8

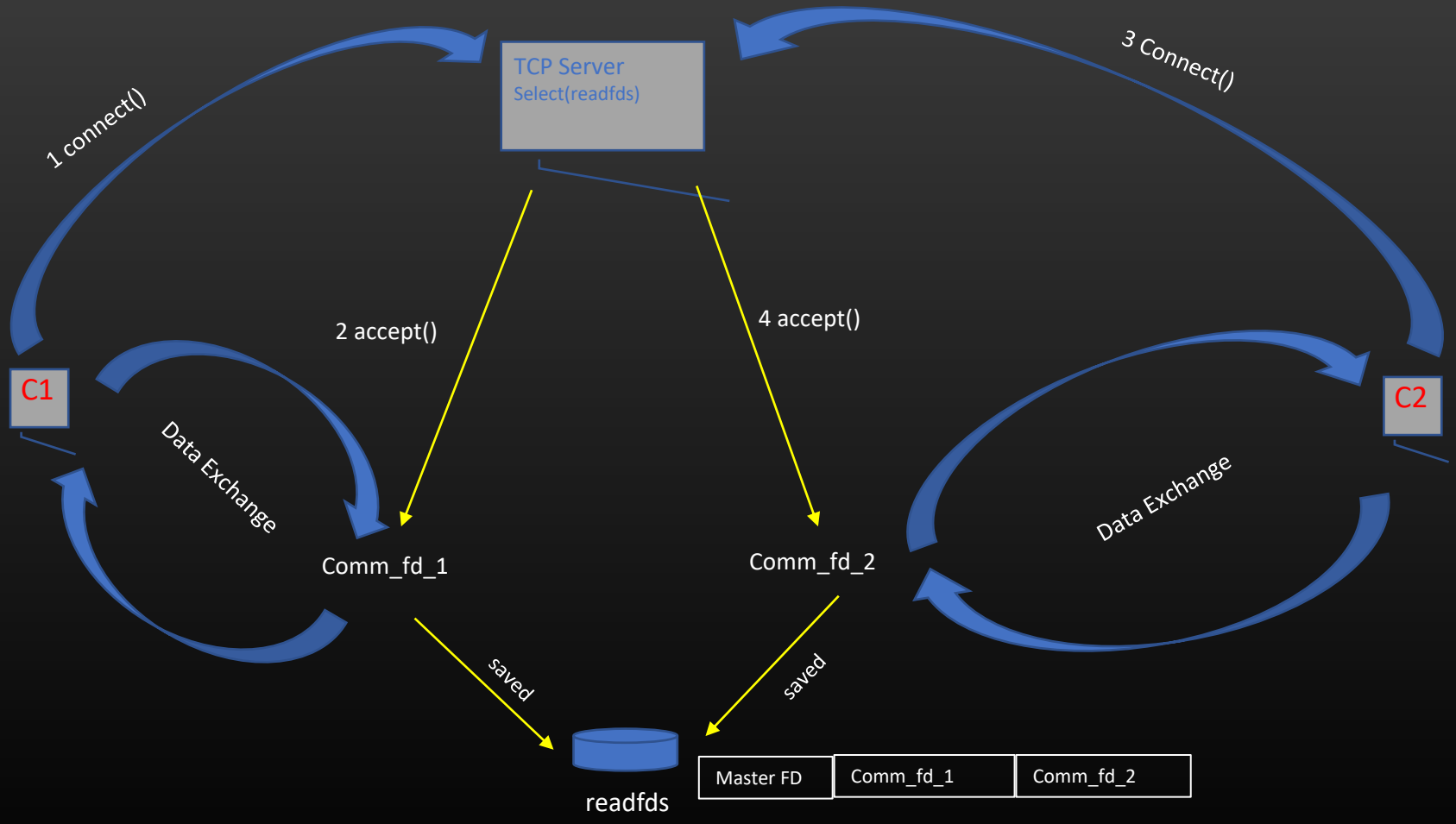
Socket Programming -> Multiplexed TCP Server Implementation

Code Walk For TCP Server which can handle multiple client connections

1. Initialize variables
2. Create Master socket
3. Bind
4. Listen
5. Initialize and fill readfds
6. Select
7. Accept the connection
8. service the client requests
9. close the connection
10. Goto 5

Lecture VDO 8

Socket Programming -> TCP Server with Multiplexing



Lecture VDO 8

Socket Programming -> Server Implementation

Socket programming APIs

1. socket() – Used to create a TCP/UDP master socket. Used on server and Client side

2. select() – Used for monitoring multiple file descriptors. Can be used on both client and server sides. Blocking system call. Blocks until new connection request Or data arrived on FDs present in readfds set.

3. accept() – Used on TCP server side. Used to accept the client connection request and complete 3-way handshake with client

4. bind() – Used on TCP/UDP server side. Used by the Server application process to inform operating system the criteria of packets of interests

5. listen() – Used on TCP/UDP server side. Used by the Server application process to inform operating system the length of Queue of incoming connection request/data request from clients

6. recvfrom() – Used on the TCP/UDP server and client side. Used by the Server/client process to read the data arrived on communication file descriptors. This call is blocking call by default. Process block if data hasn't arrived yet.

7. sendto() – Used to send data to client/server. Used on Server and client sides.

8. close() – Used to close the connection. Used by both – clients and Server

Lecture VDO 8

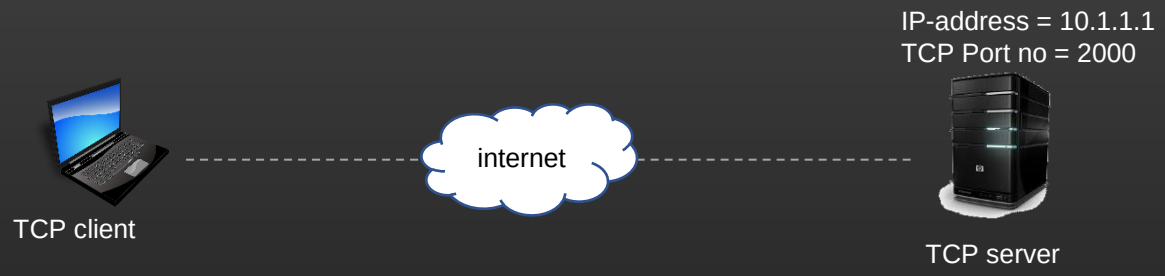
Socket Programming -> Select() Accept() code walk through

- Observations
  - Server Can accept and maintain upto 31 client's connection request (1 being Master skt fd)
  - Server can service one client at a time, however can maintain 31 connections
  - If multiple client's data request comes together (upto n), all client will be serviced but one by one, where n is the value specified in listen()
  - With the increase in no of clients, clients can begin experience latency as they are serviced one by one
  - Thus, this server design is way better than previous one, but has scalability limitations
  - Scalability limitation will always be there, no matter how you design the server. To handle more no of clients, we need to deploy more independent server machines. That is why Facebook, Google have data centers with thousands of servers to handle millions of client request.
  - So don't expect a single server machine to handle significant no of clients. Such a server is not limited by software design, but by hardware limitations.

Lecture VDO 8

Socket Programming -> TCP Client Implementation

Code Walk For TCP client



1. Initialize variables
2. Specify Server Credentials
3. Create a Communication socket
4. Connect with Server (Send connection initiation request to server)
5. Send data to Server
6. Receive response
7. (optional) close the connection
8. Goto 5 if wants to communicate more with the server over same connection

- No port number assignment by the programmer
- No bind() , no fd_set, no accept()

Lecture VDO 8

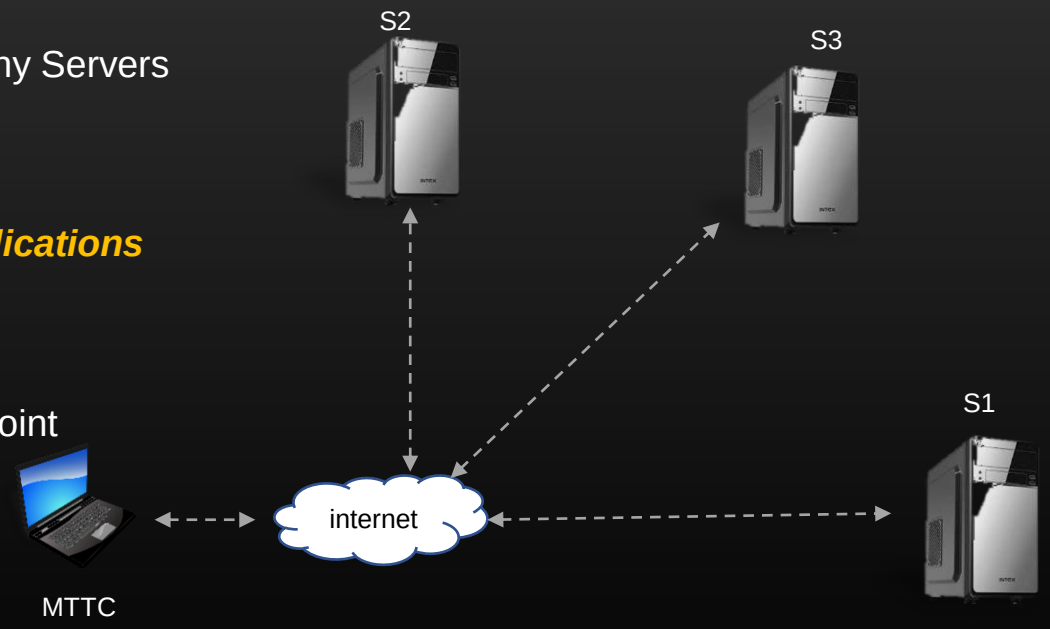
Socket Programming -> UDP Programming

- How to create UDP server machine and UDP client machine !
- If you have learnt creating TCP server machine, UDP programming is way much simpler than TCP programming …
- Homework : Write your UDP Server and UDP client. Verrryyy easy ….. As compared to TCP counterparts. Infinite material available all over internet. Google…

Lecture VDO 8

Socket Programming -> Coding assignment

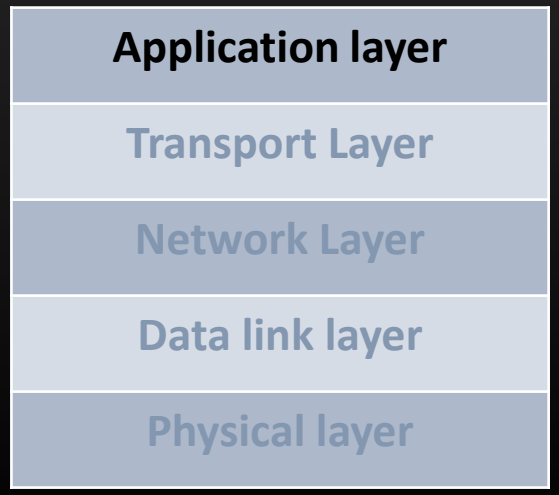
- Implement a **Multi-talking TCP client**
  - A Multi-Talking TCP client (MTTC) is a client which communicates with multiple TCP servers at the same time
  - MTTC do not ***blind-wait*** for the response from server
  - MTTC can send data request to any server in any sequence
  - MTTC should be able to process the result received in any sequence from any Servers
  - We need to Design MTTC TCP client which meets above properties
  - This assignment will teach you how to design ***Asynchronous Network applications***
    - Asynchronous means
      - Sending Or receiving events (data packets in this case) at any point of time
      - Network Applications should be Asynchronous in Design ☺☺



Lecture VDO 9

Application Layer

Application Layer



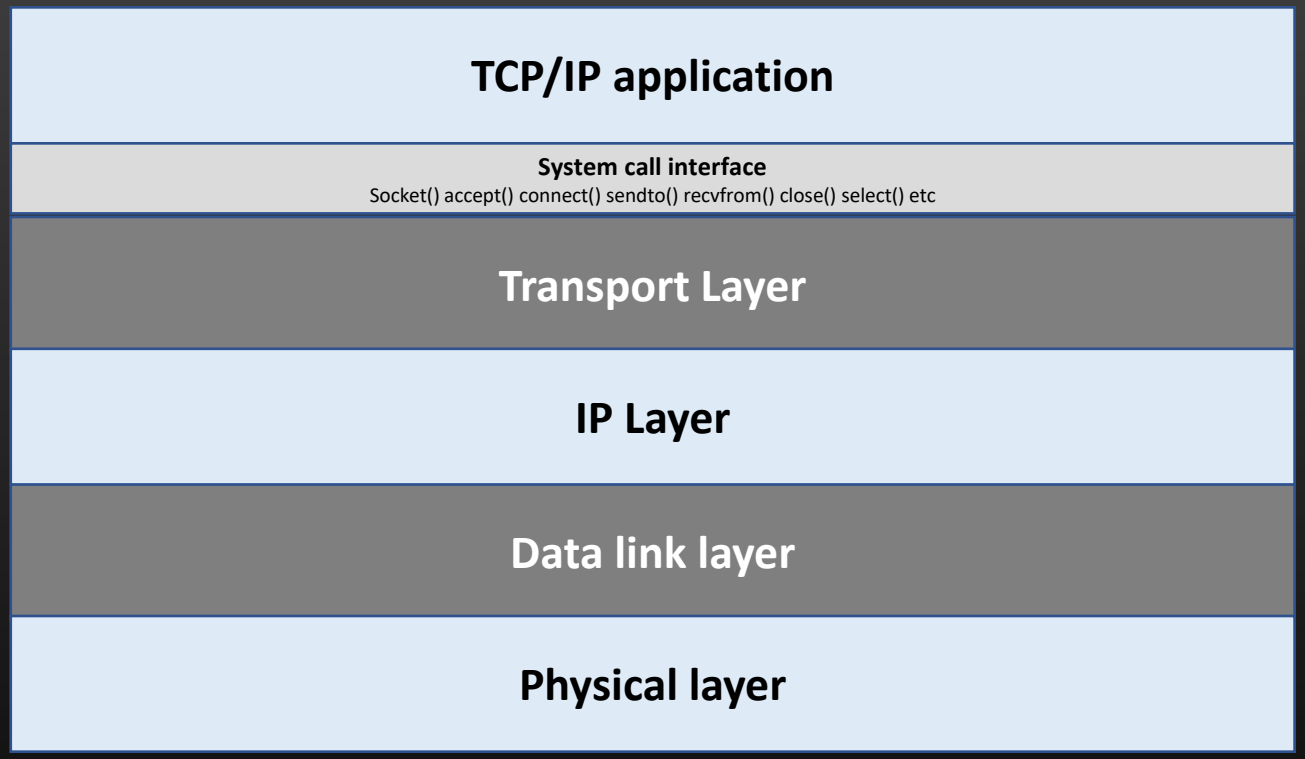
Lecture VDO 9

Application Layer

- Application Layer is the top most layer in the TCP/IP Stack Or OSI Model
- It is a layer at which the packet journey starts On sending machine and Ends on destination machine
- Application Layer lies outside the OS(Kernel)
- The user space program that we right in our day to day life are all examples of applications
- The TCP client server program that you wrote were also an example of Application programs running in application layer of TCP/IP stack
- One Network application program you are using right now is the Brower which is making use of Application layer protocol HTTP
- Other examples of Network application programs we discussed in this course is – ICMP protocol
- In this module, we will learn how to write our own application program which have its own application headers

Lecture VDO 9

Application Layer



OSI Model | TCP/IP stack

- We Will going to see how to implement Custom Application protocol

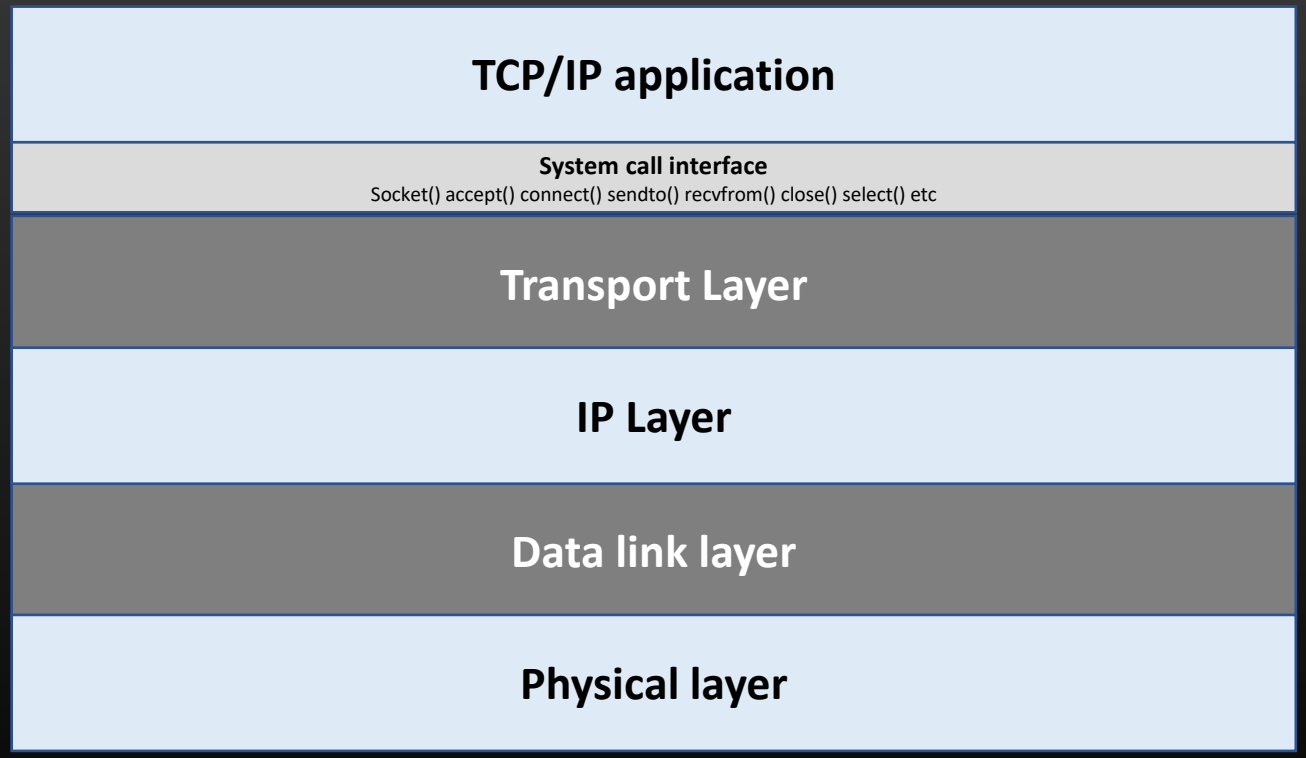
- Just like other layers of TCP IP Stack, Our appln protocol will also have its own application header

- Our application protocol will have its own sub-msgs type as well.
  - ICMP – ICMP echo req and ICMP echo reply
  - ARP – ARP broadcast req and ARP reply

Lecture VDO 9

Application Layer

- Application Protocol can be implemented in three ways :
  - On top of Transport Layer Protocol
  - On top of Network layer protocol
  - On top of Data link later protocol



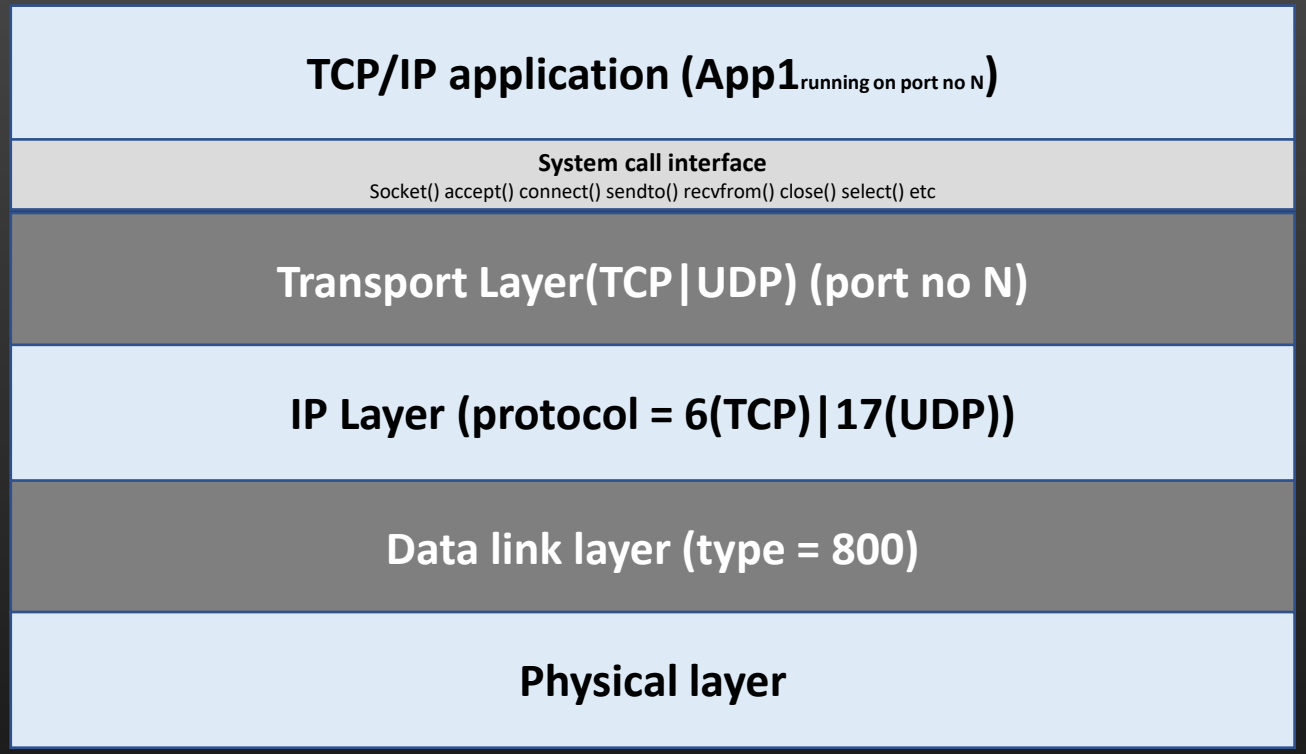
OSI Model | TCP/IP stack

Lecture VDO 9

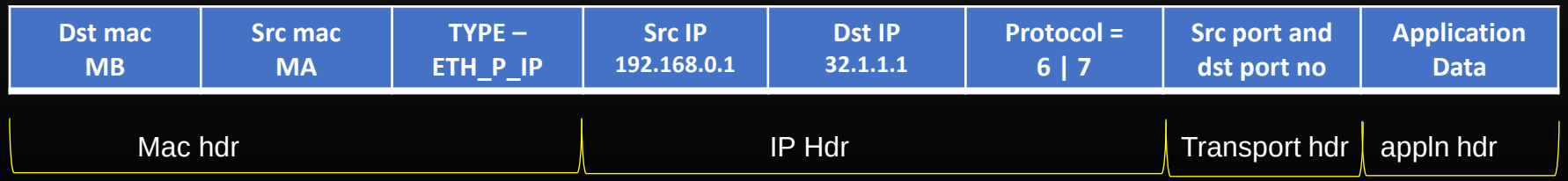
Application Layer

- When Application Protocol is run :
  - On top of Transport Layer Protocol (eg : HTTP webserver)

- It means, Transport layer protocol is UDP/TCP
- Application need to listen on TCP/UDP port number
- TCP/UDP protocol will take the responsibility for packet transmission between sending and receiving process
- Network Layer takes the responsibility for packet transmission from source to destination machine
- Data link layer takes the responsibility for packet transmission hop by hop



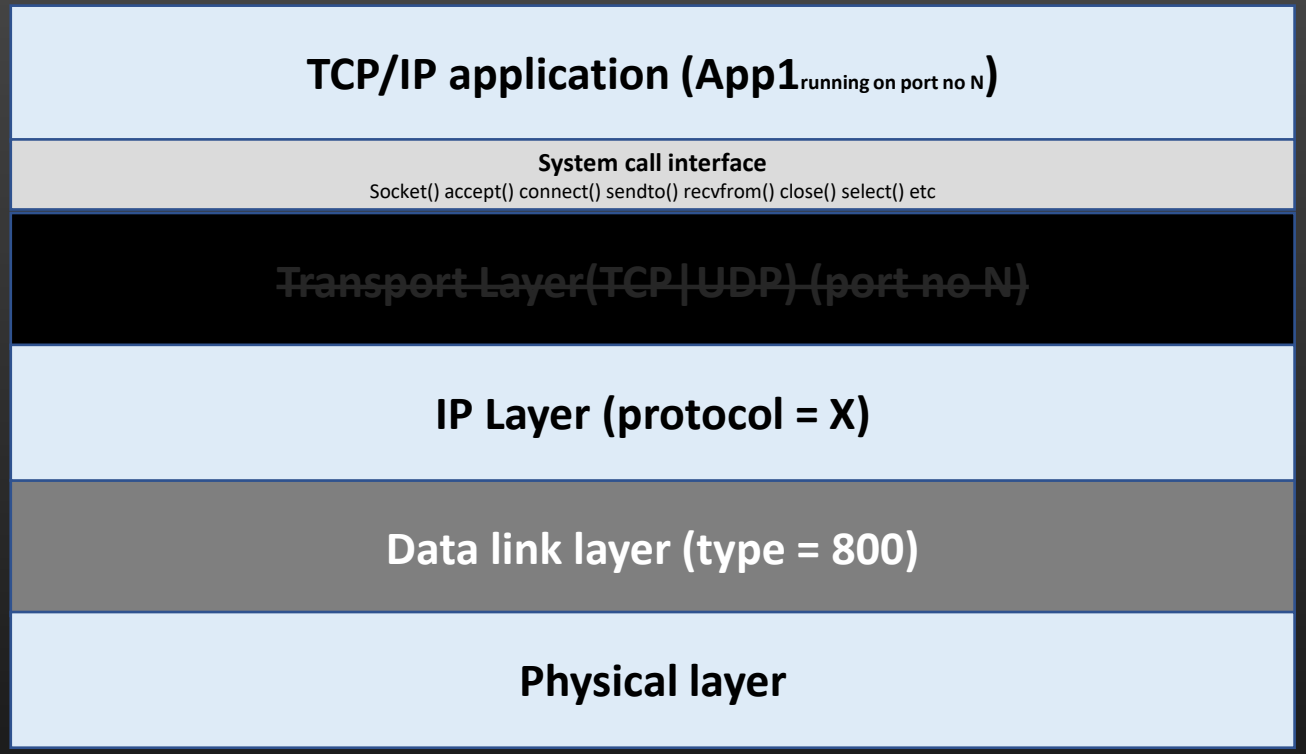
OSI Model | TCP/IP stack



Lecture VDO 9

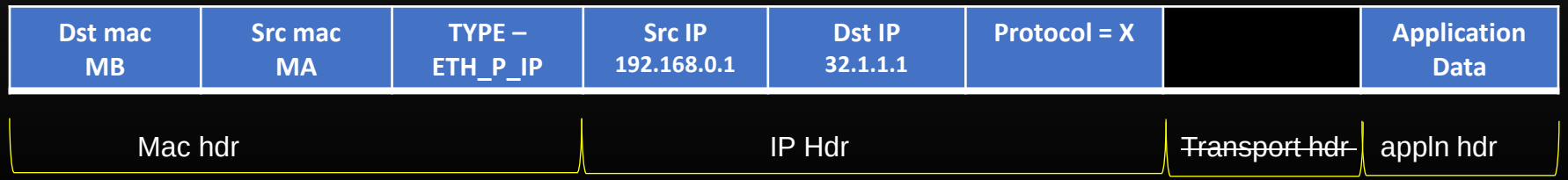
Application Layer

- When Application Protocol is run :
  - On top of Network Layer Protocol



OSI Model | TCP/IP stack

- It means, there is no Transport layer protocol
- Application defines its own protocol value X
- Network Layer takes the responsibility for packet transmission from source to destination machine
- Network Layer takes the responsibility to handover the packet to application based on protocol field value X
- Data link layer takes the responsibility for packet transmission hop by hop

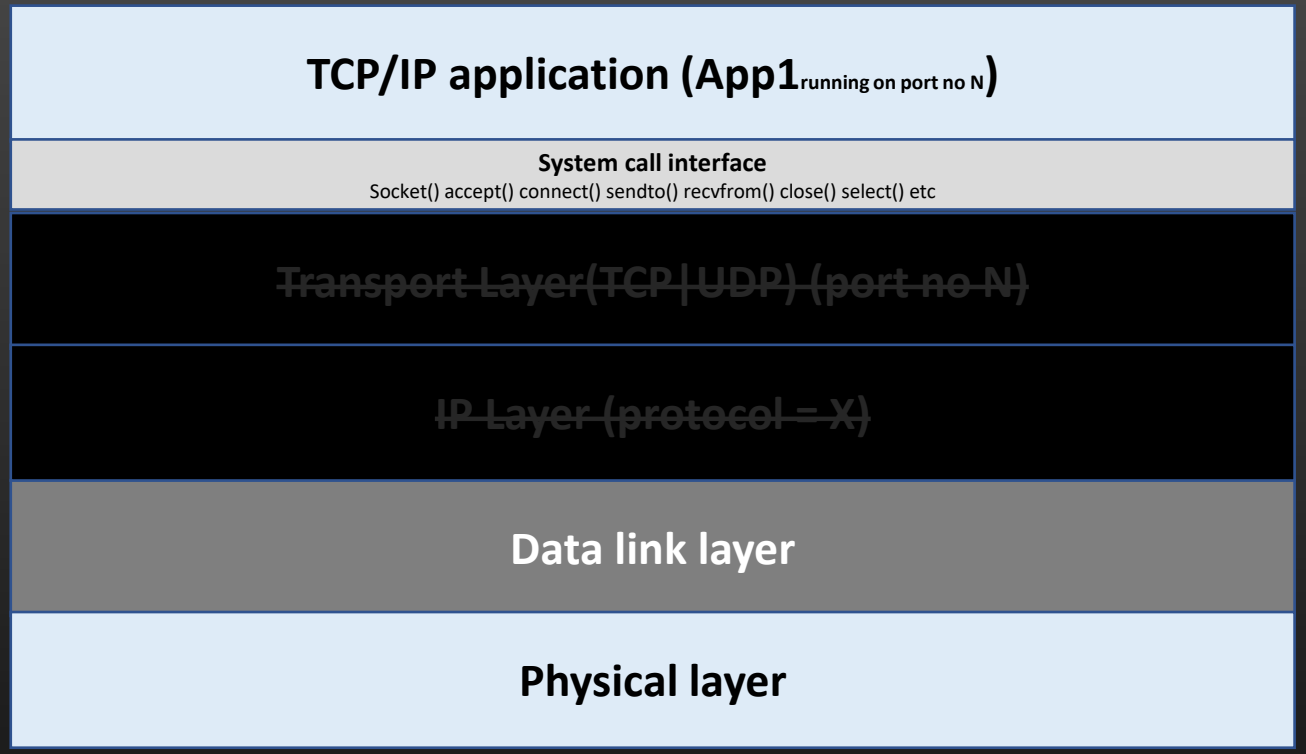


Lecture VDO 9

Application Layer

- When Application Protocol is run :
  - On top of Datalink layer Protocol

- It means, there is no Transport layer protocol
- There is network Layer either
- No L3 routing
- Data link layer takes the responsibility for packet transmission from source to destination machine (source and dest machine have to be in same subnet)
- All machines in the same subnet are nbrs of each other, hence Data link layer is still doing its hop by hop work



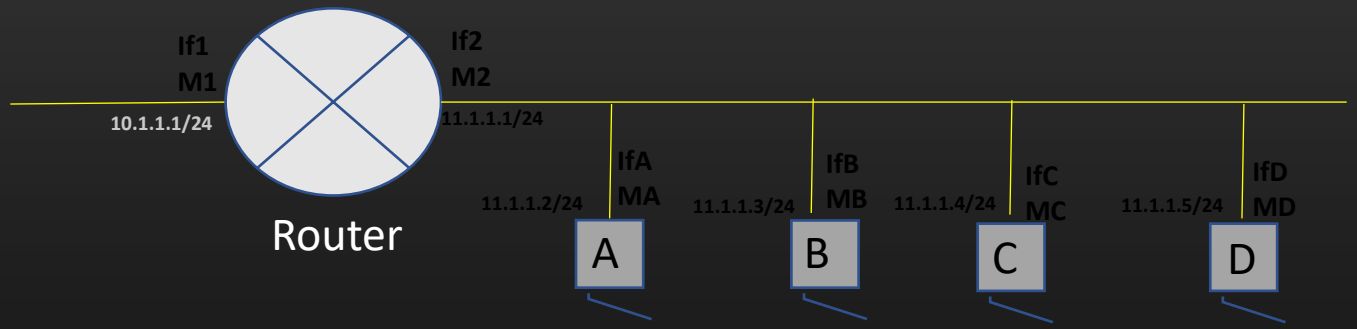
OSI Model | TCP/IP stack

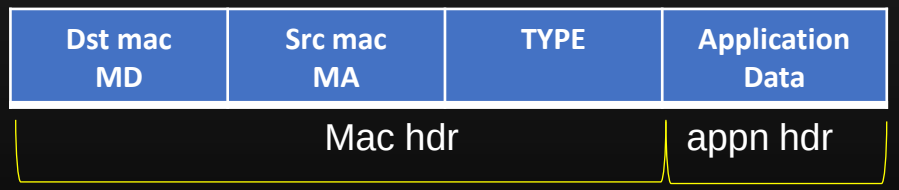
Packet Doesn't have L3 hdr ?
Lets discuss !

Lecture VDO 9

Application Layer -> Packet without L3 hdr

- When Sending machine A needs to send the packet to Destination machine D, and A already knows that D is in its own local subnet
  - Then Packet wont ever be routed by L3 router across subnets
  - Packet CAN be delivered to B using L2 routing alone (i.e. based on mac addresses only)
  - IP Hdr CAN be omitted out in such cases





Lecture VDO 9

Application Layer -> HTTP Web server Implementation

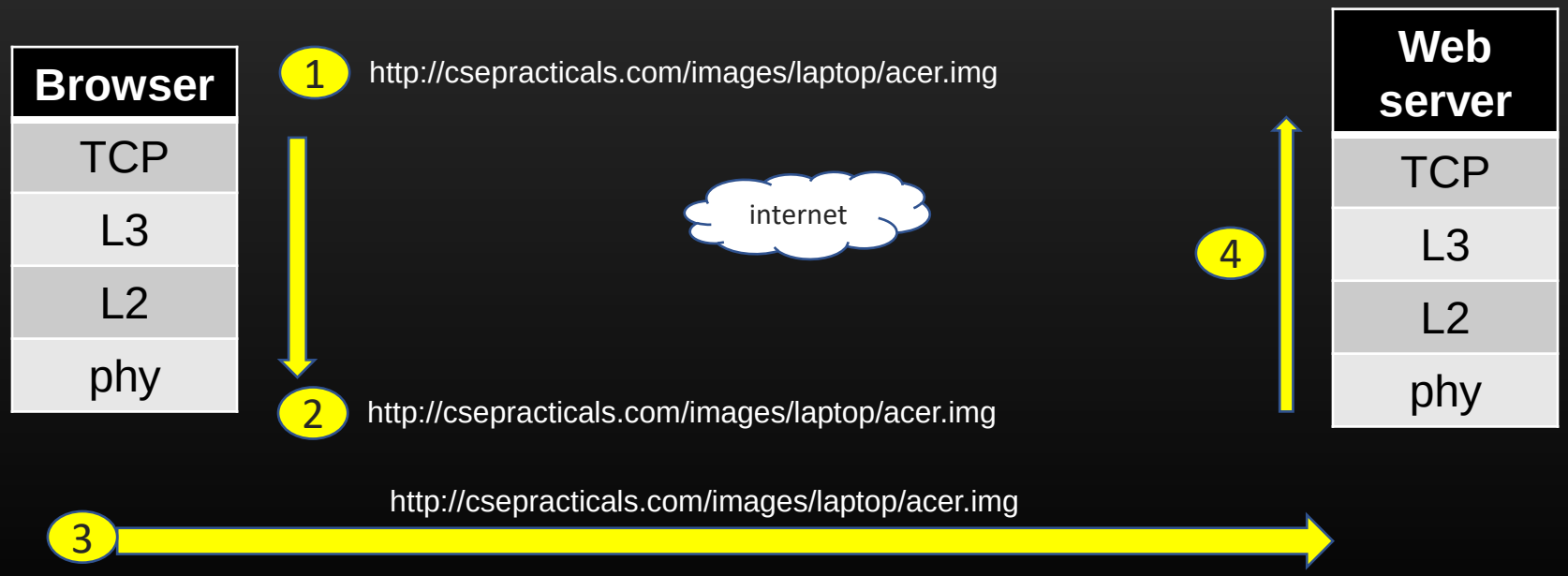
- Let us implement Application Protocol
  - We shall implement a simple client – Server program which implement our own application protocol
  - Let us learn how to implement Application protocol by implementing our own minimal HTTP web server
  - Let us first understand how the HTTP web server works

Lecture VDO 9

Application Layer -> HTTP Web server Implementation

The HTTP webserver works as per the following steps :

1. Client (the Browser prepares the HTML request message)
   This request msg is the application payload
2. The application payload is pushed down the TCP ip stack
3. Client machine sends the HTTP request to the server asking for some information
4. HTTP request is recvd by the server, and moves up the TCP ip stack

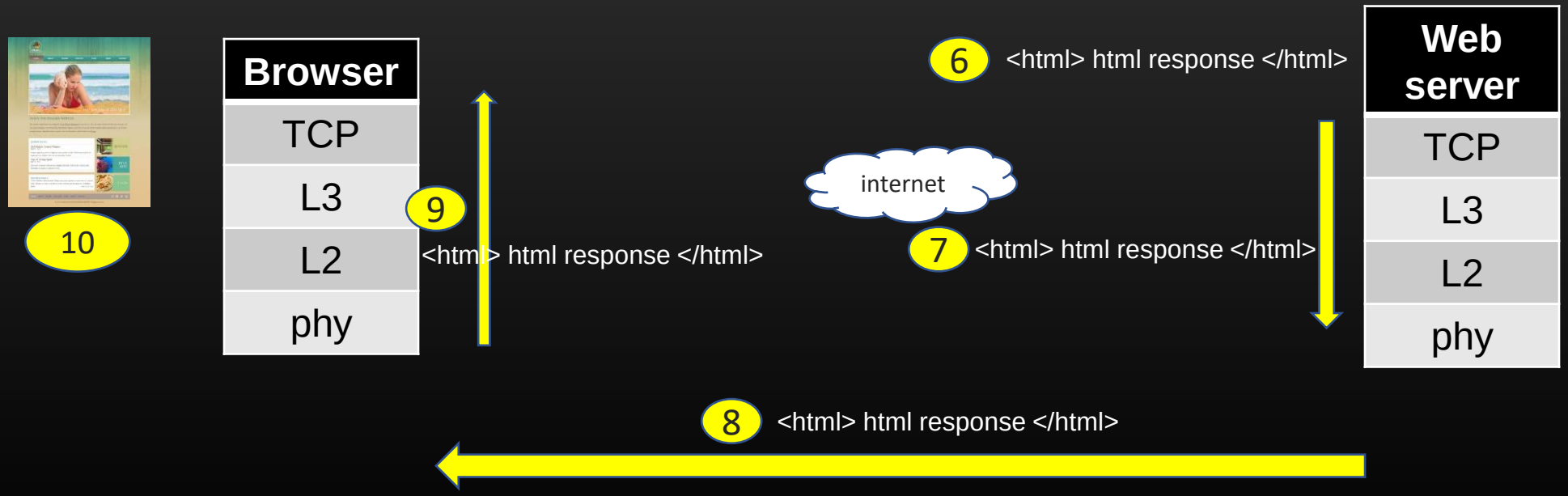


Lecture VDO 9

Application Layer -> HTTP Web server Implementation

The HTTP webserver works as per the following steps :

5. Server do the processing as per the client request
6. Server prepares the HTML response. This is appln payload from server side
7. Server push down the application payload down the TCP ip stack
8. Server sends HTML response back to client
9. This HTML response is processed by client TCP IP stack and delivered to Brower(application)
10. Browser renders the HTML page



Lecture VDO 9

Application Layer -> HTTP Web server Implementation

The HTTP client requests types :

- The request generated by HTTP client are of several types. We will going to discuss two most commonly used one's. You should Google as "HTTP request methods" to learn for more of them. Check resource section
    - GET request
    - POST request

- GET Request
    - Fetch the data from the webserver.
        - Example : Entering the roll no, and getting the Exam result
        - Link clicks to load/navigate to new page

- POST request
    - Sending the data to the web server, which will be saved on the server side
        - Example : Filling up job forms
        - Uploading resumes
        - Face book posts, comments etc

When HTTP client sends the HTTP request to the web server, Client has to encode the request type in the request.

Lecture VDO 9

Application Layer -> HTTP Web server Implementation

The HTTP message format :

- When HTTP client sends the HTTP request to server (GET or POST or any other), it sends the HTTP request in a defined/standard format
- Let us take the example of GET request
- Below is the HTTP GET request msg contents

```
GET /College/IIT/?dept=CSE&rollno=10305042 HTTP/1.1
Host: 192.168.56.100:80
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.79 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

- You can see, HTTP msg types are simple plain text strings.

Lecture VDO 9

Application Layer -> HTTP Web server Implementation

The HTTP message format :

- The HTTP Webserver replies to the HTTP client with the HTTP response, which is html web page
- Below is the HTTP response msg format returned by webserver to the client

```
HTTP/1.1 200 OK
Server: IIT_STUDENT_SERVER
Content-Length: 88
Content-Type: text/html; charset=UTF-8

<html>
                … HTML web page code . . .
</html>
```

Feeling a bit confused ?? Ok, lets do the actual implementation . ..

Lecture VDO 9

Application Layer -> HTTP Web server Implementation -> code walk

Code Walk . . .

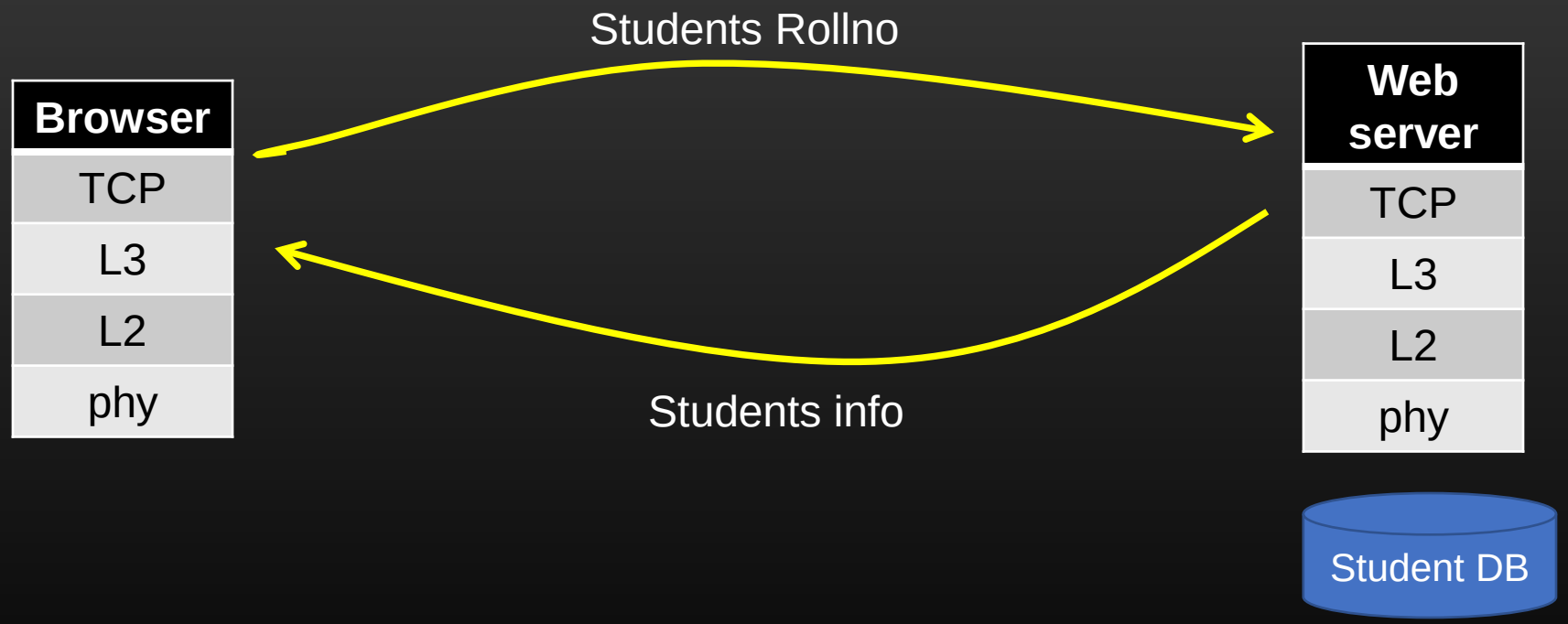
- Implementing the HTTP webserver is no different than implementing a normal tcp server
- The only difference is that – HTTP server expects the client data as per HTTP standards & prepare the response as per HTTP standards only.
- Prerequisite to understand the HTTP server code :
  - Little bit of HTML
  - How ***strtok()*** in C works

Code location : git clone http://github.com/csepracticals/SerVerDesign
file : SerVerDesign/webserver/TCPWebserver/tcp_web_server.c

Lecture VDO 9

Application Layer -> HTTP Web server Implementation -> code walk

Code Walk . . .



Lecture VDO 9

Application Layer -> HTTP Web server Implementation -> Observation

Observation :

- We implemented the HTTP web server which could understand only the HTML client requests
- Our HTTP webserver runs as an Application on top of TCP transport layer protocol. Thus, HTTP is an application layer protocol.
- If, some packet in the network is lost for some reason , TCP protocol will take care to make up for the lost packets (TCP functionality)
- HTTP webserver parse the client's URL and accordingly take the action to be performed
- HTTP webserver returns the HTML code, because HTTP clients – the browsers understands HTML and render it (display)
- HTTP protocol message types are simple plain text strings, hence, we need to struggle through string manipulation
- This is how Internet browsing works, any link click sends GET request to the HTTP server, HTTP server prepares the HTML code of the web page pointed to by the link , and sends back to the client
- Client (browser) loads the page.

Lecture VDO 9

Application Layer -> HTTP Web server Implementation->Assignment

- I have shown you how to implement the HTTP GET request

- You assignment is :

Implement the HTTP POST request and add the details of one of the student to server local database. Server can return Success or fail to the client(Browser) as a result.

Assignment location :

git clone http://github.com/csepracticals/SerVerDesign
file : SerVerDesign/webserver/TCPWebserver/tcp_web_server.c

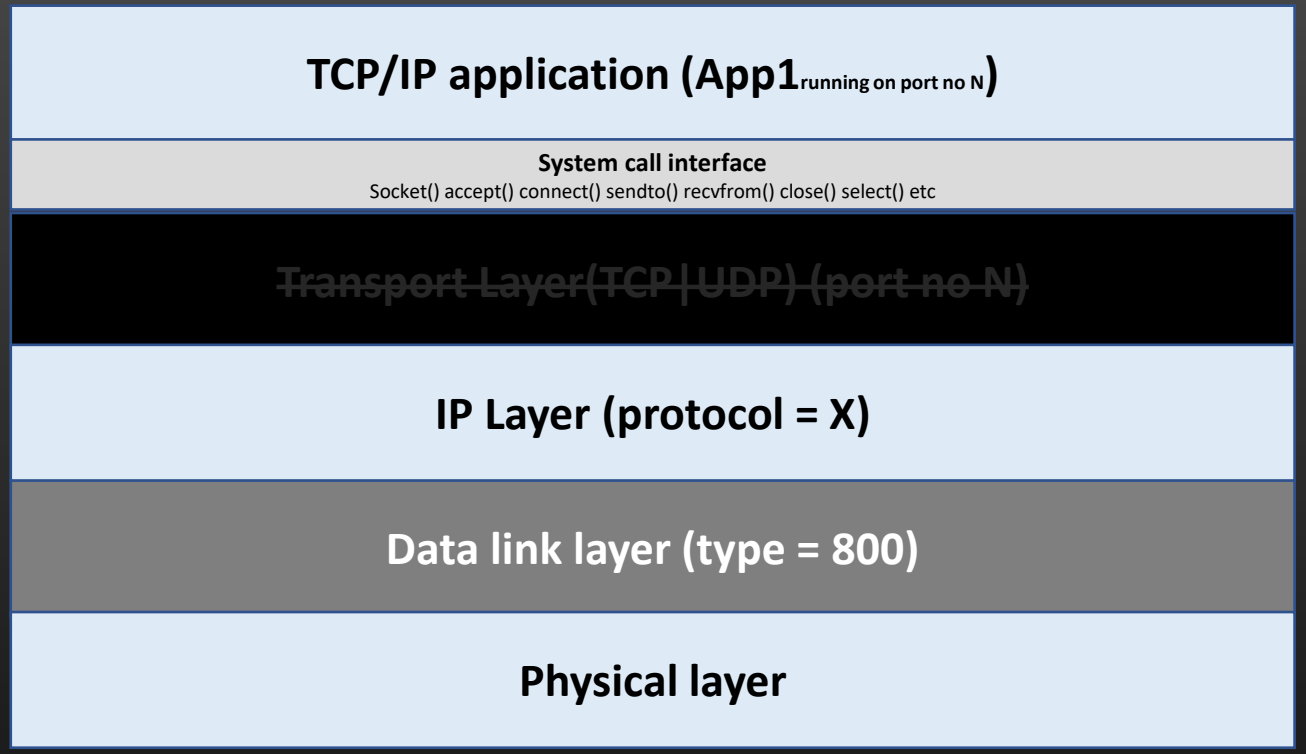
Implement the function :

```
static char *
process_POST_request(char *URL, unsigned int *response_len){

    return NULL;
}
```

Lecture VDO 9

Application Layer -> User defined Application Protocol

- We had just seen an example of our web server which was an application running on top of TCP/IP protocol. Next we will see how to run the application directly:
  - On top of Network Layer Protocol
- When we run our application without the transport layer, the IP layer itself work as transport layer for our application protocol
- We need to define a new value X to our application protocol in this case
- Sending machine : All the packets generated by our application will have X specified in the *protocol* field in the IP hdr
- Receiving machine : When Network layer sees X in the protocol field in the IP hdr, Network layer checks which application is interested in receiving such packets. Network layer, then handover the packet to such applications (Remember this job was being done based on port number by transport layer)
- This is actually very easy to achieve . . . Let's see how ?



OSI Model | TCP/IP stack

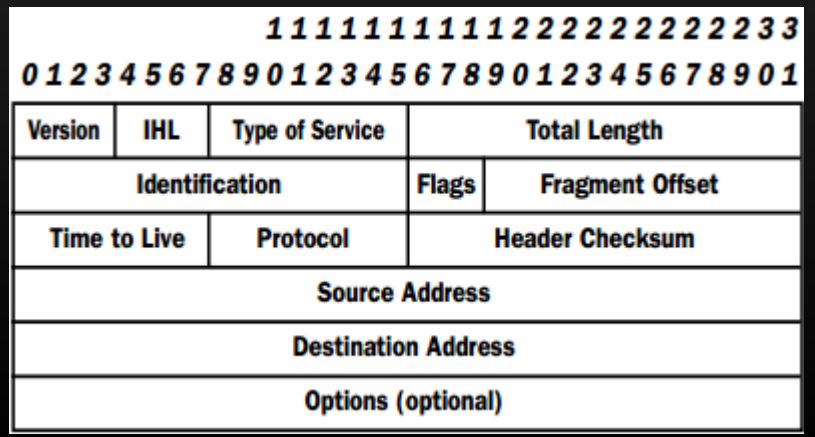
Lecture VDO 9

Application Layer -> User defined Application Protocol

- Implementing your own Application on top of network layer of TCP/IP stack
- The only thing you need to do is to Change the 2nd and 3rd argument of *socket()* sys call on server and client side

```
socket (AF_INET,  SOCK_RAW,  <YOUR APPLN PROTO NUMBER N>))
```

- *SOCK_RAW* : Allows to bypass the Transport layer , application communicate directly with network layer
- *YOUR APPLN PROTO NUMBER 'N' :* Your own application protocol should be identified by unique identifier (like TCP by 6, UDP by 17 etc)
- When Server sends the data , the *protocol* field in the ip hdr is *N*
- *When client receives the packet in which protocol* field is N, the network layer delivers the packet to the application which has open the socket with protocol identifier as N
- The application receives an (IP HDR + application payload) from network layer
- Your Own application protocol would want to have set of its own sub messages. You must use concept of Header cascading to define and use application specific sub messages



Demo ….

Lecture VDO 9

Application Layer -> User defined Application Protocol -> Demo

How Network Packets are cooked using Sockets ?

Application Layer

Thank you

Socket Programming Project

Socket Programming Project

Distributed
Transparent
Memory

Distributed Transparent Memory

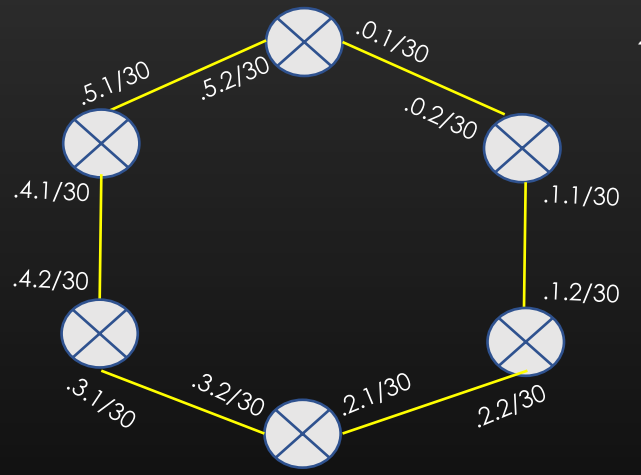
Project Goals

- Many nodes shared their local chunk of memory to form one big chunk of memory
- Memory is distributed across nodes, but user is not aware of this distributed-ness (Transparent)
- Such memory model is used when voluminous amounts of data needs to be stored on several machines
  Example : Hadoop Distributed File System (HDFS) is built on this model
- We will built the distributed memory while user would not have to keep track of the information about where data is located or being fetched from

Socket Programming -> Project

- **Project deployment**
- To deploy your project
  - Create 6 node topology as we created in multi-node environment using your virtualization software

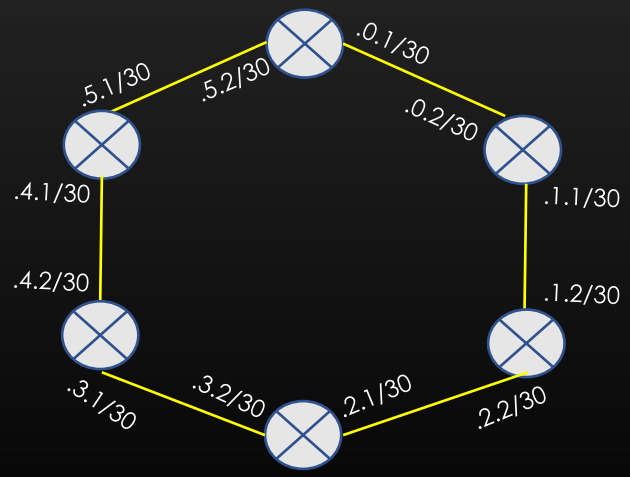
Node #	Lo1 IP	TCP port	UDP port
0	10.0.0.1/32	2000	2001
1	10.0.0.2/32	2002	2003
2	10.0.0.3/32	2004	2005
3	10.0.0.4/32	2006	2007
4	10.0.0.5/32	2008	2009
5	10.0.0.6/32	2010	2011



192.168.x.x/30

Socket Programming -> Project

- **Project deployment**
- To deploy your project
  - Node i knows the node j's lo ip address and UDP port no, where j = i + 1. That is,
    - node 3 knows node 4 lo ip-address and UDP port no
    - node 4 knows node 5 lo ip-address and UDP port no
    - node 5 knows node 0 lo ip-address and UDP port no and so on . . .
  - Install L3 routes in routing table of all nodes in the topology. You should be able to ping every other nodes lo address from any node. No need to install L3 routes for interface subnet ip addresses, Only loopbacks suffice.



192.168.x.x/30

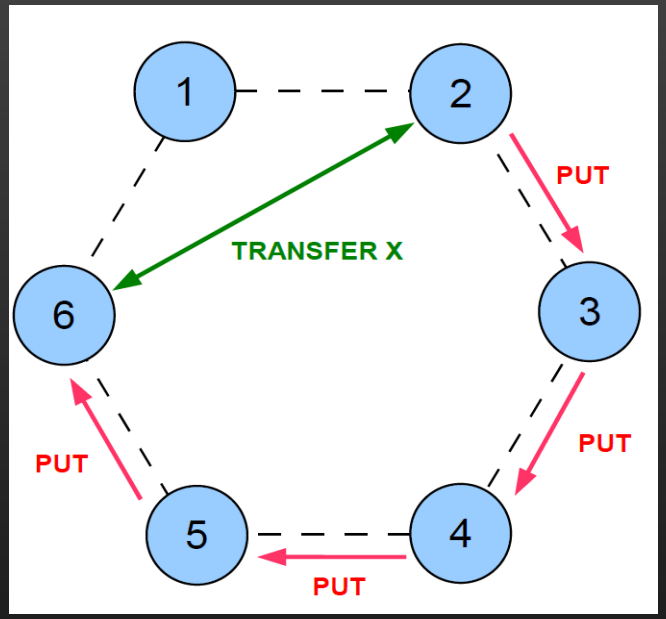
Socket Programming -> Project

- **Problem statement**

  - You have n (= 6 in eg) server machines/nodes
  - User can perform two operations on each node :
  - Put K X, where k and X are two integers, K = key, X = value
  - Get K, where K is an integer, returns X for K
  - *Originator* is a node on which user has issued put/get request

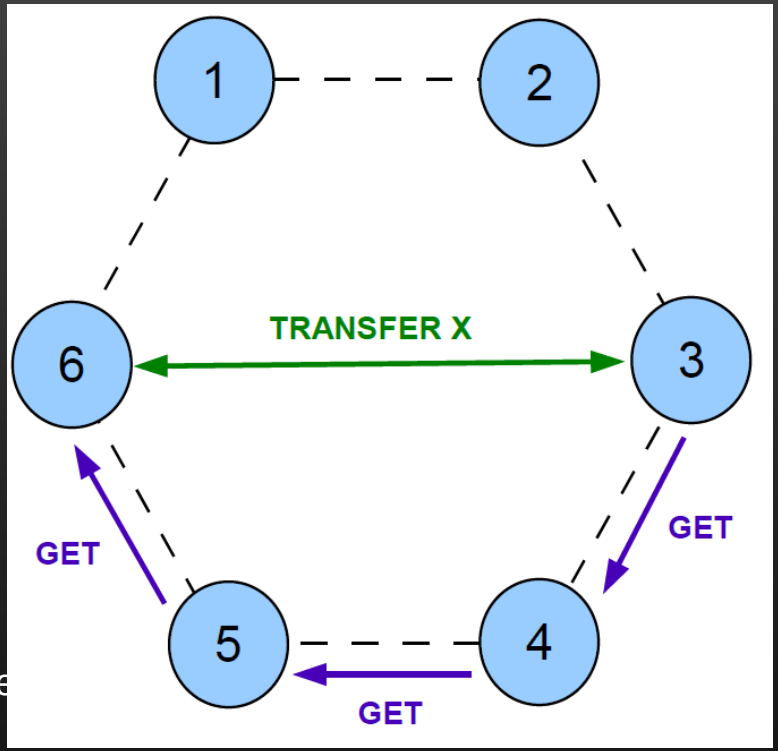
  - Eg - Put request

    - Put k = 3 , x = 5 is performed on node 2
    - Node 2 perform hash operation on K ,i.e h(3)
    - Let h(3) returns 6. So, X = 5 needs to be saved on node 6's hash table against key k = 3
    - So, now node 2 knows it do not have to save k =3 , X= 5 in its own hash table, hence it forwards K to node 3 using UDP communication (msg id : PUT_FORWARD).
    - Node 3 repeats the process
    - Node 3, too forwards K to node 4 (using UDP), and so on until node 6 received K.
    - Node 6 will know it has to save K = 3, X = ? in local hash table because, h(3) = 6. But node 6 do not know value of X
    - Node 6 establish TCP connection with original originator of Put request – i.e. node 2 and asks for value of X (*But how node 6 would know that node 2 was the req originator ?*) (msg id : WHAT_X).
    - Node 2 sends X to node 6 directly now as TCP reply (msg id : PUT_REPLY_X).
    - Node 6 saves K = 3, X = 5 in its hash table and closes the TCP connection



Socket Programming -> Project

- **Problem statement**
  - You have n (= 6 in eg) server machines/nodes
  - User can perform two operations on each node
  - Put K X, where k and X are two integers, K = key, X = value
  - Get K, where K is an integer, returns X for K
  - *Originator* is a node on which user has issued put/get request
  - Eg - Get request
    - get k = 3 is performed on node 3
    - Node 3 perform hash operation on K ,i.e h(3)
    - Let h(3) returns 6.
    - So, now node 3 knows it do not have to fetch X against K from its local hash table, hence forwards K to node 4 using UDP (msg id : GET_FORWARD).
    - Node 4 repeats the process
    - Node 4, too forwards K to node 5 (using UDP), and so on until node 6 received K.
    - Node 6 will know it has to fetch X against K = 3 from its local hash table because, h(3) = 6.
    - Node 6 now knows K = 3, X = 5
    - Node 6 establish TCP connection with original originator of Get request – i.e. node 3 and reports to node 3 the value X = 5 (msg id : GET_REPLY_X).
    - Node 3 closes TCP connection with node 6 after receiving X = 5



Socket Programming -> Project

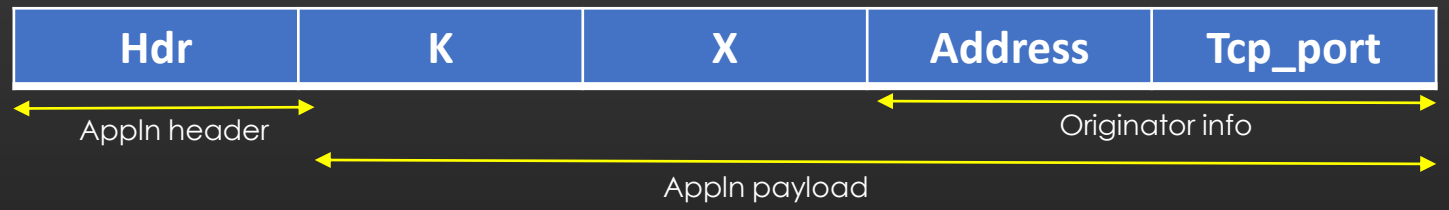
- So, there are five application msg types. Node when receives these msg should take the following action as explained.

Msg id	Action when received
PUT_FORWARD	If h(k) == self if self == originator save K,X /*K,X saved on the same node on which admin triggered put req*/ else send WHAT_X to originator /*Ask the originator for value X over TCP*/ else send PUT_FORWARD to successor node over UDP
WHAT_X	Send PUT_REPLY_X to the sender of “WHAT_X” over TCP
PUT_REPLY_X	Save K,X and return
GET_FORWARD	If h(k) == self if self == originator print K, X /*X was available on local node on which admin ran GET req*/ else send GET_REPLY_X to originator over TCP else send GET_FORWARD to successor node over UDP
GET_REPLY_X	Print K, X

Socket Programming -> Project

- Our application headers and exchange messages can be structured as follows:

```
typedef struct app_hdr_{

        unsigned int msg_id;
} app_hdr_t;

typedef struct appln_msg_{

        unsigned int k;
        unsigned int x;
        unsigned int address;
        unsigned int tcp_port
} appln_msg_t;
```



Hdr value	Contents of various fields in the message			
PUT_FORWARD	K	--	Address	Tcp_port
GET_FORWARD	K	--	Address	Tcp_port
WHAT_X	K	--	--	--
PUT_REPLY_X	K	x	--	--
GET_REPLY_X	K	x	--	--

Note : *unsigned int address* can be encoded as unsigned integer equivalent of IP address
Depending on hdr value, the receiving machine would know how to read the trailing msg bits followed after hdr

Socket Programming -> Project

- Project pseudocode :

```
        tcp_master_fd = socket ( . . . );

        udp_socket_fd = socket(. . . );

While(1){

        FD_SET(tcp_master_fd, &readfds);

        FD_SET(udp_master_fd, &readfds);

        FD_SET(0, &readfds);   /*console*/

        select(&readfds, . . . );

        if(FD_IS_SET(tcp_master_fd, &readfds){
                process msg m , where m = WHAT_X Or GET_REPLY_X Or PUT_REPLY_X
        }
        else if(FD_IS_SET(udp_master_fd, &readfds){
                 process msg m , where m =  PUT_FORWARD or GET_FORWARD
        }
        else if(FD_IS_SET(0, &readfds){
                /*User put/get request entered */
        }
} // while ends
```

Socket Programming -> Project

- **Points to Note**
- h(k) – is a hash fn which returns node no. for some K. You can take h(k) = k mod N , where N = total no of nodes
- Each node is a TCP Server/UDP Server and TCP client/UDP client also. So, you should open TCP master skt, UDP socket and monitor both in select system call. Also, Your node should also be able to accept user Put/get request from console. Console acts as a client with comm_fd = 0. So, always keep 0 as member to your readfds set.
- Use *scp* utility command to copy file from one VM to another.
- Good luck guys … Take your time to understand/design your application…

DNS

Domain Name System

DNS

Agenda

1. What is DNS and Why we need it (Problem Statement) ?
2. DNS Architecture and Design
3. FQDN - Fully Qualified Domain Name
4. First Hop DNS Server (FHDNS)
5. How DNS Work - Step by Step
   Recursive Query
   Iterative Query
6. Reverse DNS
7. Summary

Domain Name Server (DNS)

Introduction

1. Domain Name Server is popularly acronymed as DNS

2. DNS technology facilitates the mapping of Website name to IP-address and IP-address to Website Name
   a. Website Domain Name -> IP-address of machine which hosts the website
      > This is functionality of DNS
      > We do this all the time

   b. IP-Address -> Website Domain Name
      > This is functionality of Reverse DNS
      > Web host server corresponds to 8.8.8.8 is google.com

3. For example :
   www.google.com -- > website hosted on Server whose ip address is 8.8.8.8

   Without DNS, you would have had been typing : https://8.8.8.8 in your browser to reach google.com Servers.

4. In this Module We shall Learn the problem statement which DNS mechanism solves, and how DNS mechanism works

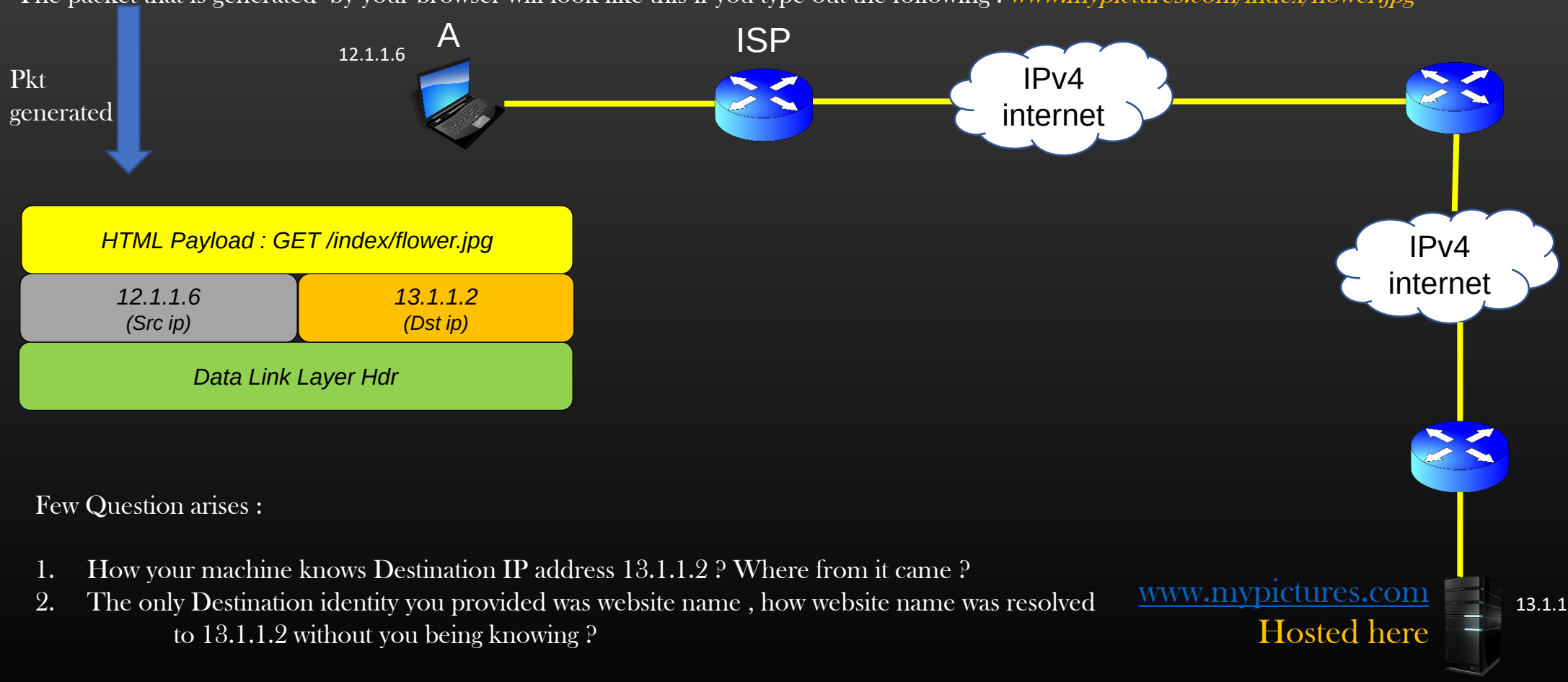
5. DNS is like a phone book for the internet !!

6. DNS plays its role whenever you visit some website Or sending an email

Problem Statement

- Whenever you type out the name of any website in your browser, you are trying to access that particular website which is hosted somewhere on the network

- The packet that is generated by your browser will look like this if you type out the following : *www.mypictures.com/index/flower.jpg*



Few Question arises :

1. How your machine knows Destination IP address 13.1.1.2 ? Where from it came ?
2. The only Destination identity you provided was website name , how website name was resolved to 13.1.1.2 without you being knowing ?

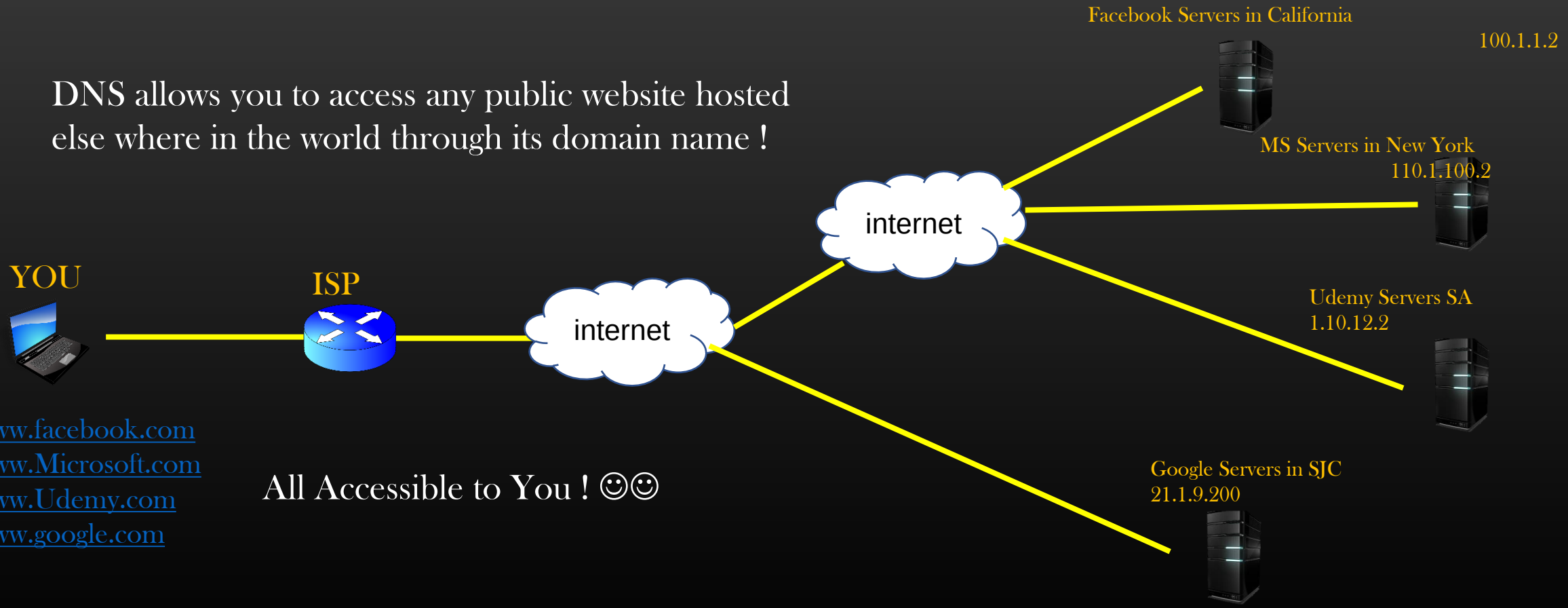
The Above Questions are answered by DNS

Domain Name Server (DNS)

What is DNS ?

- DNS is application layer protocol, and it works completely transparent to the end user
- As am end user, you never need to learn which website is hosted on which Server in the world, and it is not possible either
- We always access the various websites through their names (called Domain names), Human cant remember several IP addresses

DNS allows you to access any public website hosted else where in the world through its domain name !

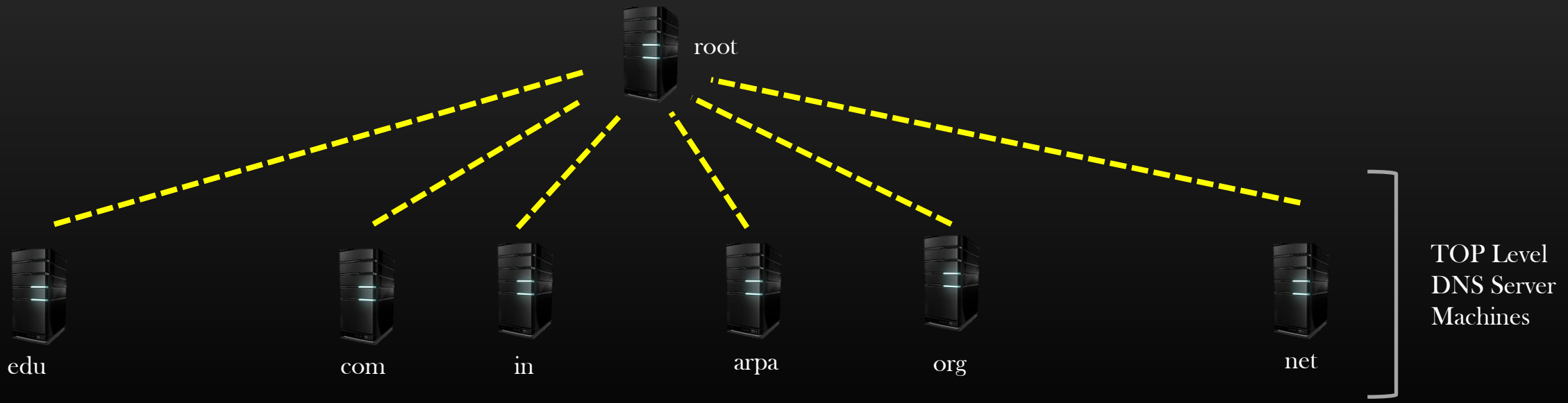


www.facebook.com
www.Microsoft.com
www.Udemy.com
www.google.com

All Accessible to You ! ☺☺

DNS System Architecture

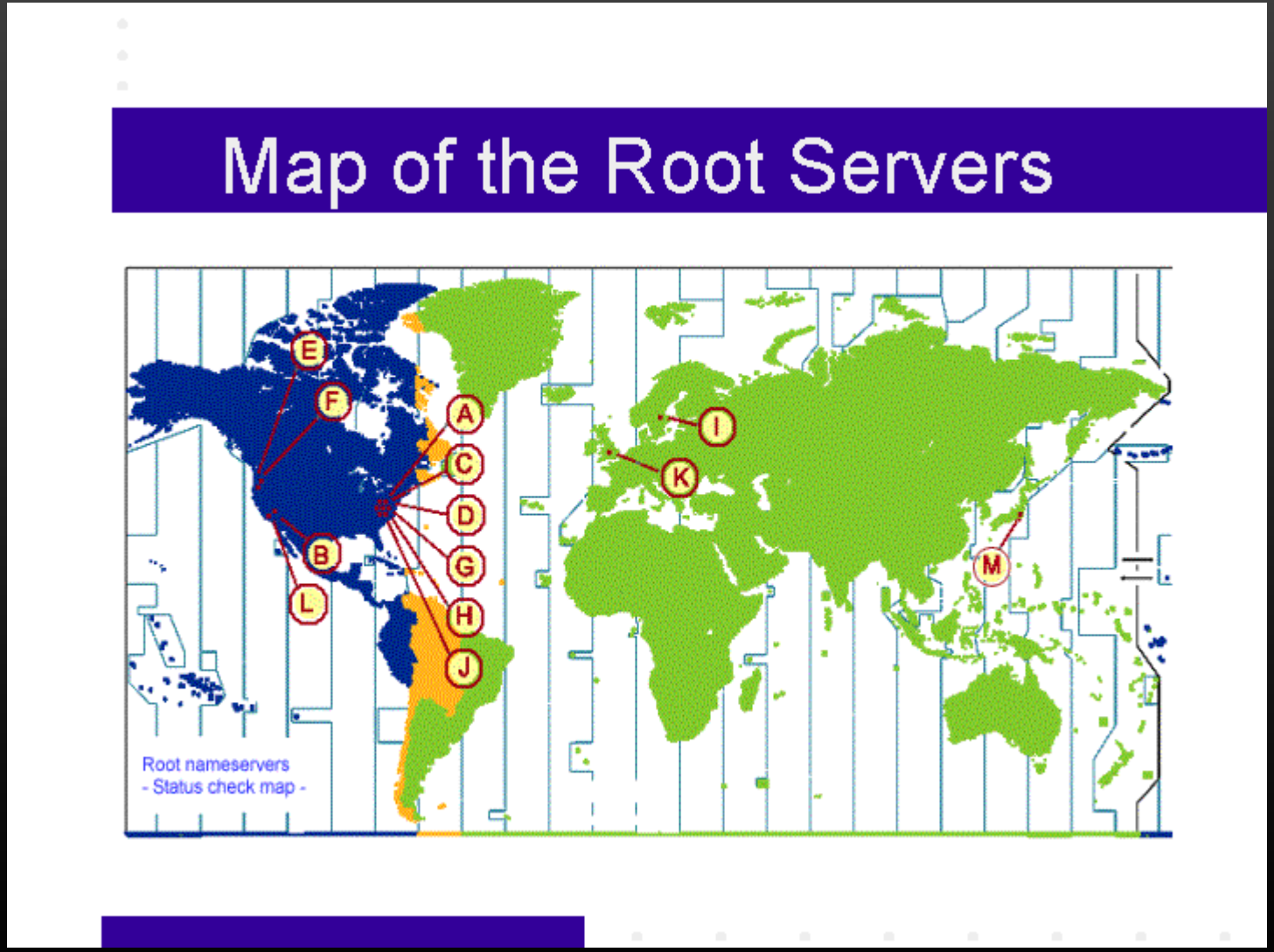
- DNS System is a groups of Servers called DNS Servers which works in Collaboration with each other to implement DNS functionality
- These DNS Servers Machines are distributed all across the globe with sufficient redundancy
- DNS machines are monitored & managed by central authority
- Global DNS Server machines (which comes under central authority) works at two levels :
    - ROOT LEVEL - Servers which work are root level are called root DNS Servers
    - TOP LEVEL - Servers which work at TOP level are called Top Level DNS Servers (TLDs)



DNS Server Geographical Distribution

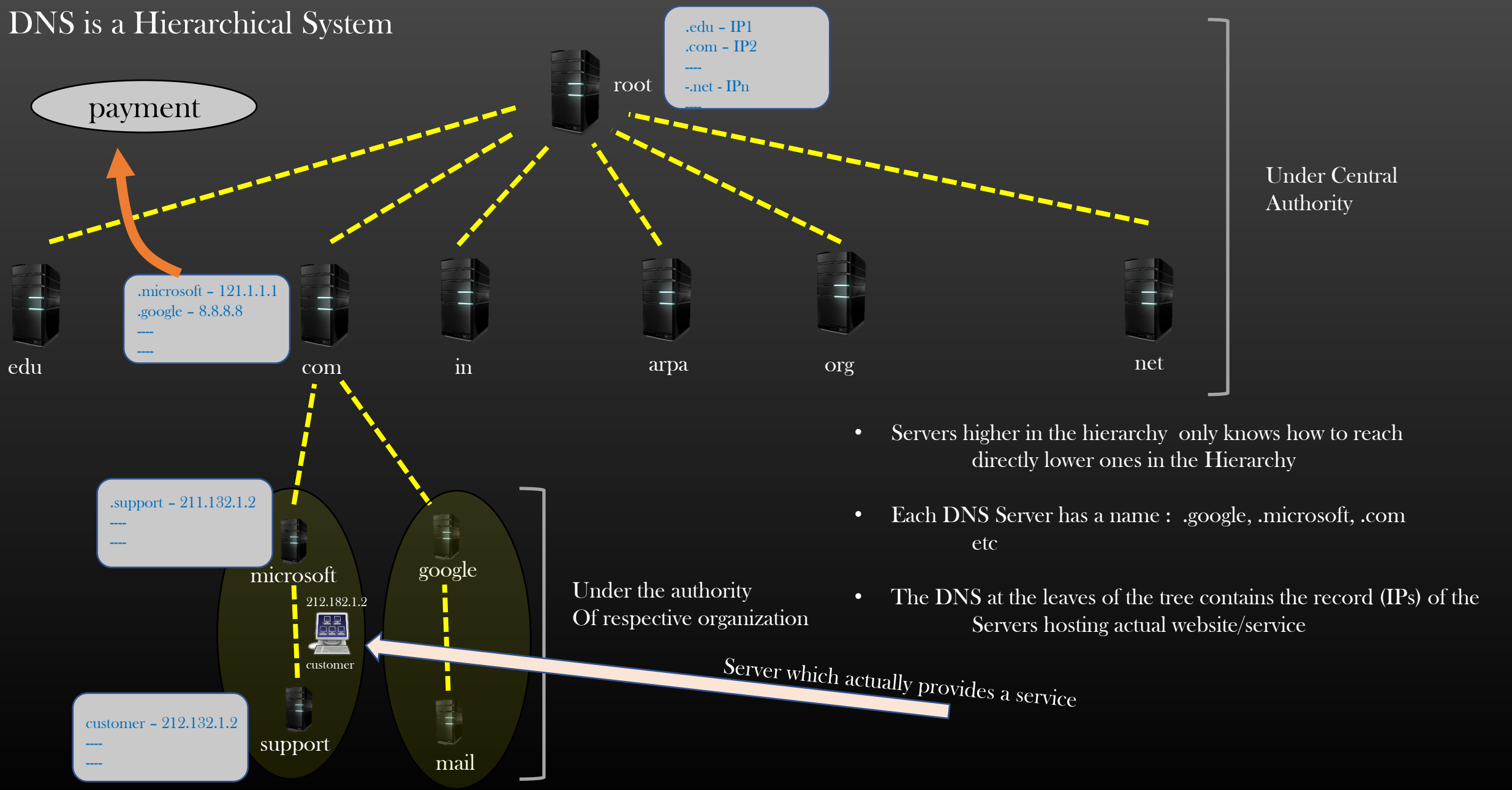
- DNS Servers (root and TLDs) are Distributed across the globe

- Redundancy
  - Fault tolerance
  - Load balancing



Domain Name Server (DNS)

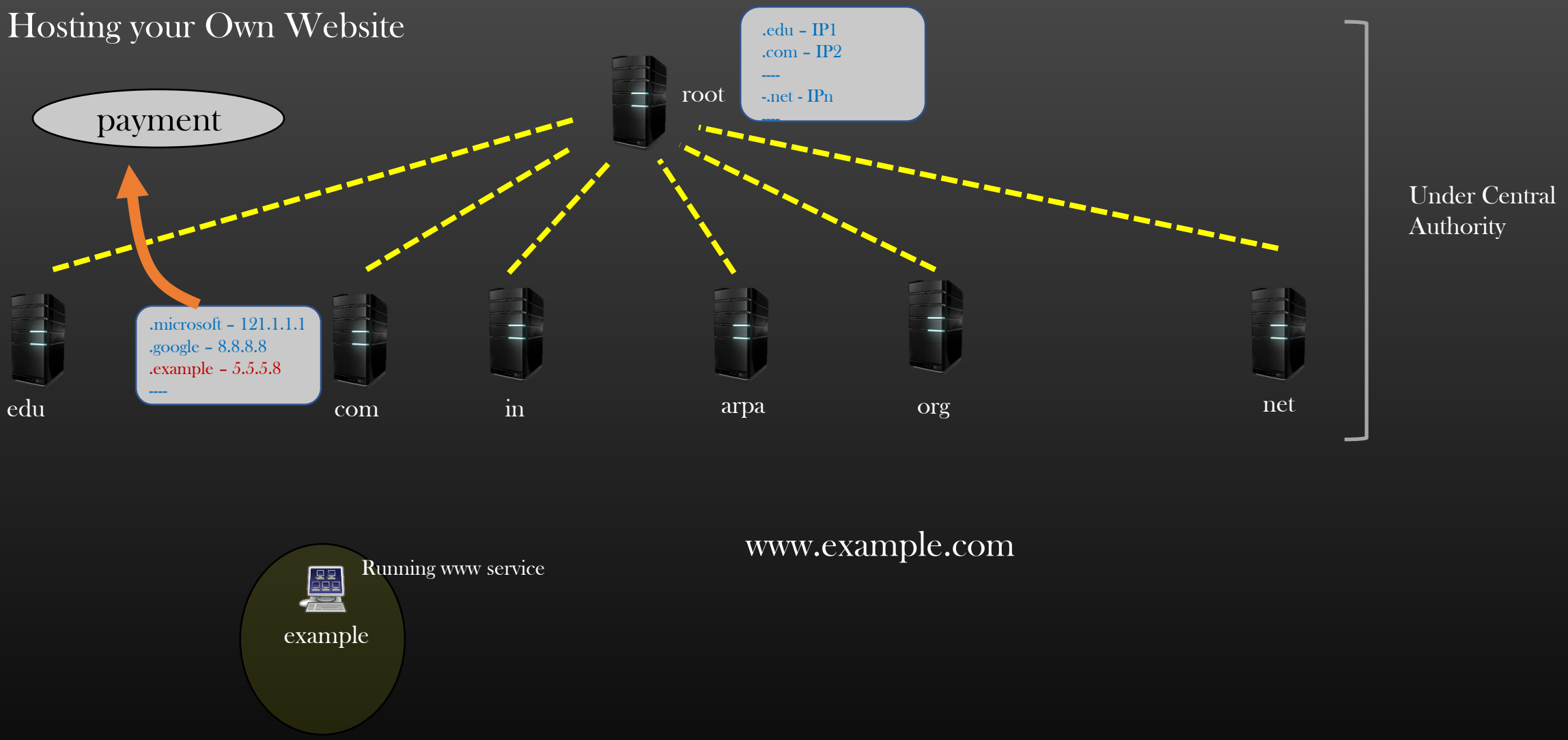
DNS is a Hierarchical System



- Servers higher in the hierarchy only knows how to reach directly lower ones in the Hierarchy
- Each DNS Server has a name : .google, .microsoft, .com etc
- The DNS at the leaves of the tree contains the record (IPs) of the Servers hosting actual website/service

Domain Name Server (DNS)

Hosting your Own Website



Website's Domain Name

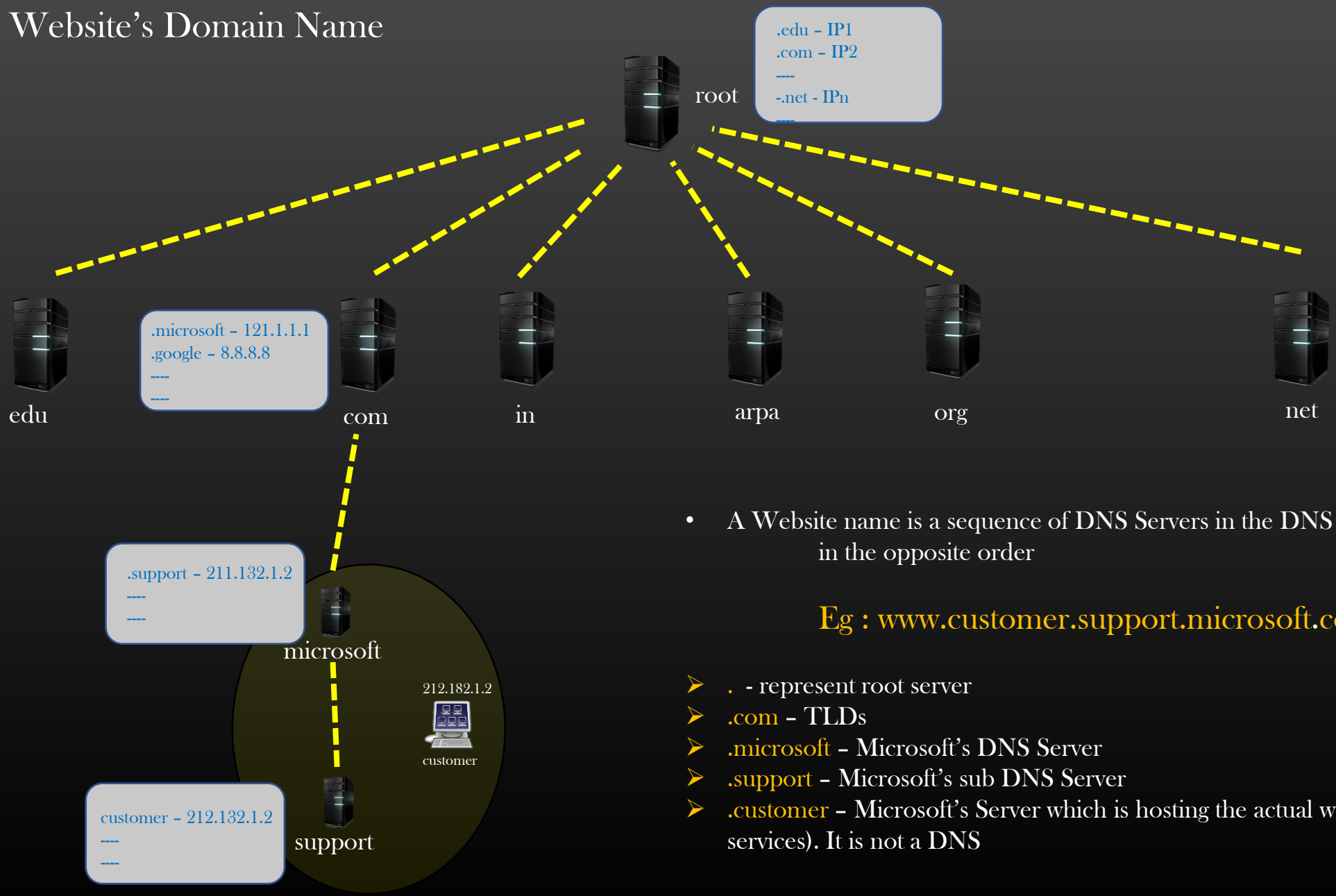
- A website name , also called, domain name is not just any random sequence of words
- A Website name needs to be constructed systematically keeping DNS hierarchy in Mind
- A website name is constructed of individual words called *literals*. These Literals are actually the name of DNS Servers in the path of the DNS tree starting from root towards the leaf of DNS tree

  Eg : customer.support.microsoft.com - - consists of 4 *literals*

- Let us see, how website domain name is constructed Or derived from DNS system with the help of example

Domain Name Server (DNS)

Website's Domain Name



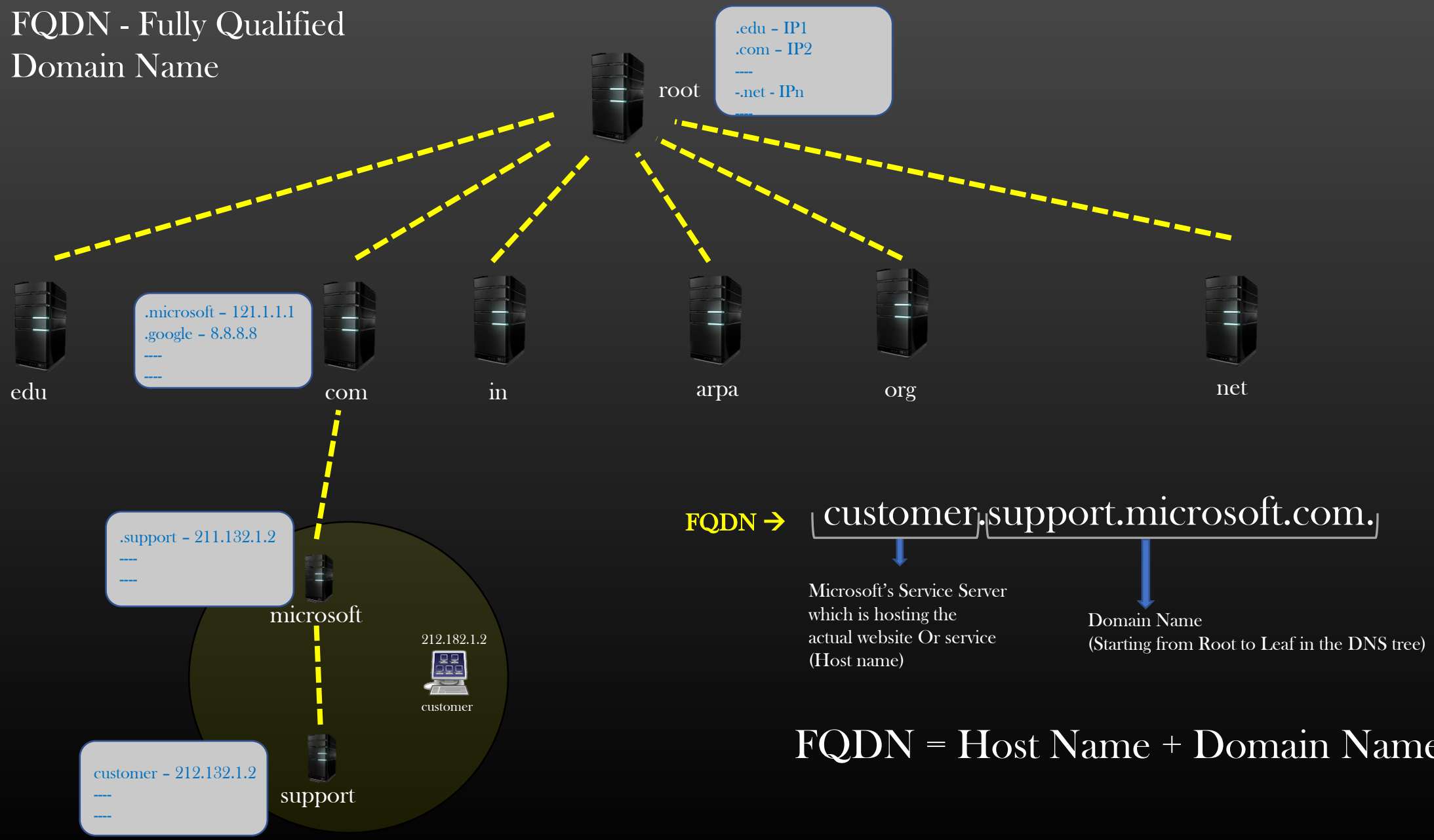
- A Website name is a sequence of DNS Servers in the DNS tree but in the opposite order

Eg : www.customer.support.microsoft.com

- . - represent root server
- .com – TLDs
- .microsoft – Microsoft's DNS Server
- .support – Microsoft's sub DNS Server
- .customer – Microsoft's Server which is hosting the actual website (www services). It is not a DNS

Domain Name Server (DNS)

FQDN - Fully Qualified Domain Name



FQDN → customer.support.microsoft.com.

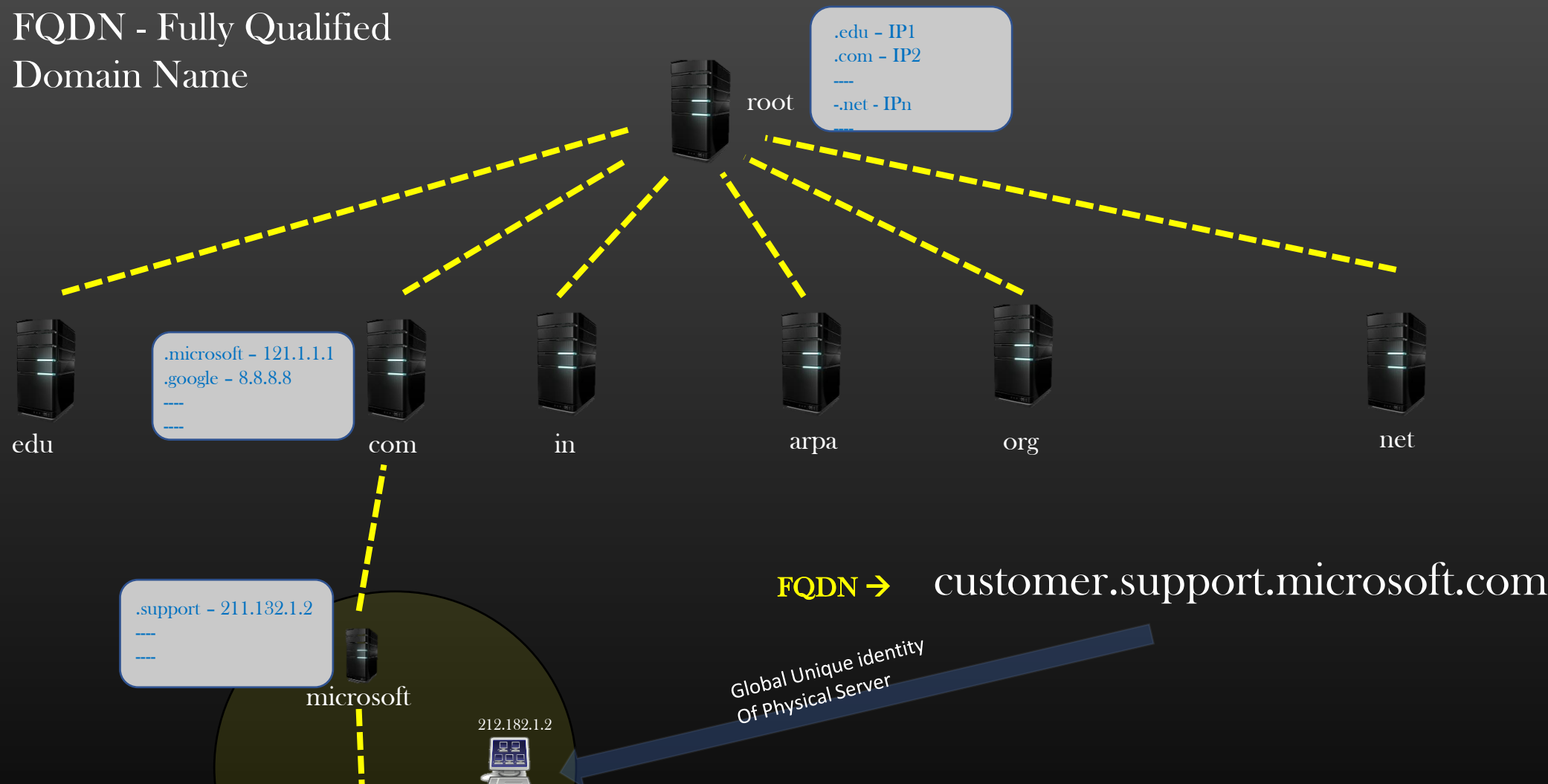
Microsoft's Service Server which is hosting the actual website Or service (Host name)

Domain Name (Starting from Root to Leaf in the DNS tree)

FQDN = Host Name + Domain Name

Domain Name Server (DNS)

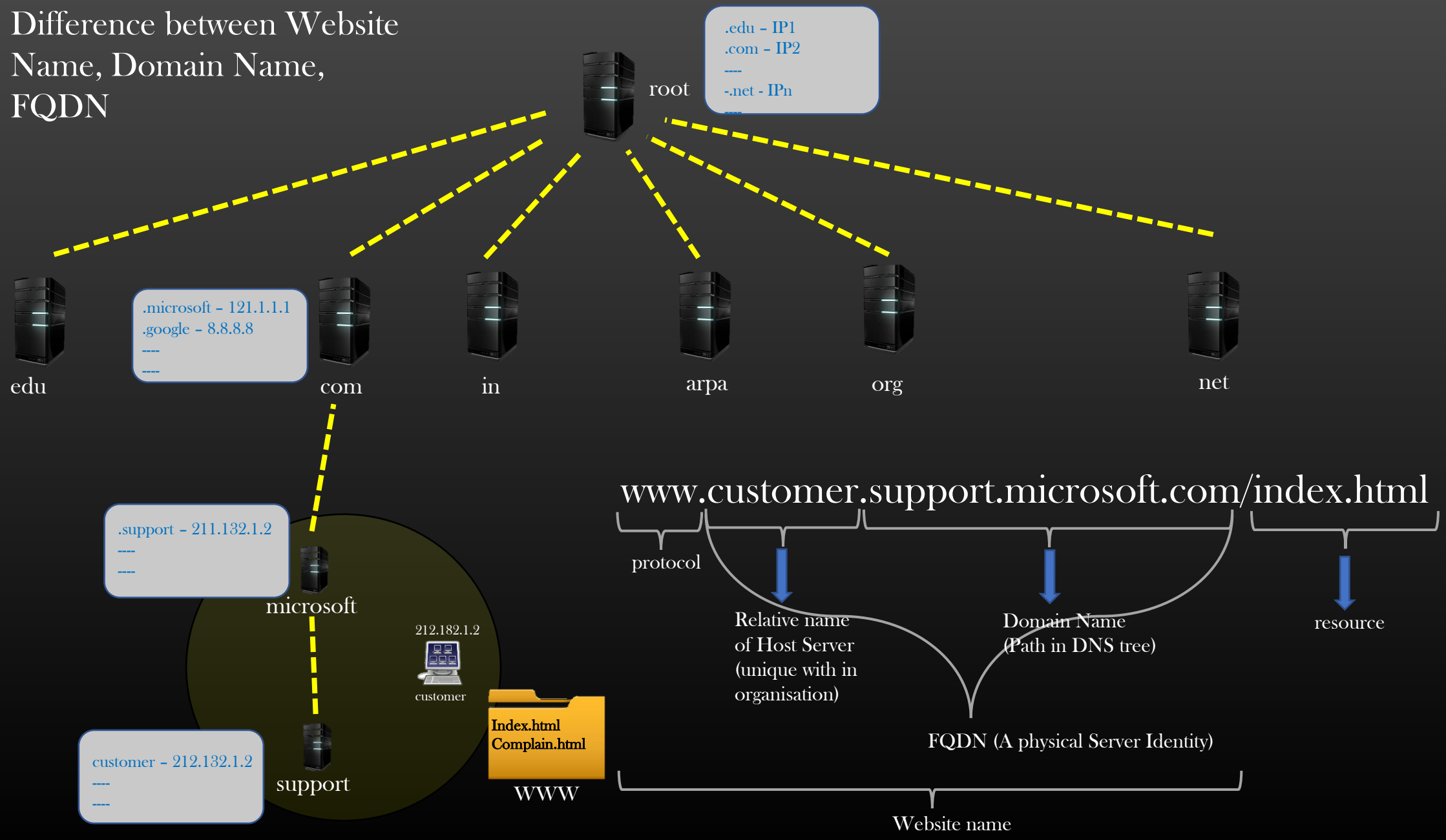
FQDN - Fully Qualified Domain Name



Thus, next time when you try to access any website, think of it like you are trying to access a physical server deployed Somewhere in the network

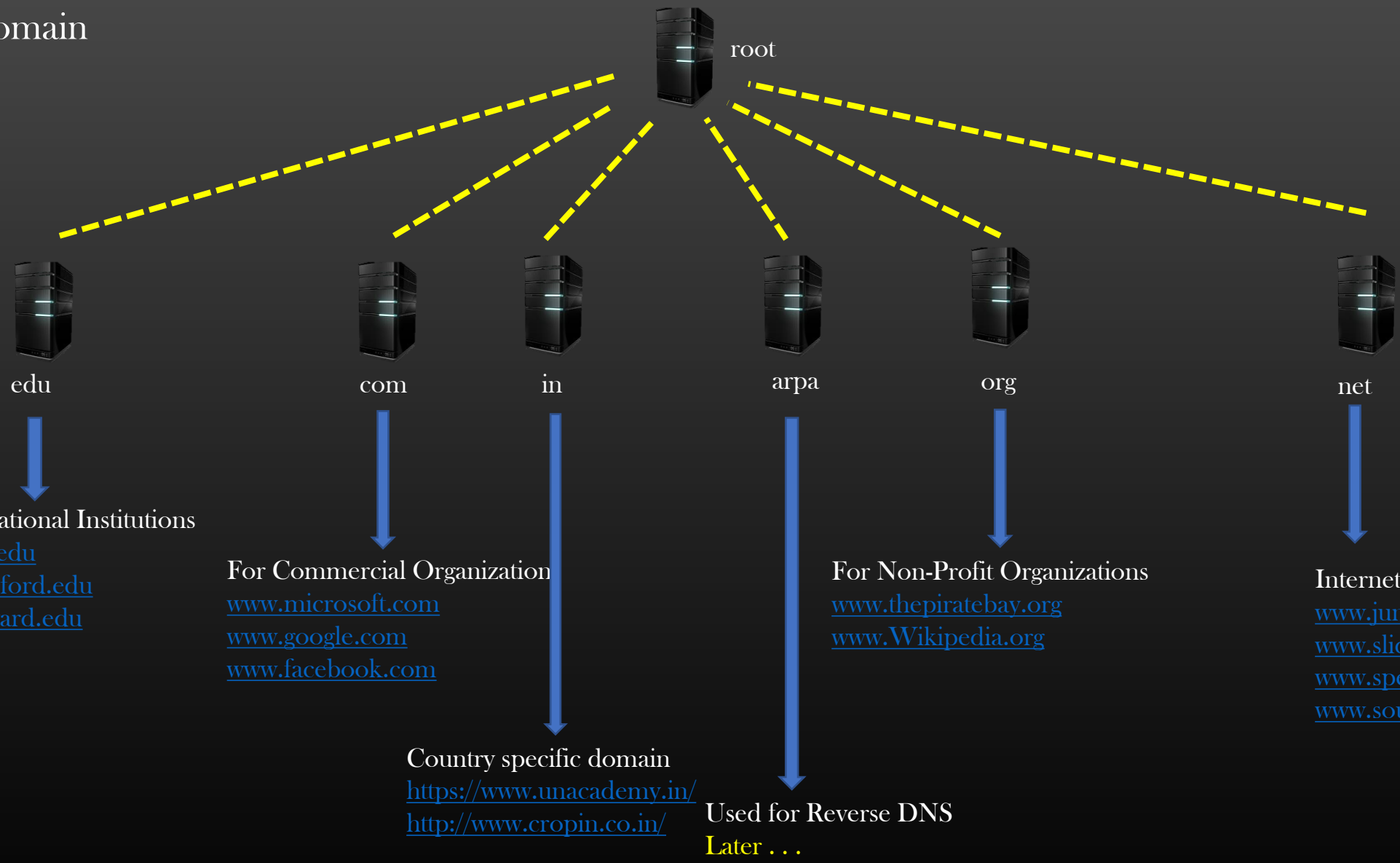
SKIPPED : Domain Name Server (DNS)

Difference between Website Name, Domain Name, FQDN



Domain Name Server (DNS)

Top Level Domain classification

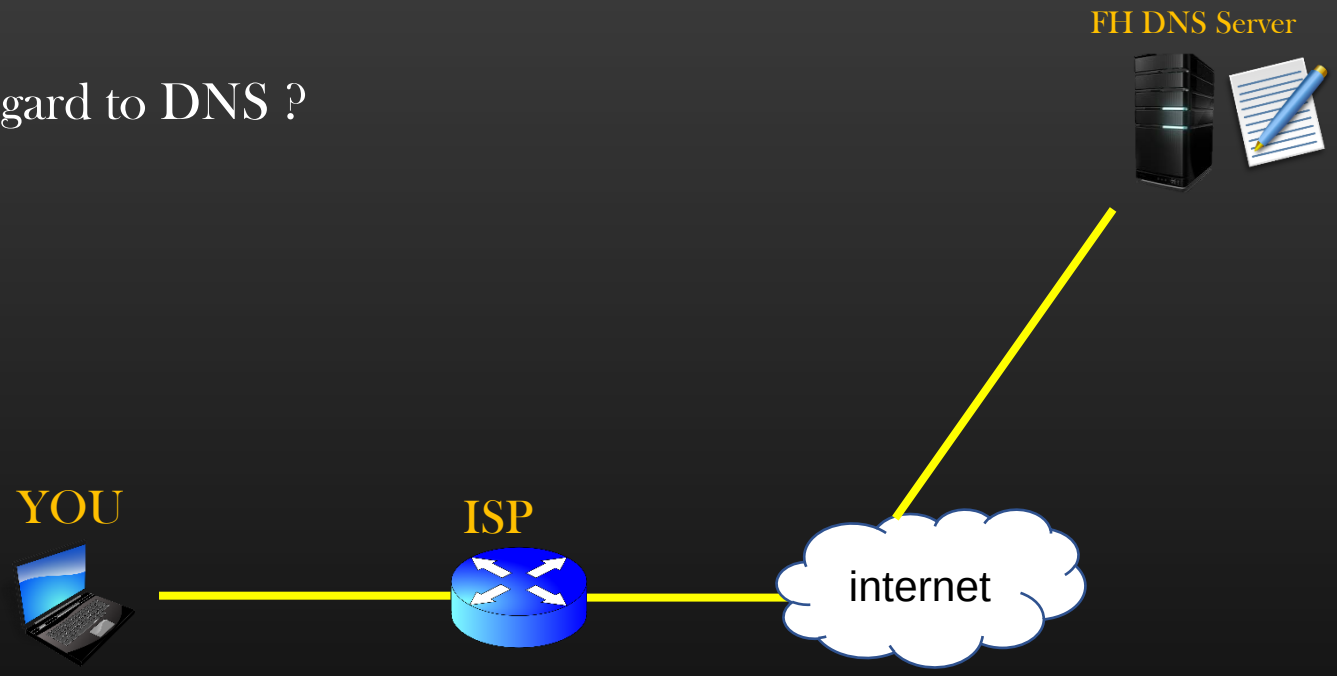


First Hop DNS Servers

What are FH DNS Server Machines ?

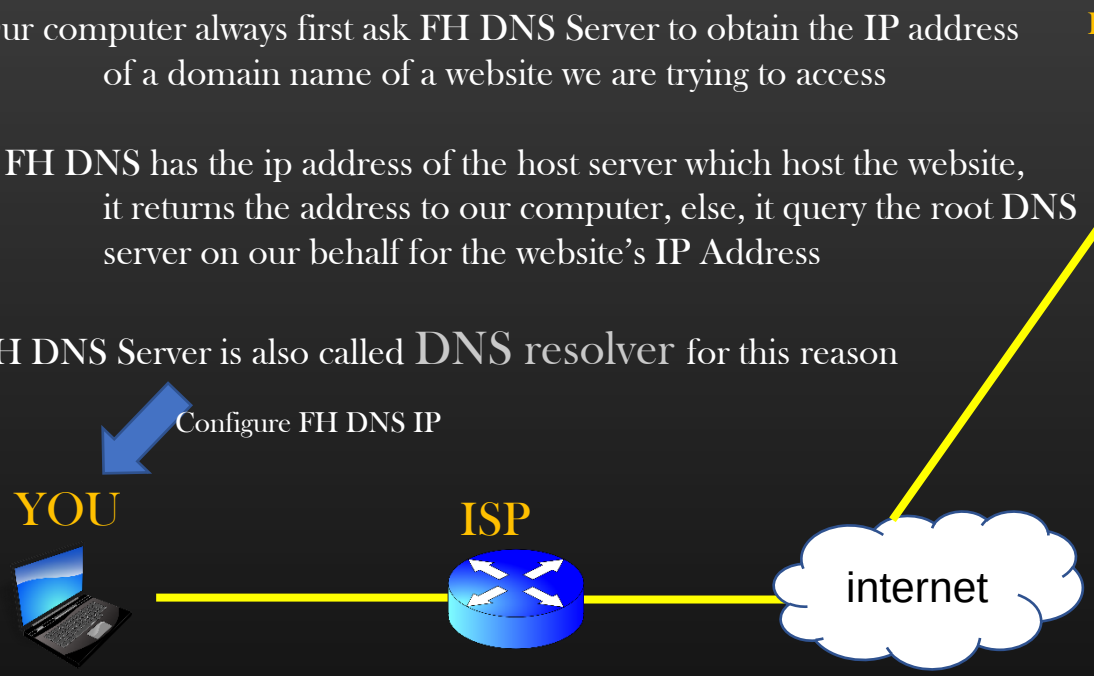
What is their purpose/role in regard to DNS ?

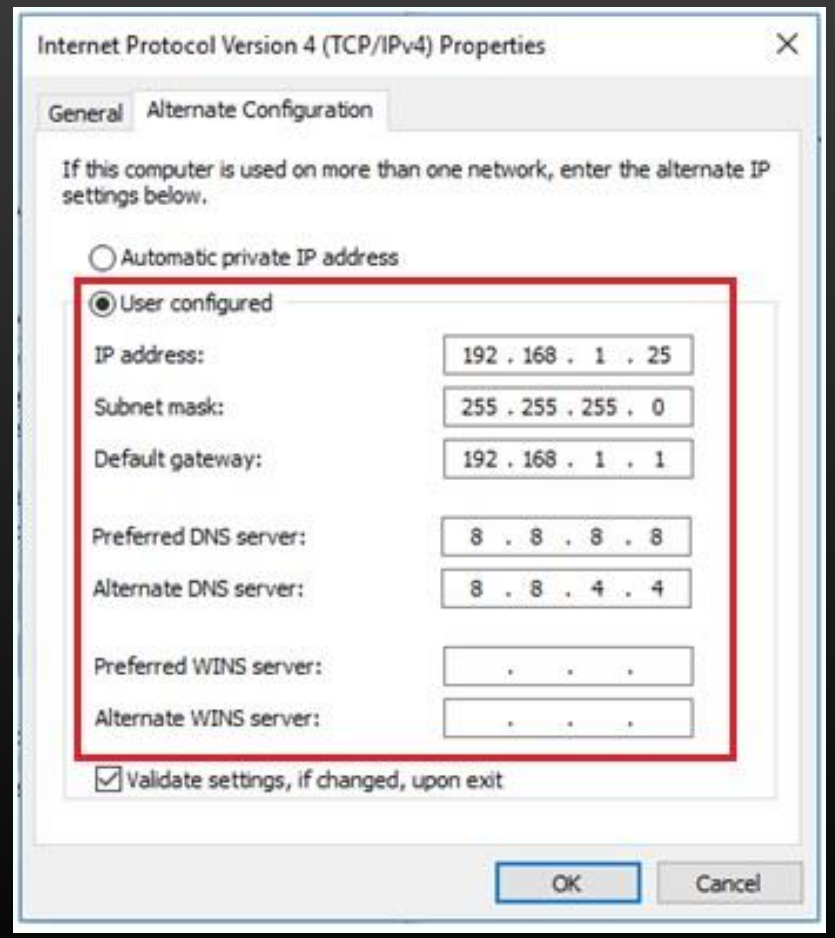
What problem do they solve ?



First Hop DNS Servers

- Our computer is assigned the address of First Hop DNS Server – Either statically/Manually configured Or Dynamically obtained from DHCP Servers
- Our computer always first ask FH DNS Server to obtain the IP address of a domain name of a website we are trying to access
- If FH DNS has the ip address of the host server which host the website, it returns the address to our computer, else, it query the root DNS server on our behalf for the website's IP Address
- FH DNS Server is also called DNS resolver for this reason



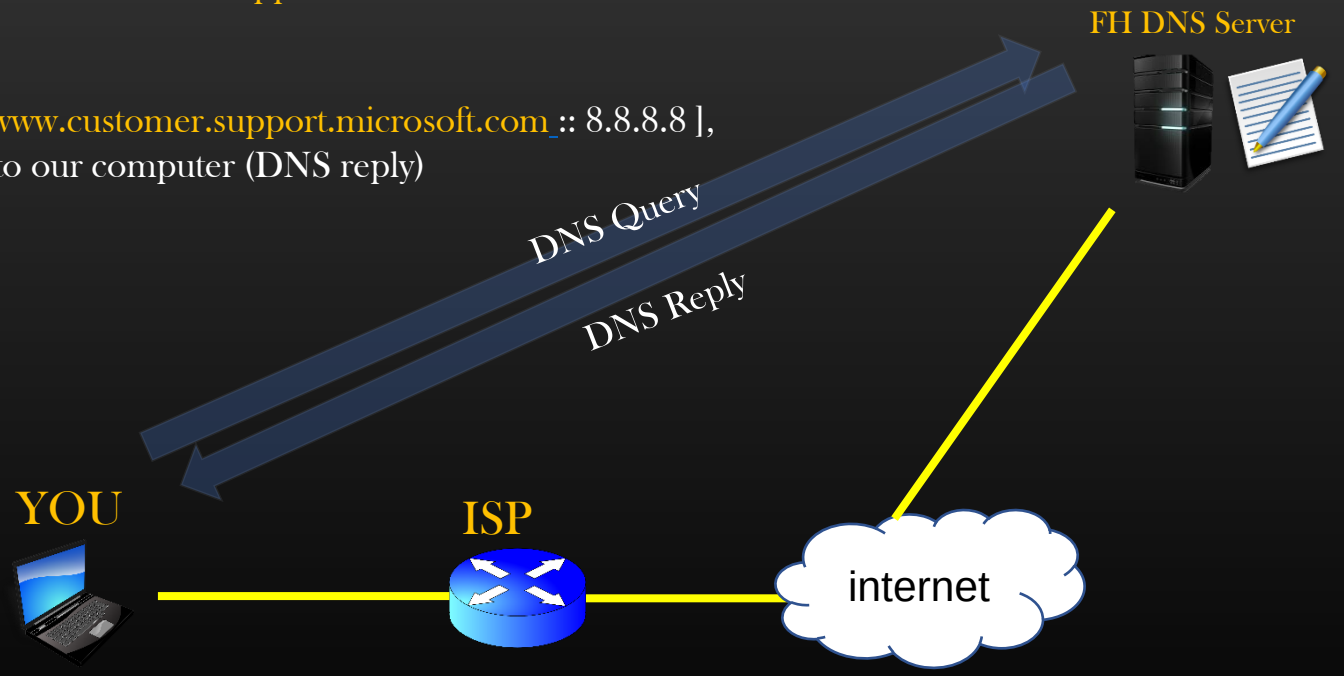


Or : *ipconfig /all* in cmd prompt

- FH DNS Server caches the recently accessed website's ip address in its local DNS Cache so that, in future, our computer can obtain the IP address of recently accessed website quickly from FH DNS only , FH DNS server is shared by several users
- Our computer also have its own local DNS Cache

First Hop DNS Servers

- Lets Say : YOU try to access the website : www.customer.support.microsoft.com
- Our computer look up in its local DNS Cache.
  Assume it don't have IP address of www.google.com
- Our computer send DNS Query to FH DNS Server asking :
  What is IP address of www.customer.support.microsoft.com ?
- If FH DNS Server has entry : [www.customer.support.microsoft.com :: 8.8.8.8],
  then it returns 8.8.8.8 to our computer (DNS reply)

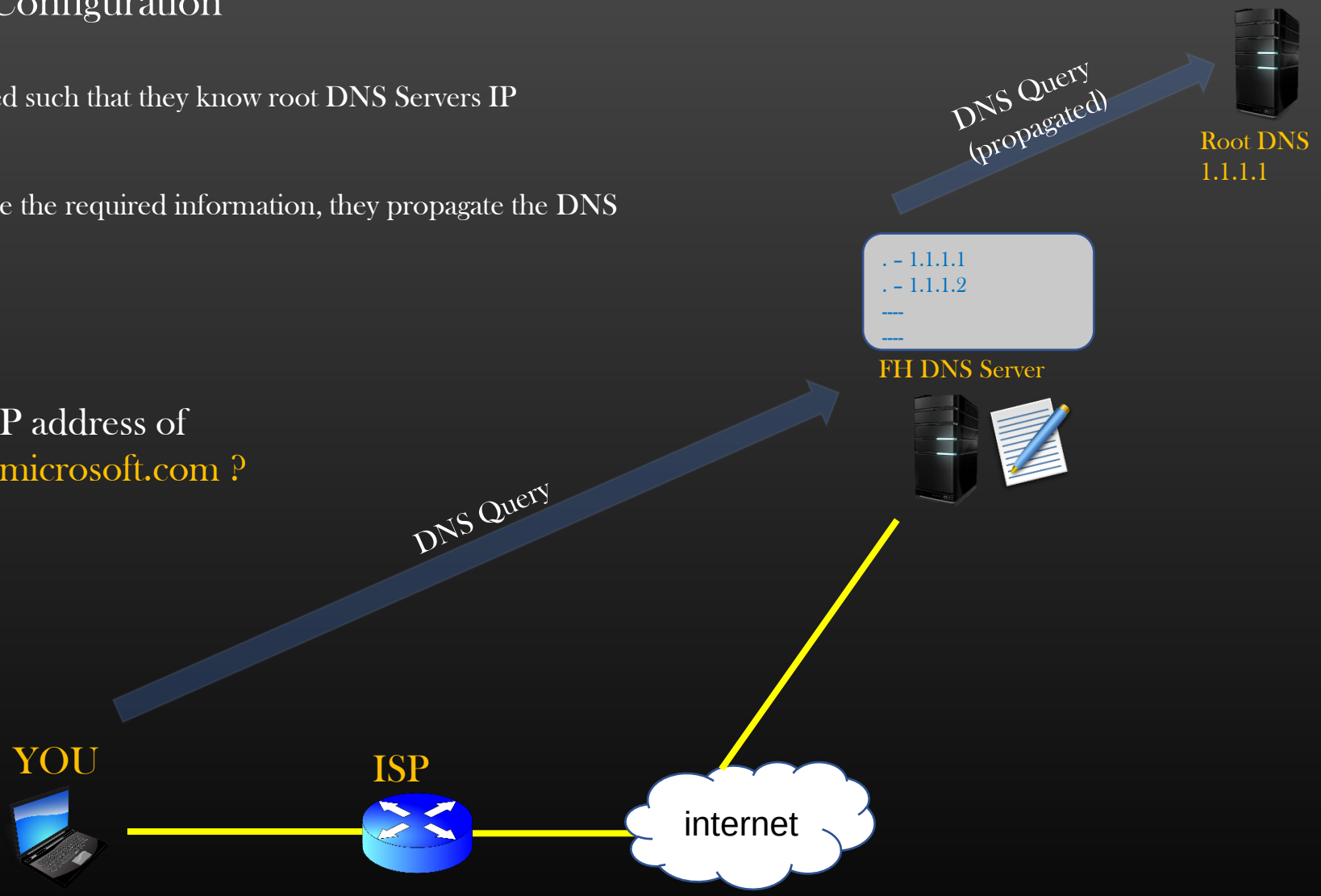


- But What if FH DNS Server also do not have IP address of www.customer.support.microsoft.com ?

First Hop DNS Servers Configuration

- FH DNS Servers are configured such that they know root DNS Servers IP Addresses
- If FH DNS Servers do not have the required information, they propagate the DNS Query to Root DNS Servers

DNS Query : What is IP address of www.customer.support.microsoft.com ?



Next : Now, Lets try to understand the complete DNS Mechanism to resolve the website domain name to IP address

The DNS Mechanism

The Goal of DNS mechanism is to find the Answer to DNS Query, that is to find the ip address of the host server which host the requested website

There are three Ways different Mechanism to resolve DNS Query :

1. Recursive Method, DNS Query is called Recursive DNS Query
2. Iterative Method, DNS Query is called Iterative DNS Query
3. Reverse look up DNS Query

Let Us discuss each one by one !

DNS Query Delegation

Your friend : you, When is the networking class scheduled ?

You : I don't know, but I think John knows , go and Ask him → Query Delegation

Your friend : Ok

Referral answer

<Your friend goes to John and ask>

Your friend : When is the networking class scheduled ?

John : I don't know, but I think William knows it, go and Ask him → Query Delegation

Your friend : Ok

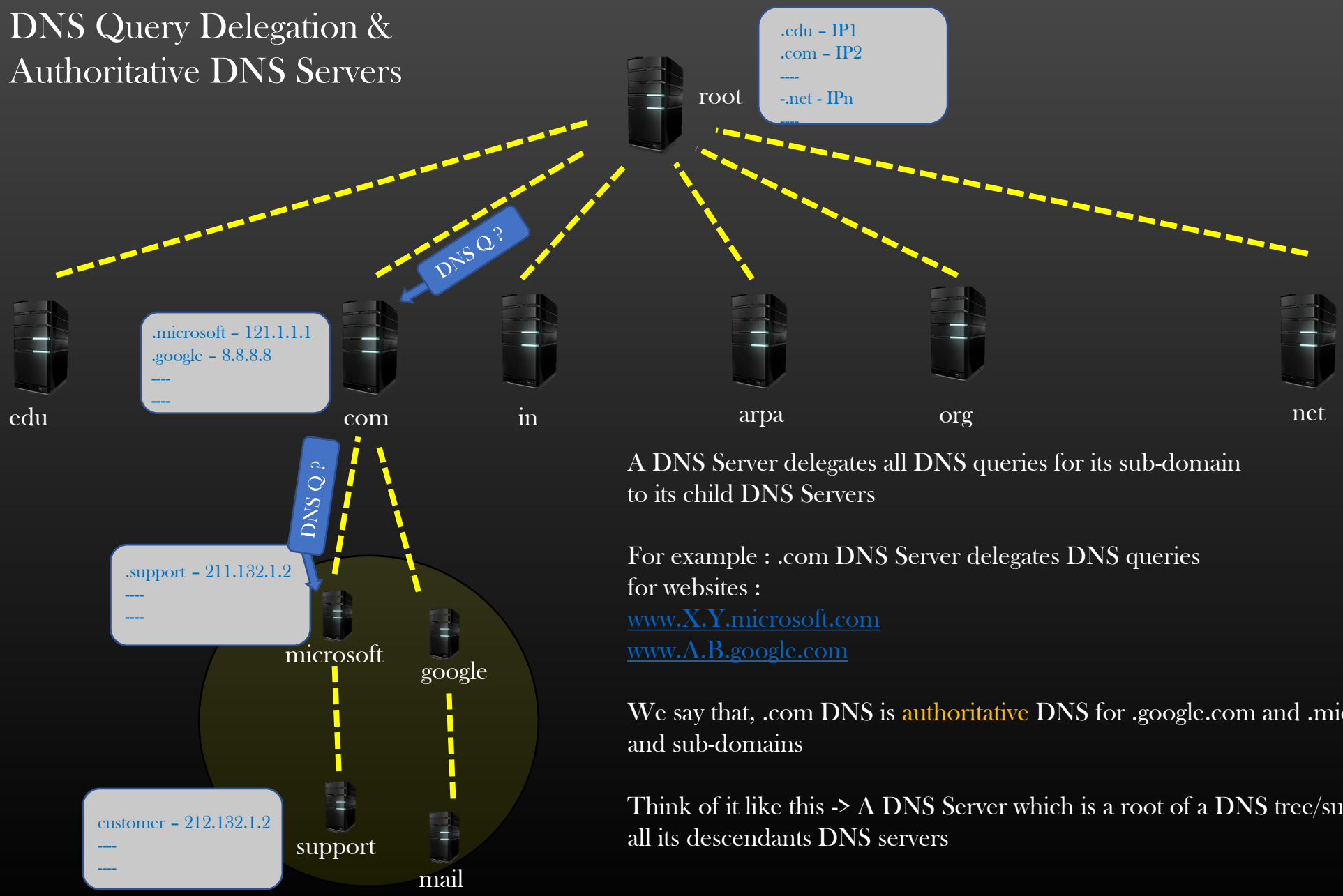
<Your friend goes to William and ask>

Your friend : When is the networking class scheduled ?

William : 9 a.m. → Definite Answer

Your friend : Thanks

DNS Query Delegation & Authoritative DNS Servers



A DNS Server delegates all DNS queries for its sub-domain to its child DNS Servers

For example : .com DNS Server delegates DNS queries for websites :
www.X.Y.microsoft.com
www.A.B.google.com

We say that, .com DNS is authoritative DNS for .google.com and .microsoft.com domains and sub-domains

Think of it like this -> A DNS Server which is a root of a DNS tree/sub-tree is authoritative for all its descendants DNS servers

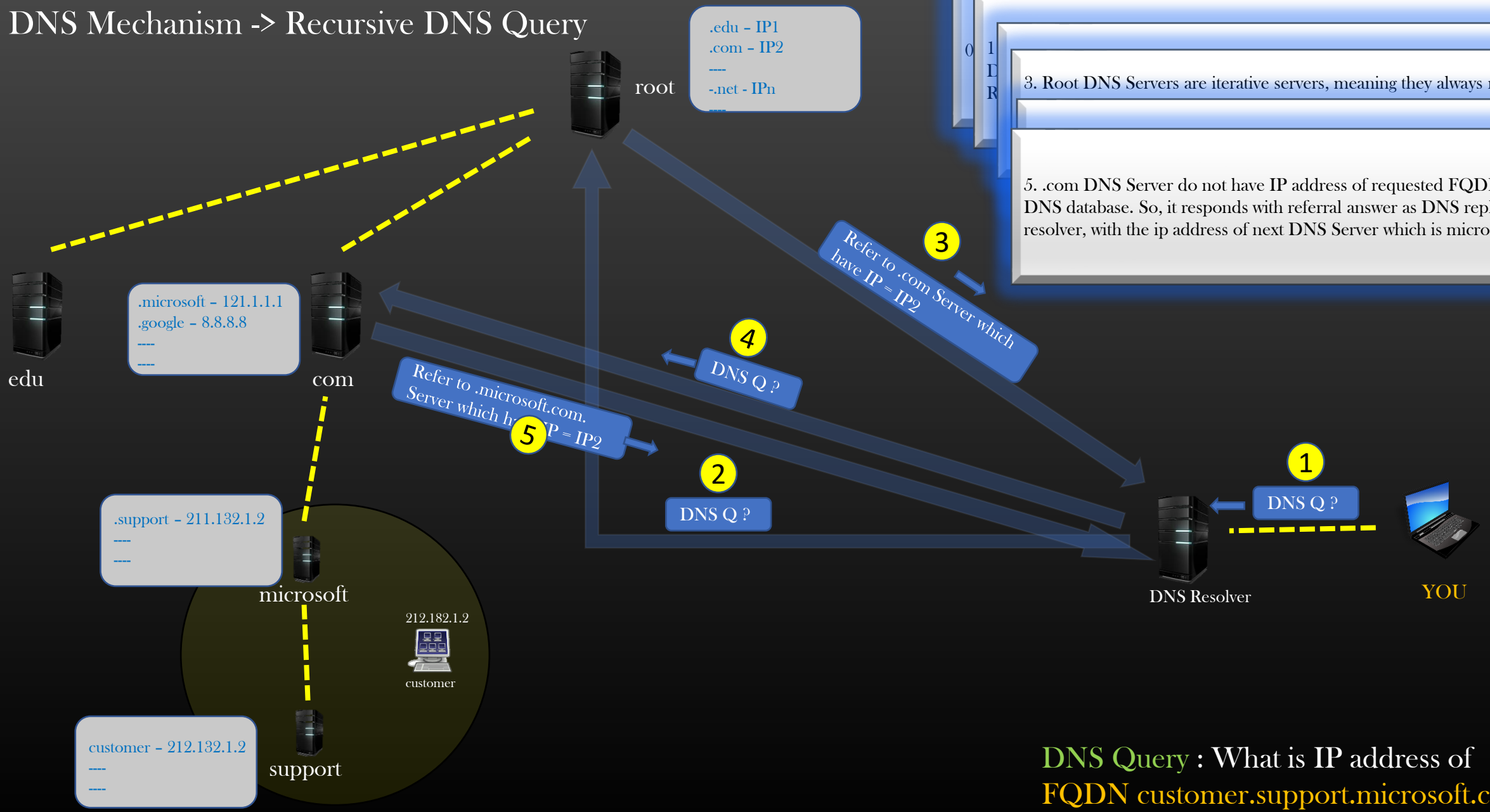
DNS Mechanism

FQDN → IP Address

customer.support.microsoft.com → 212.132.1.2

Domain Name Server (DNS)

DNS Mechanism -> Recursive DNS Query



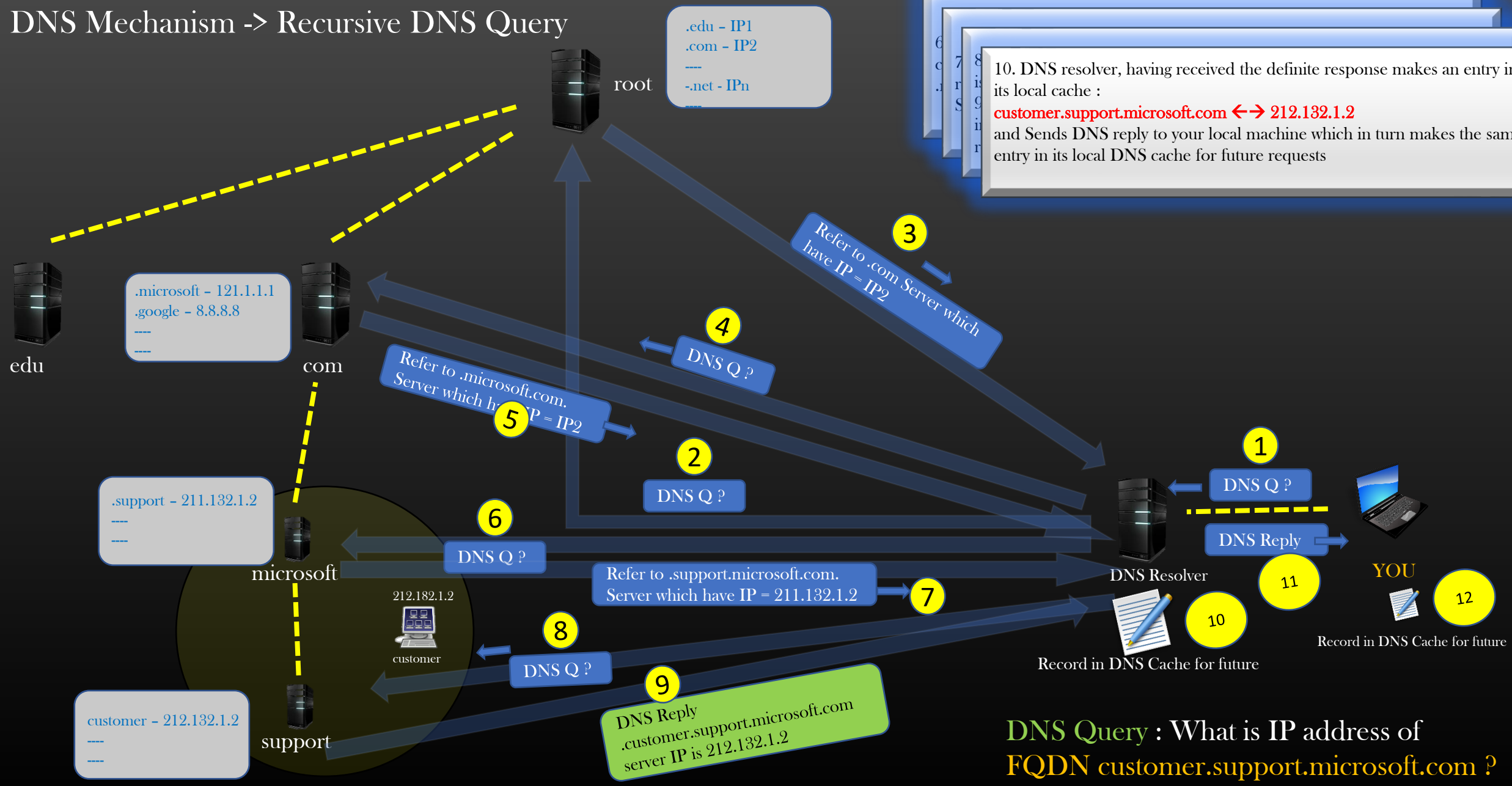
3. Root DNS Servers are iterative servers, meaning they always reply back

5. .com DNS Server do not have IP address of requested FQDN in its DNS database. So, it responds with referral answer as DNS reply to DNS resolver, with the ip address of next DNS Server which is microsoft.com.

DNS Query : What is IP address of FQDN customer.support.microsoft.com ?

Domain Name Server (DNS)

DNS Mechanism -> Recursive DNS Query



10. DNS resolver, having received the definite response makes an entry in its local cache :
customer.support.microsoft.com ←→ 212.132.1.2
and Sends DNS reply to your local machine which in turn makes the same entry in its local DNS cache for future requests

DNS Query : What is IP address of FQDN customer.support.microsoft.com ?

DNS Mechanism -> Recursive DNS Query

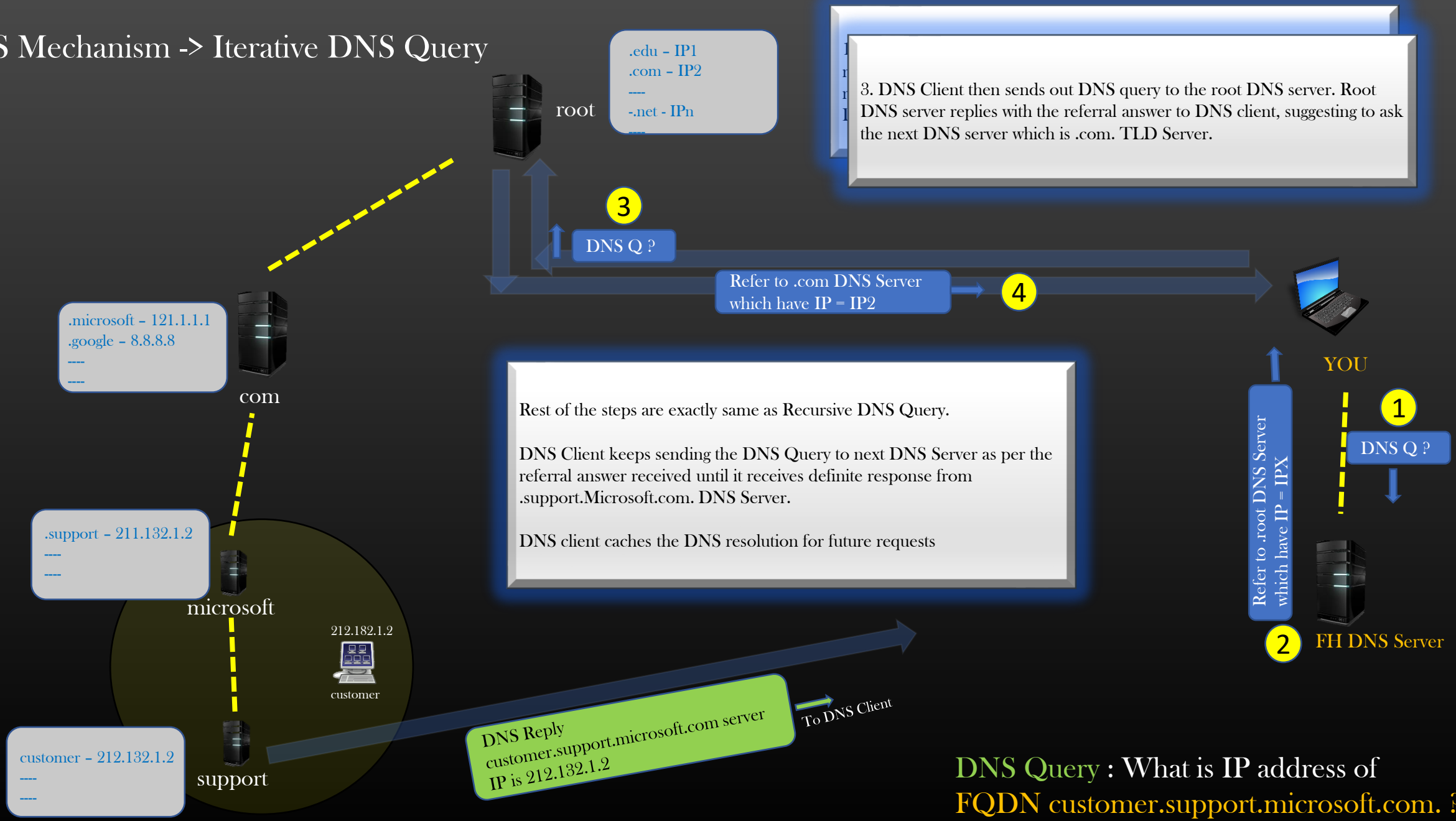
- It is the responsibility of DNS resolver DNS Server to yield a definite response to the requested DNS query
- DNS resolver keeps on sending the DNS query to subsequent DNS Servers in an attempt to obtain definite response to the DNS query
- DNS resolver replies to the host machines (DNS client) either with a definite response or an error msg
- Root and TLD DNS Servers replies with the referral answer to DNS resolver
- DNS Resolver makes an entry in its local DNS cache when it receives the definite DNS resolution for a DNS query to entertain future DNS query requests
- You host machine also caches the DNS resolution to entertain future DNS query requests

DNS Mechanism -> Iterative DNS Query

- In this DNS Mechanism, the host machine generates the Iterative DNS query
- DNS query packet format has a flag which states whether it is recursive DNS query Or Iterative DNS Query
- In this mode, DNS Resolver do not take responsibility to recursively ask the other DNS Servers in a hope for a definite response
- If DNS resolver itself do not know the DNS resolution for requested FQDN, it too, returns with the referral response to the DNS client host machine
- The DNS client machine, having received the referral response either from DNS resolver or other DNS Server, sends DNS query to next DNS Server specified in the last referral response
- DNS client machine, it self, takes up the role which DNS resolver was doing in case of recursive DNS query
- Rest of the mechanism is same as that of recursive DNS query

Domain Name Server (DNS)

DNS Mechanism -> Iterative DNS Query



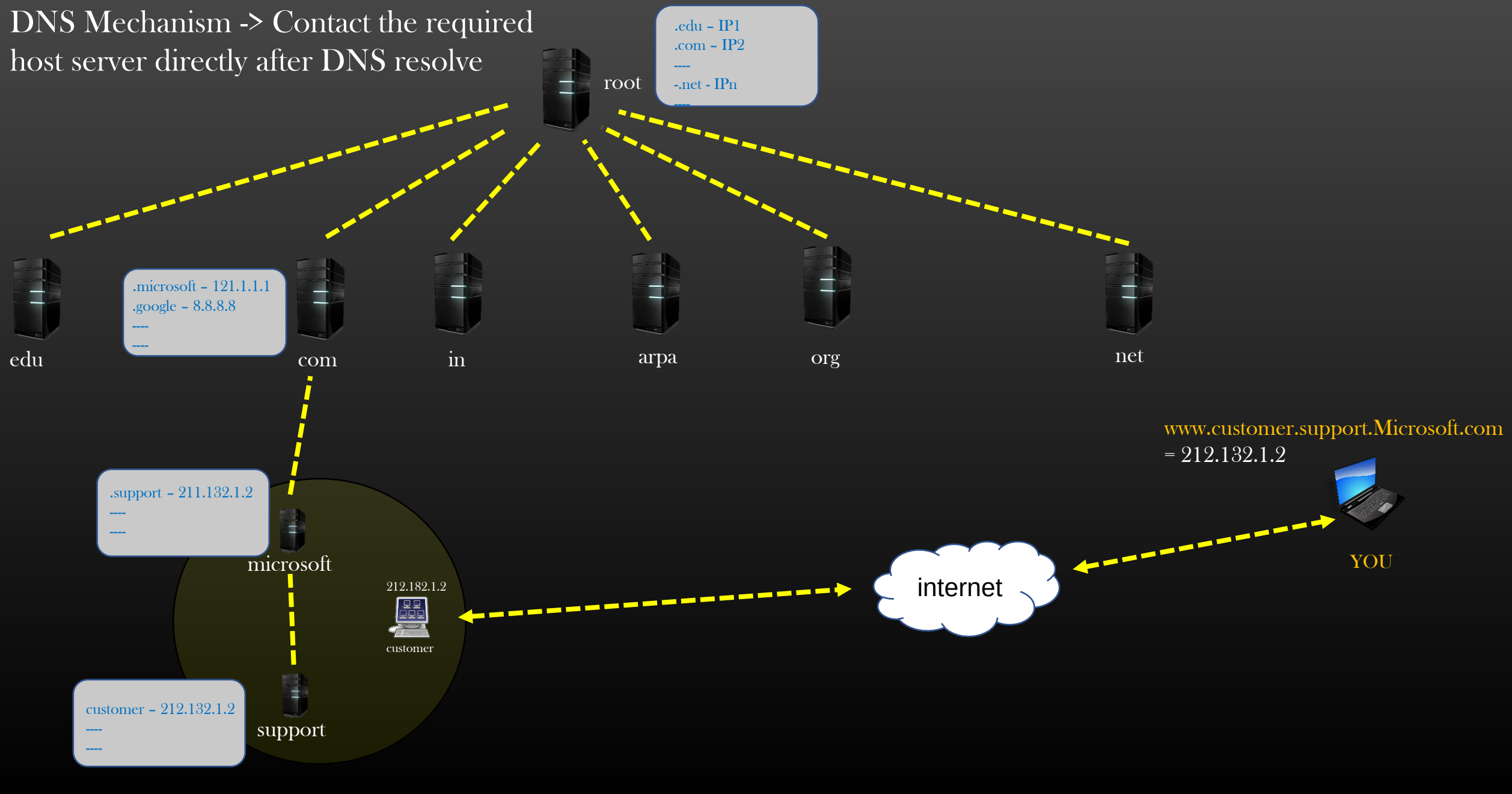
DNS Query : What is IP address of FQDN customer.support.microsoft.com. ?

DNS Mechanism -> Iterative DNS Query Vs Recursive DNS Query

Recursive DNS Query	Iterative DNS Query
DNS Resolver takes the responsibility to get the final DNS resolution	DNS client has to work itself to get the DNS resolution
DNS resolver keep sending DNS Query to subsequent DNS server based on referral answer	DNS client keep sending DNS Query to subsequent DNS server based on referral answer
DNS client receives a definite answer : Either the DNS resolution Or error msg	DNS client receives a referral answer or error msg
Burden is on DNS resolver	Burden is on DNS client

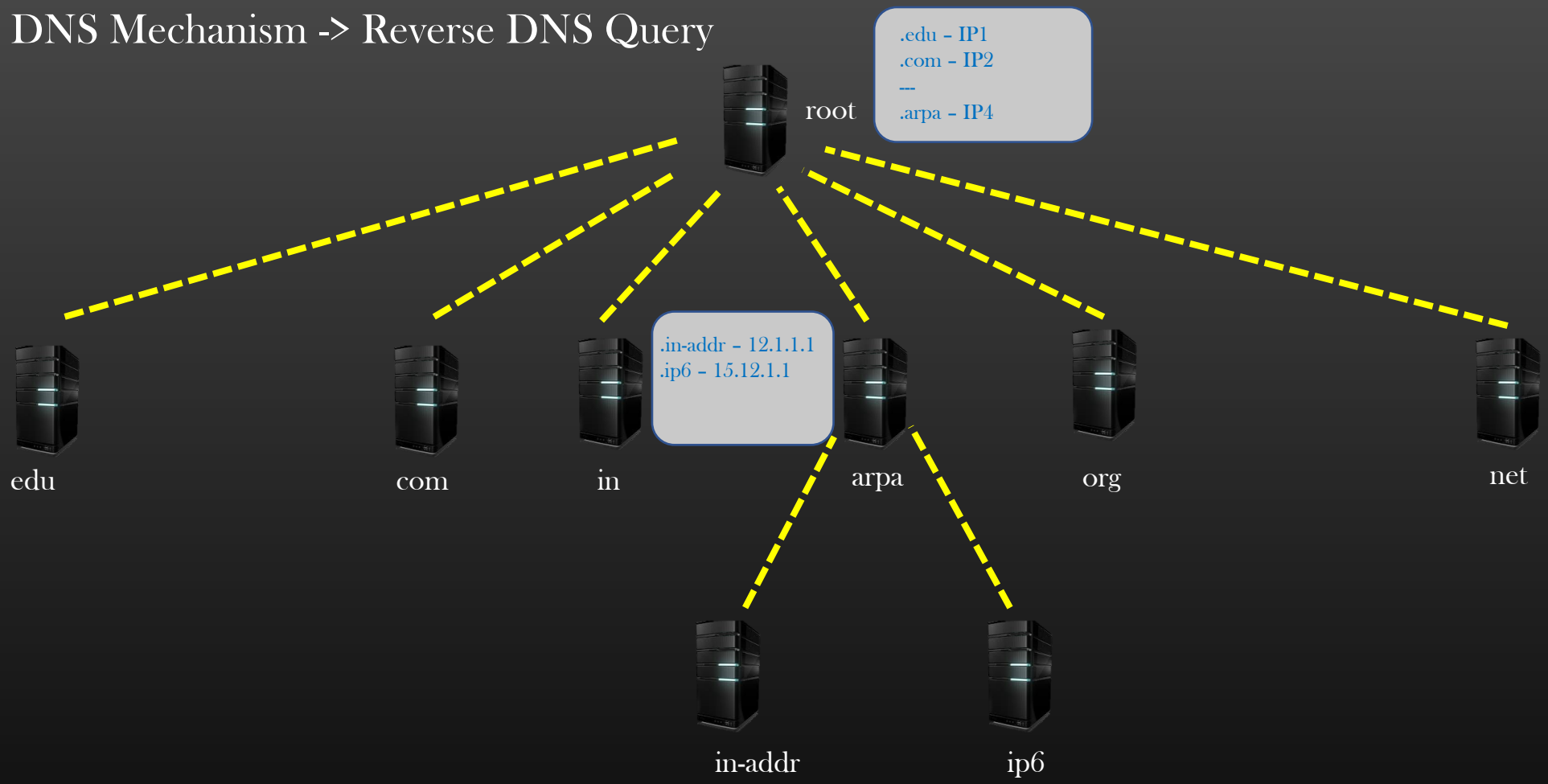
Domain Name Server (DNS)

DNS Mechanism -> Contact the required host server directly after DNS resolve



Domain Name Server (DNS)

DNS Mechanism -> Reverse DNS Query



GOAL :

Given : 212.132.1.2

To Determine : customer.support.microsoft.com

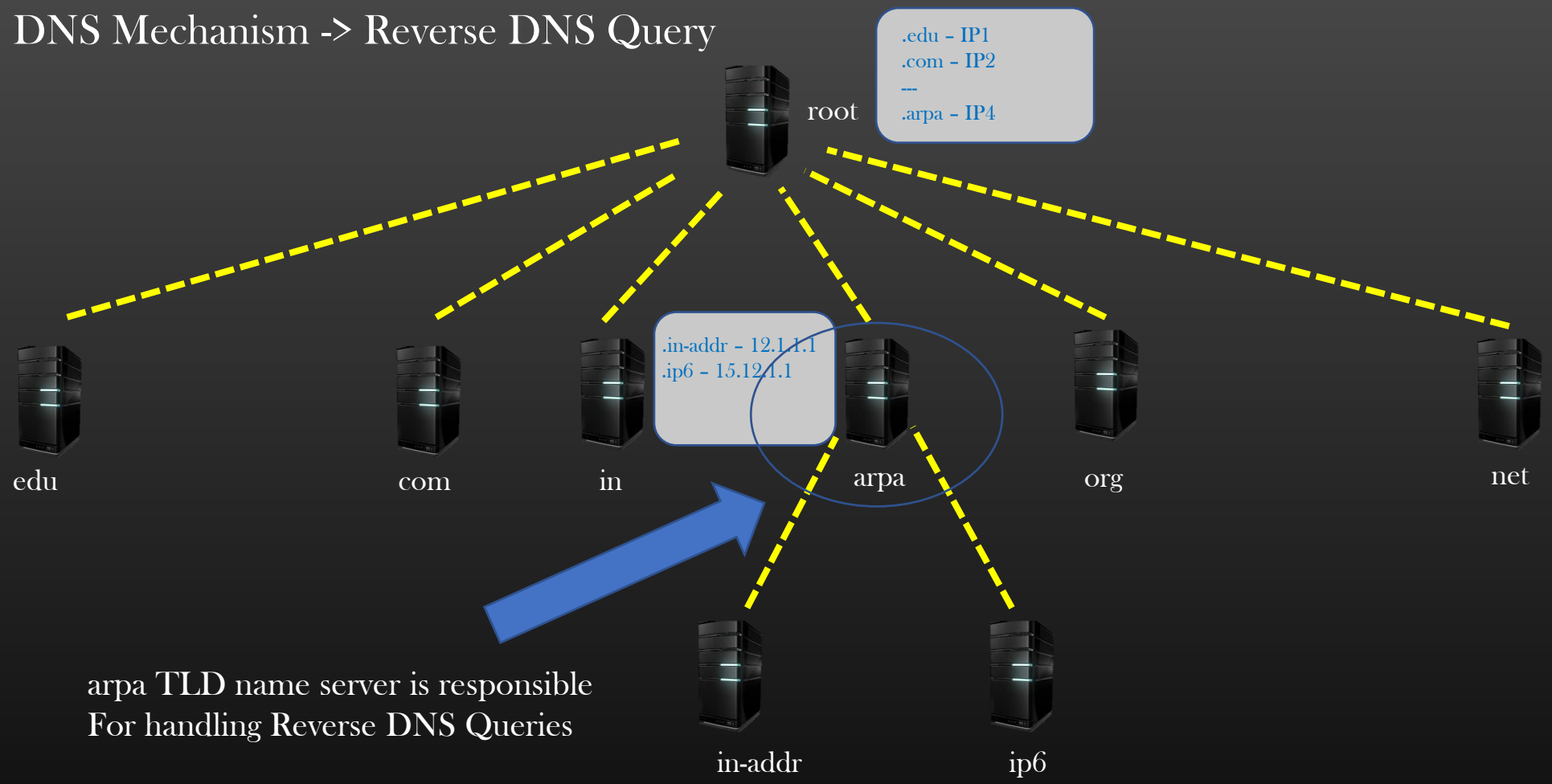
Reverse DNS Mechanism

IP Address → FQDN

212.132.1.2 → customer.support.microsoft.com

Domain Name Server (DNS)

DNS Mechanism -> Reverse DNS Query



arpa TLD name server is responsible
For handling Reverse DNS Queries

YOU

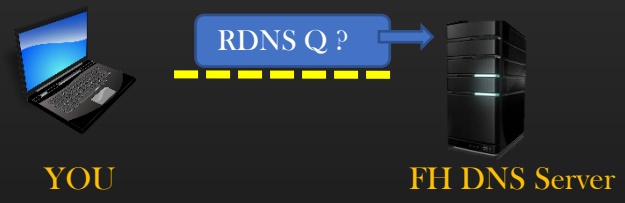
GOAL :

Given : 212.132.1.2

To Determine : customer.support.microsoft.com

DNS Mechanism -> Reverse DNS Query

- The procedure is exactly same as Iterative/Recursive Query processing by DNS System



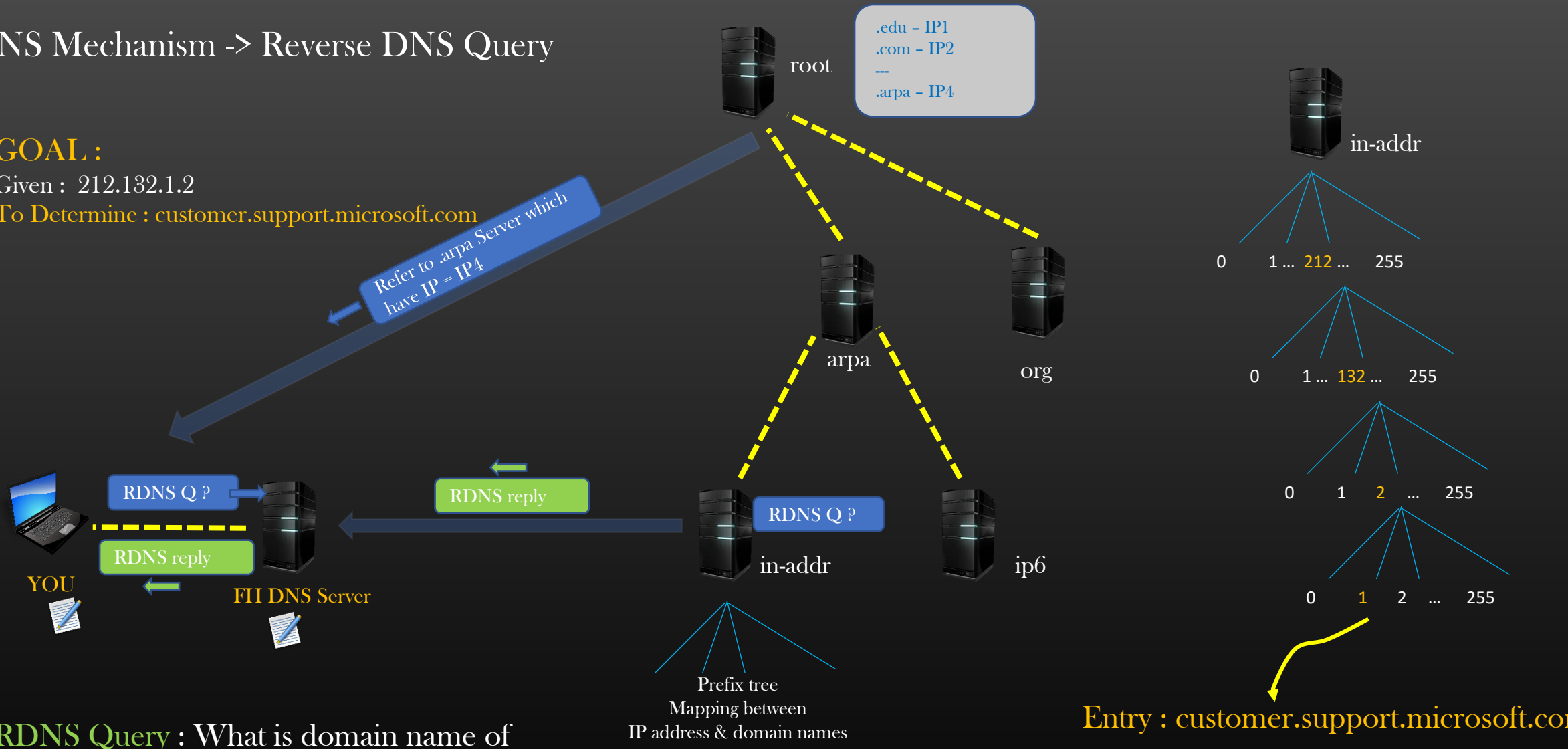
- When you issue a RDNS Query for IP 212.132.1.2, Your system do the following before issuing a RDNS Query :
  1. Reverse the IP address : 2.1.132.212
  2. append labels : in-addr.arpa.

Final RDNS Query formed :
What is domain name of
1.2.132.212.in-addr.arpa. ?

RDNS Query -> What is Domain name for IP 212.132.1.2 ?
Expected Answer to be returned by DNS System : customer.support.microsoft.com

Domain Name Server (DNS)

DNS Mechanism -> Reverse DNS Query



GOAL :

Given : 212.132.1.2

To Determine : customer.support.microsoft.com

RDNS Query : What is domain name of 1.2.132.212.in-addr.arpa. ?

RDNS Reply : customer.support.microsoft.com domain name of 1.2.132.212.in-addr.arpa.

☺

DNS Mechanism -> Reverse DNS Query

- Reverse DNS lookup is opposite of normal DNS lookup
- Reverse DNS lookup query can be both – Iterative Or recursive
- *arpa* is TLD Server responsible for handling all RDNS queries
- *in-addr* for ipv4 addresses is the 2nd level DNS responsible to provide definite response to RDNS query
- *ip6* DNS is ipv6 counter part of in-addr DNS
- Reverse DNS is used in troubleshooting / traceroute commands etc

Domain Name Server (DNS)

Summary

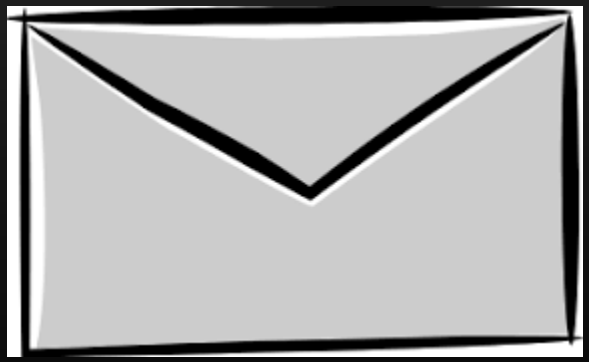
1. DNS System is hierarchical and decentralized, root and top level DNS are under central authority called IANA
2. Second level and below DNS Servers are under the authority of respective organization
3. FHDNS is also called DNS Resolver
4. DNS Resolver and Your own system caches the recent DNS resolutions to speed up future requests
5. Recursive DNS Query is returned with the definite response
6. Iterative DNS Query is returned by the referral response
7. DNS Servers are replicated at multiple geographical locations
8. *arpa* TLD DNS Server is responsible for IP to domain name resolution
9. DNS System is like an internet phone book

Domain Name System DNS

Thank you

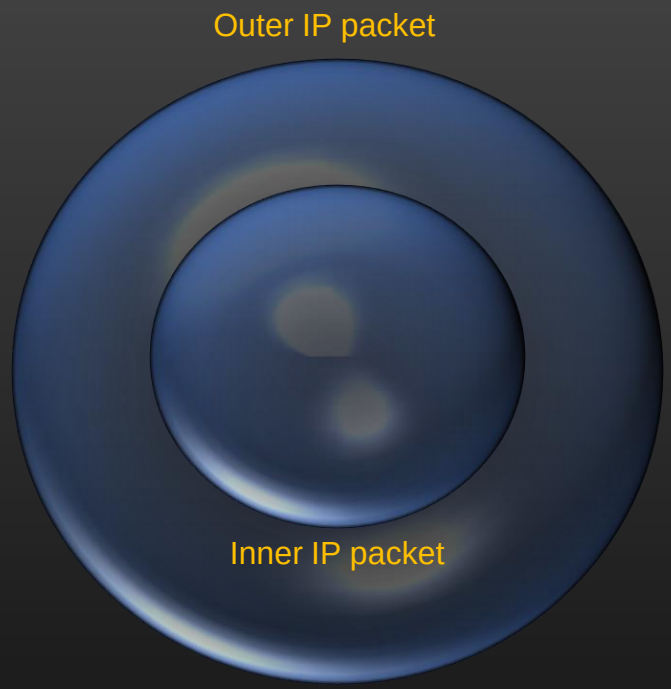
IP-in-IP Encapsulation

(Hiding inside)



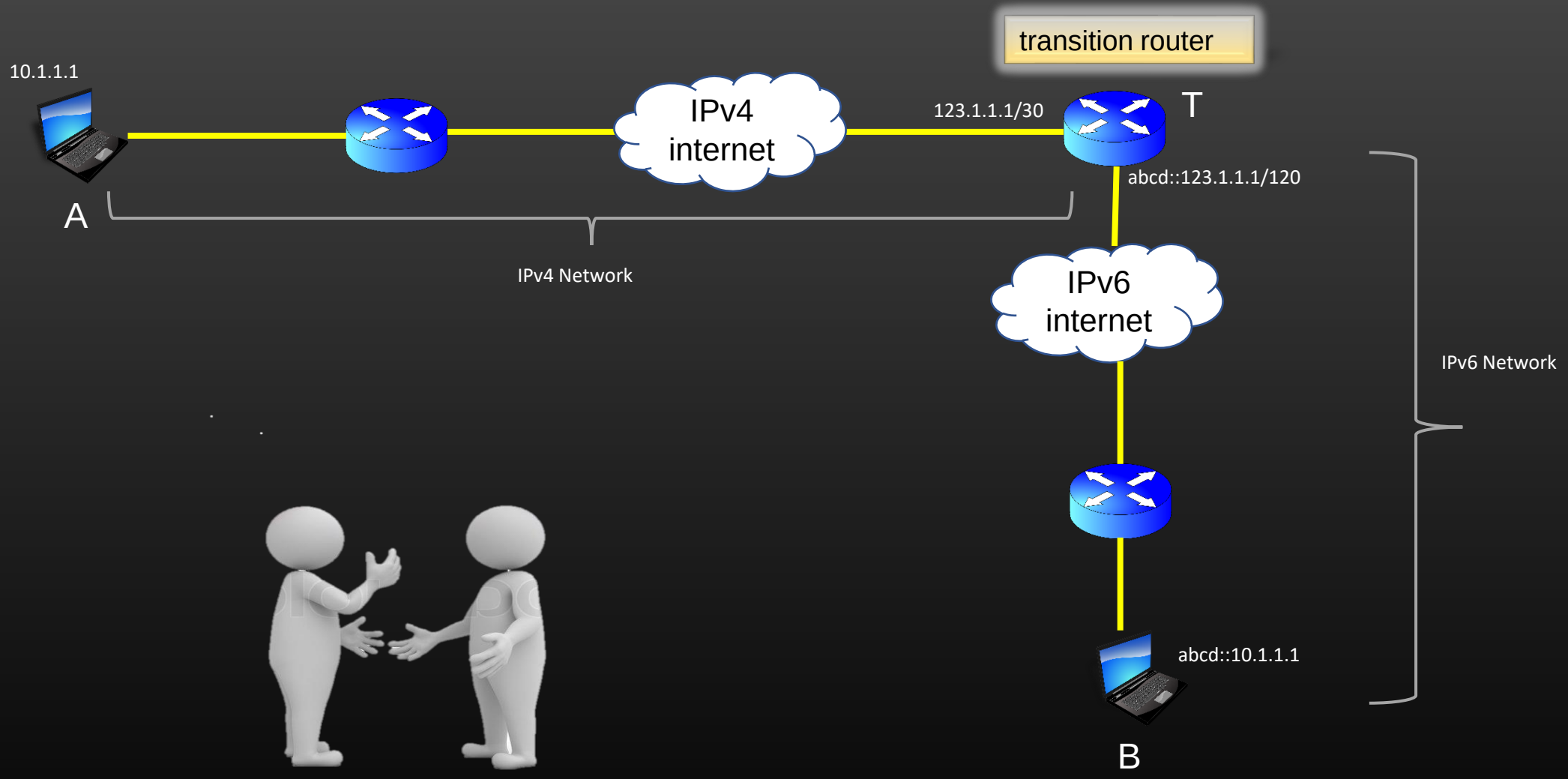
IP in IP Encapsulation

- In General, Encapsulation means hiding something inside
- IP-in-IP Encapsulation means – IP packet inside an IP packet
- IP-in-IP Encapsulation protocol = 4
- Encapsulation In networking is also called Tunneling
- Other variants
  - IPv6-in-IP Encapsulation (protocol = 41)
  - IP-in-IPv6 Encapsulation (protocol = 4)
  - IPv6-in-IPv6 Encapsulation (protocol = 41)
- Why do we need Encapsulation ? What problem does it solve ?
- How does the encapsulated packet structure look like ?
- How Encapsulated packet is treated by the L3 router ?
- Let us demystify encapsulation/Tunneling in networking ..

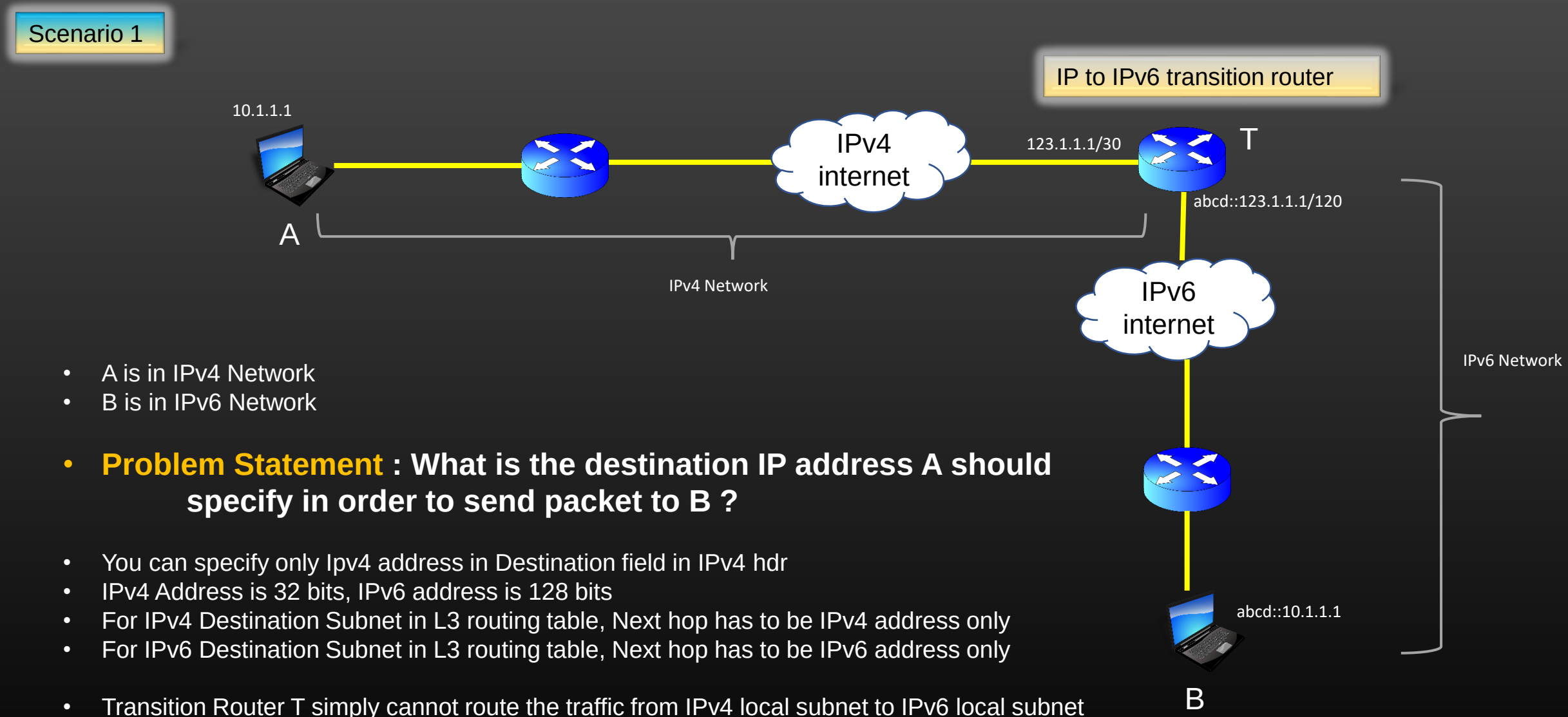


IP-in-IP Encapsulation

Heterogeneous Network



IP in IP Encapsulation -> Why do we need Encapsulation ?



- A is in IPv4 Network
- B is in IPv6 Network

- **Problem Statement : What is the destination IP address A should specify in order to send packet to B ?**

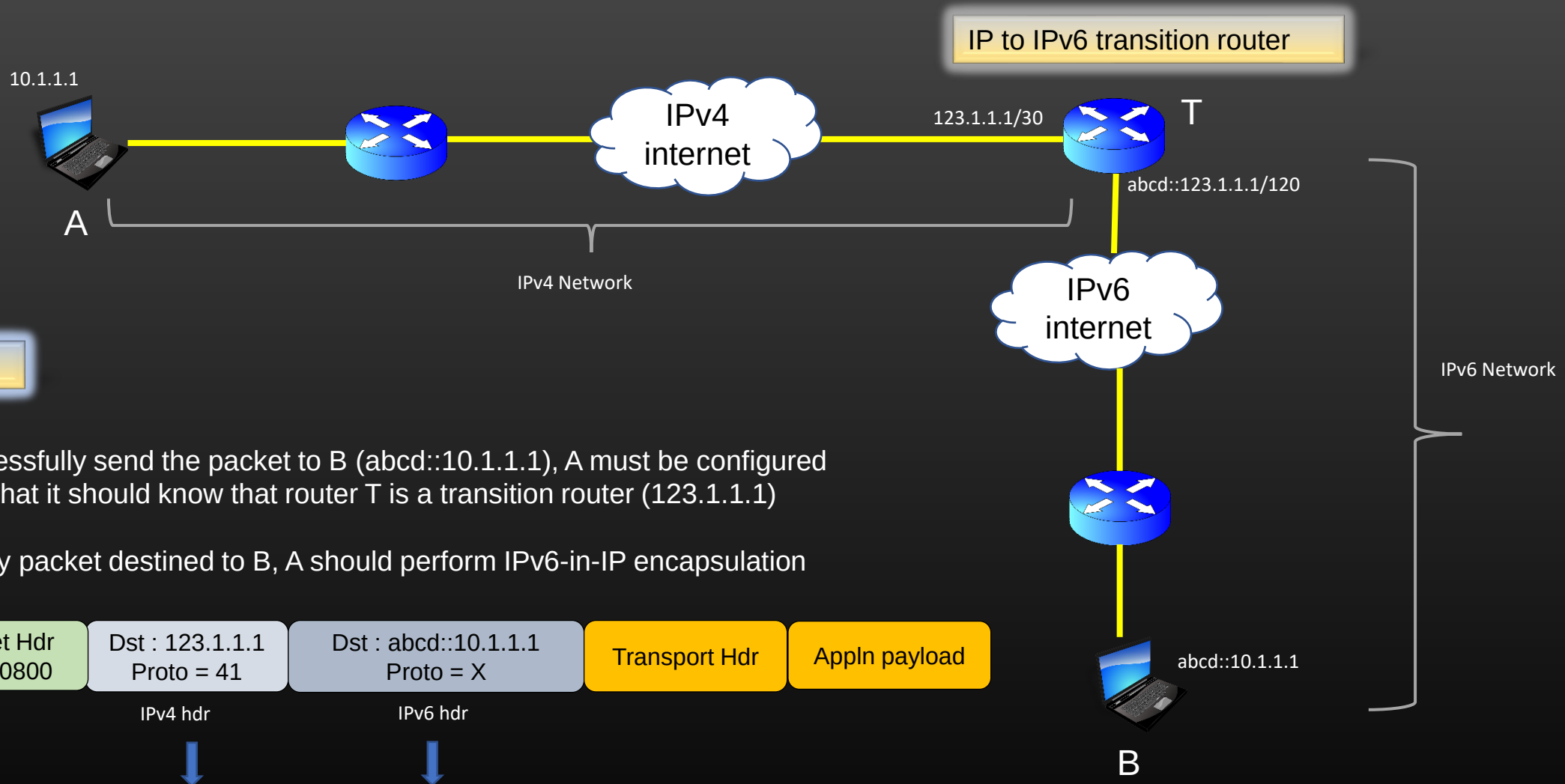
- You can specify only Ipv4 address in Destination field in IPv4 hdr
- IPv4 Address is 32 bits, IPv6 address is 128 bits
- For IPv4 Destination Subnet in L3 routing table, Next hop has to be IPv4 address only
- For IPv6 Destination Subnet in L3 routing table, Next hop has to be IPv6 address only

- Transition Router T simply cannot route the traffic from IPv4 local subnet to IPv6 local subnet using traditional L3 routing concepts

- There is a need that A and B should still be able to talk to each other
- Network Layer Encapsulation solves this problem

IP in IP Encapsulation -> IPv6-in-IP Encapsulation

Scenario 1



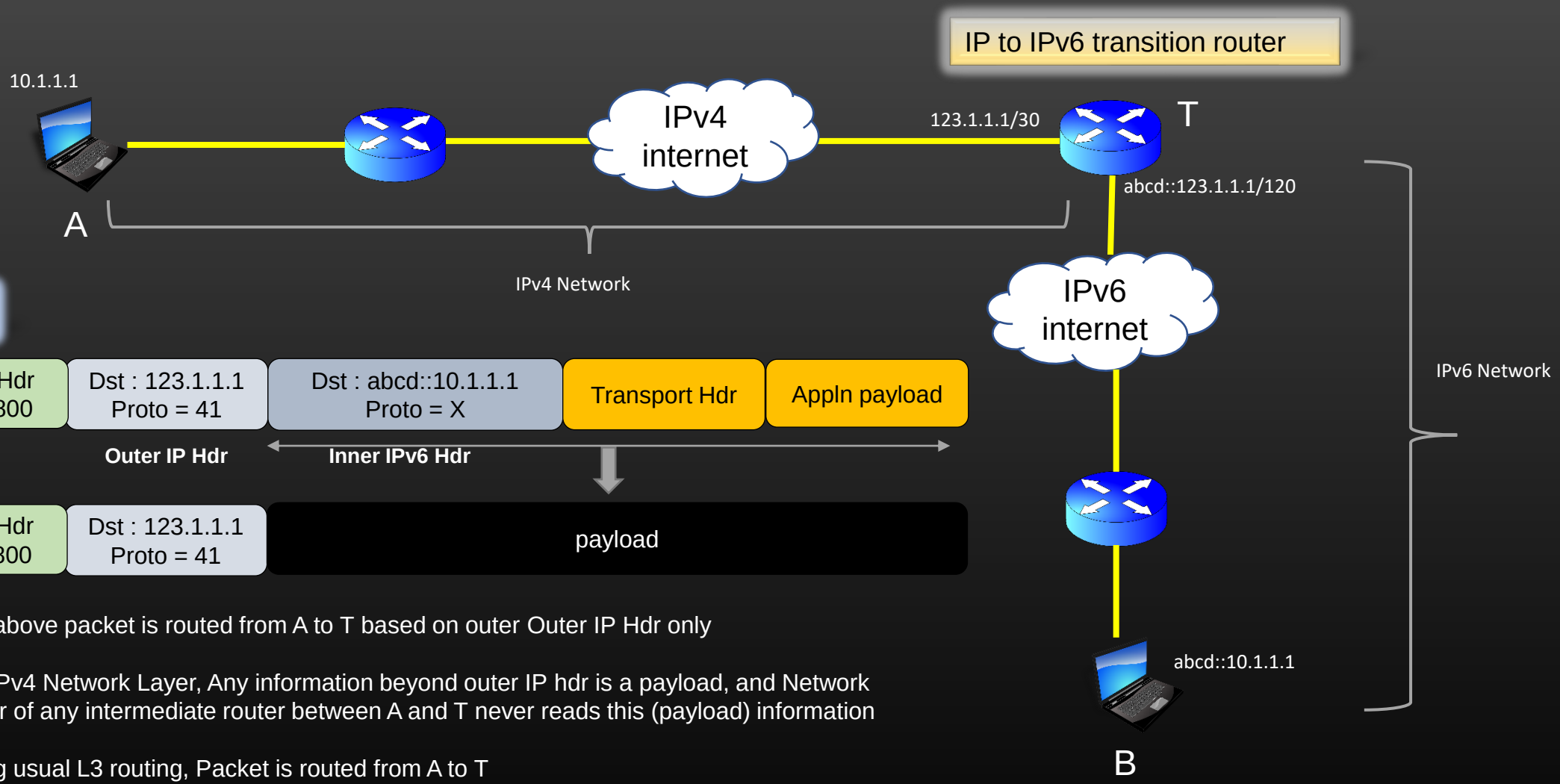
Encapsulation

- To Successfully send the packet to B (abcd::10.1.1.1), A must be configured that it should know that router T is a transition router (123.1.1.1)
- For every packet destined to B, A should perform IPv6-in-IP encapsulation

Ethernet Hdr Type = 0800	Dst : 123.1.1.1 Proto = 41	Dst : abcd::10.1.1.1 Proto = X	Transport Hdr	Appln payload
	IPv4 hdr	IPv6 hdr		
	Outer Hdr	**Inner Hdr**		

IP in IP Encapsulation -> IPv6-in-IP Encapsulation

Scenario 1



Encapsulation

Ethernet Hdr Type = 0800	Dst : 123.1.1.1 Proto = 41	Dst : abcd::10.1.1.1 Proto = X	Transport Hdr	Appln payload

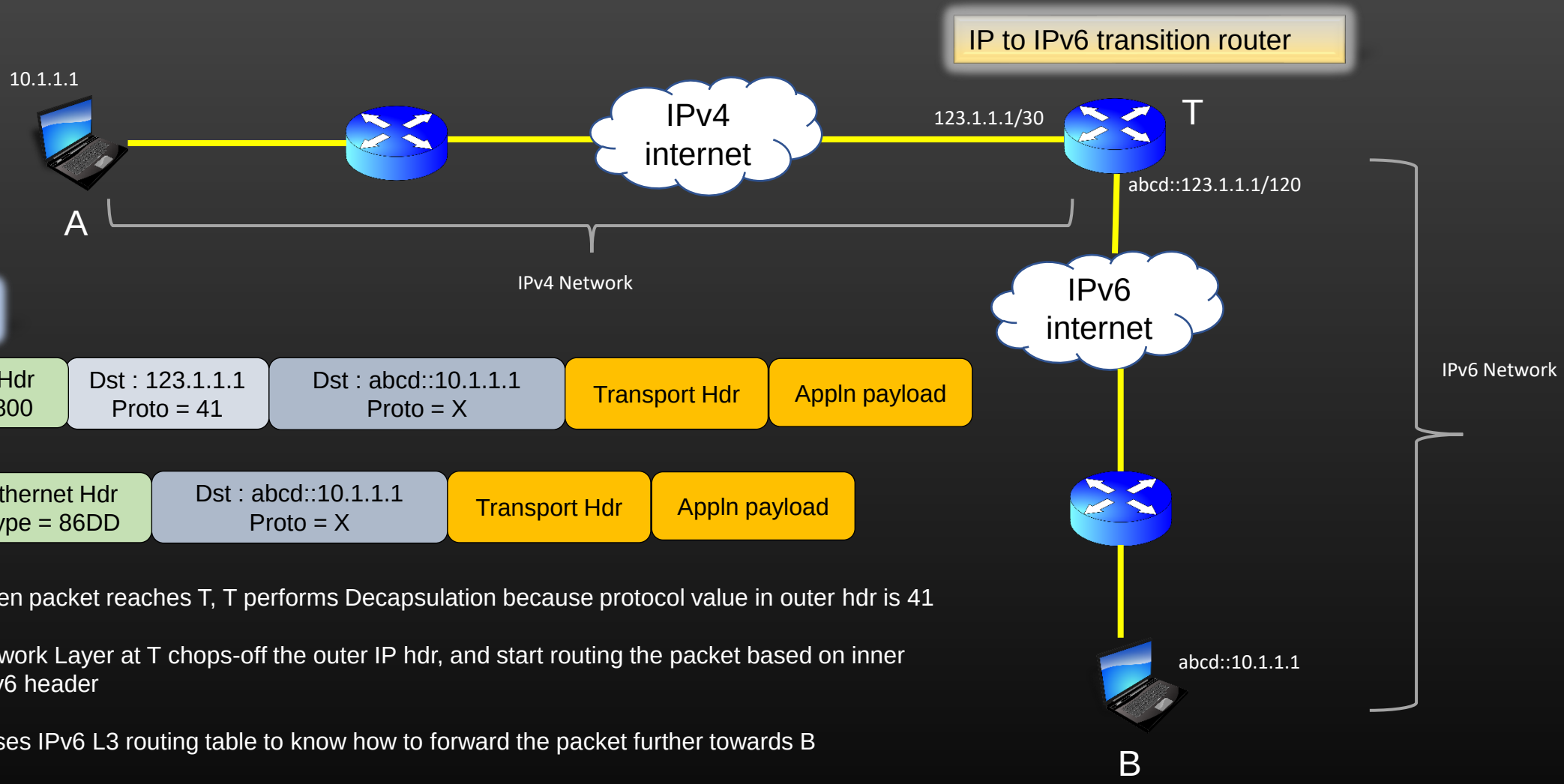
Outer IP Hdr — Inner IPv6 Hdr

Ethernet Hdr Type = 0800	Dst : 123.1.1.1 Proto = 41	payload

- Step 1 : The above packet is routed from A to T based on outer Outer IP Hdr only
- Step 2 : For IPv4 Network Layer, Any information beyond outer IP hdr is a payload, and Network layer of any intermediate router between A and T never reads this (payload) information
- Step 3 : Using usual L3 routing, Packet is routed from A to T
- Step 4 : T sees, Dst ip address in IPv4 hdr = IP of its local subnet, so, next it inspect the protocol field in the ipv4 hdr
- Step 5 : Seeing the protocol value = 41 in IPv6 hdr, T now knows that following hdr is IPv6 hdr, T then performs Decapsulation

IP in IP Encapsulation -> IPv6-in-IP Encapsulation

Scenario 1



Decapsulation

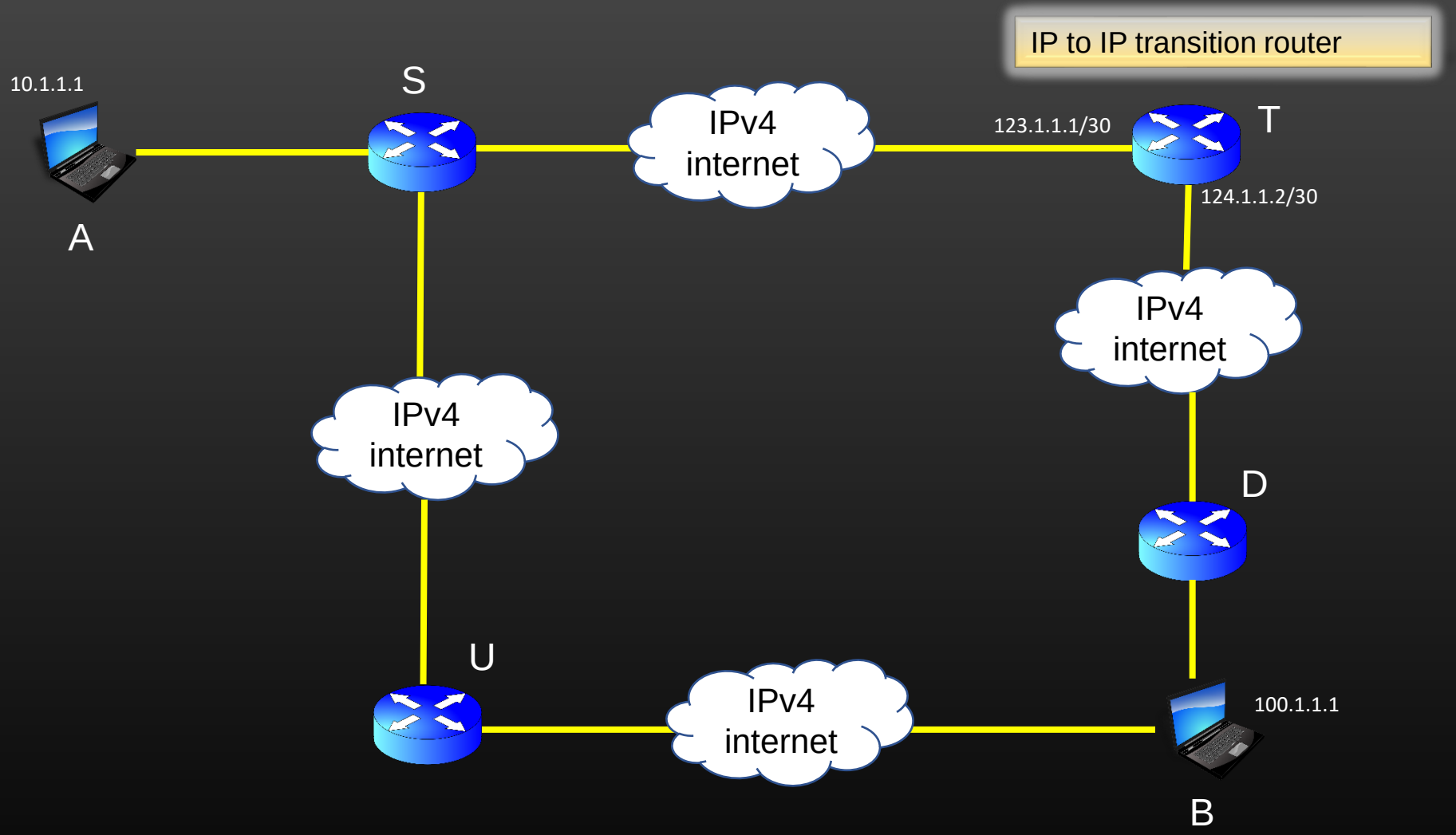
Ethernet Hdr Type = 0800	Dst : 123.1.1.1 Proto = 41	Dst : abcd::10.1.1.1 Proto = X	Transport Hdr	Appln payload

Ethernet Hdr Type = 86DD	Dst : abcd::10.1.1.1 Proto = X	Transport Hdr	Appln payload

- Step 6 : When packet reaches T, T performs Decapsulation because protocol value in outer hdr is 41
- Step 7 : Network Layer at T chops-off the outer IP hdr, and start routing the packet based on inner IPv6 header
- Step 8 : T uses IPv6 L3 routing table to know how to forward the packet further towards B
- Step 9 : Using normal IPv6 L3 routing, packet eventually reaches to B
- Step 10 : When B replies to A, it will perform IP-inIPV6 encapsulation, the opposite

IP in IP Encapsulation -> IP-in-IP Encapsulation

Scenario 2



Problem Statement : A and B must exchange data, but all Data must pass through the router T

This constraint is called “Source packet routing”

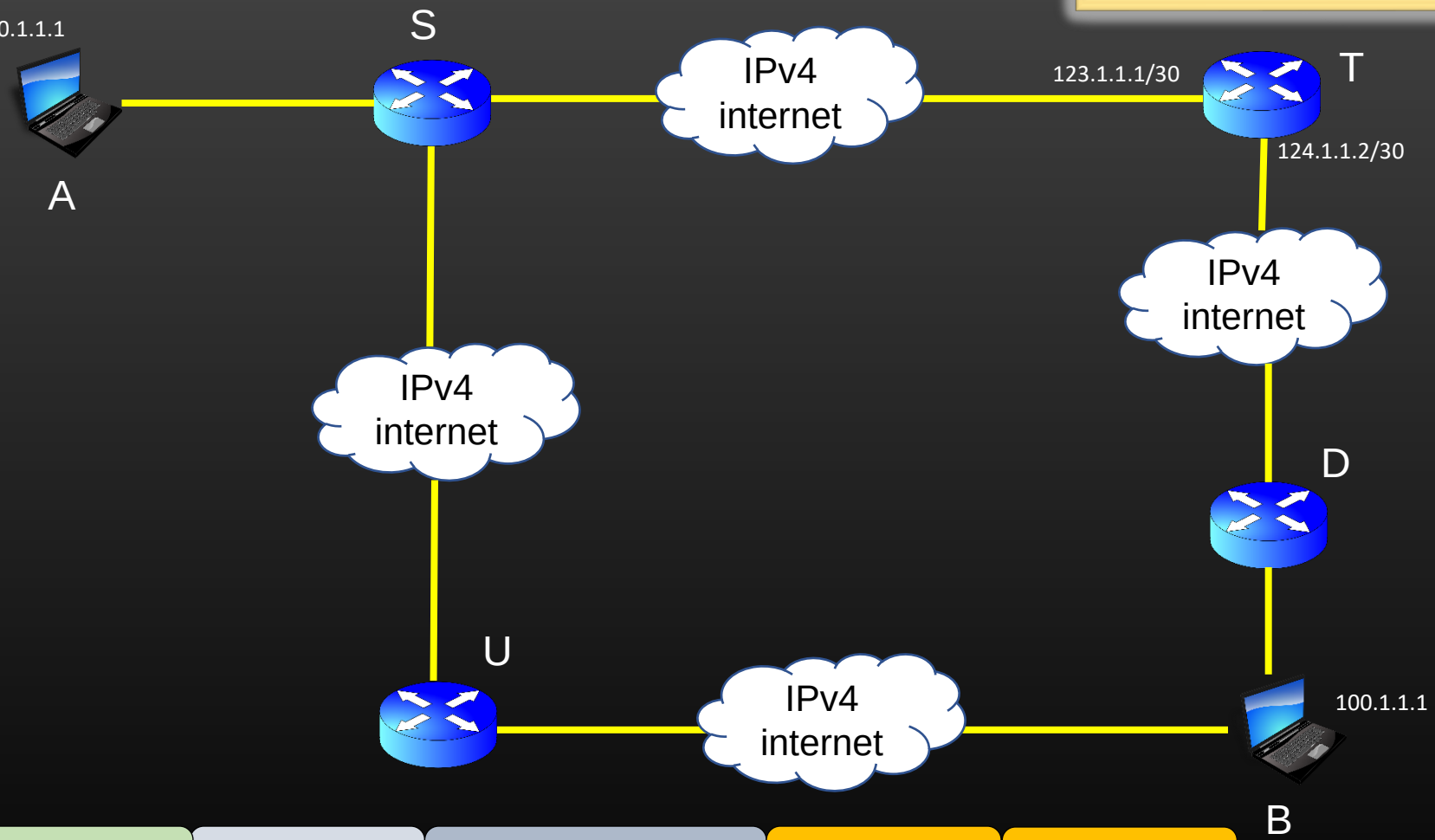
IP in IP Encapsulation -> IP-in-IP Encapsulation

Scenario 2

When pkt is routed from A to T

Ethernet Hdr Type = 0800	Dst : 123.1.1.1 Proto = 4	P a y l o a d

IP to IP transition router



When pkt is routed from T to B

Ethernet Hdr Type = 0800	Dst : 100.1.1.1 Proto = X	Transport Hdr	Appln payload

Pkt generated by A

Ethernet Hdr Type = 0800	Dst : 123.1.1.1 Proto = 4	Dst : 100.1.1.1 Proto = X	Transport Hdr	Appln payload
	Outer IPv4 hdr	Inner IPv4 hdr		

IP in IP Encapsulation -> IP-in-IP Encapsulation

Scenario 2

- What is the Practical Use case for Scenario 2 ?

- When would we require to enforce a constraint on the traffic to always pass through a certain Machine in the network before traffic eventually reaches the final destination ?

Answer : When you send an Instant message on facebook chat window, all your messages first goes to facebook Data centers located in CA, and then from there they are routed to your friend's machine

Facebook Analyze all your data to analyze your interest goals, what you talk about so that they can push relevant advertisement into your news feeds.

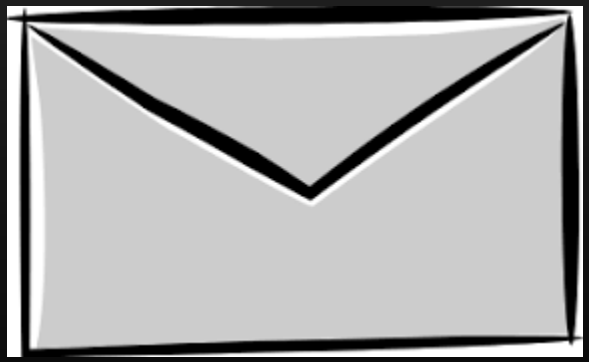
IP in IP Encapsulation -> Summary

Points to Remember

- Source → Encapsulation → Transition Router → Decapsulation → Destination
- The path of the packet from Source to Transition Router is called “Tunnel”
- The inner Network Layer header is read and exposed for the first time only by transition router
- Packet is routed from Source to Transition router using outer IP Hdr only
- While the packet is being routed from Source to Transition router, any intermediate L3 router do not inspect the packet contents beyond outermost IP hdr
- Packet is routed from Transition router to Destination machine using inner IP Hdr only (Inner IP Hdr becomes the outer IP Hdr)
- Tunneling solves the problem in injecting the packet from one Network type into another network type
- Tunneling also enable us to do *“Source packet Routing”*
- In SPR-ing, the path which the packet wishes to take to reach a destination is embedded inside the packet itself (Outer IP hdr)

IP-in-IP Encapsulation

(Hiding inside)



Thank you

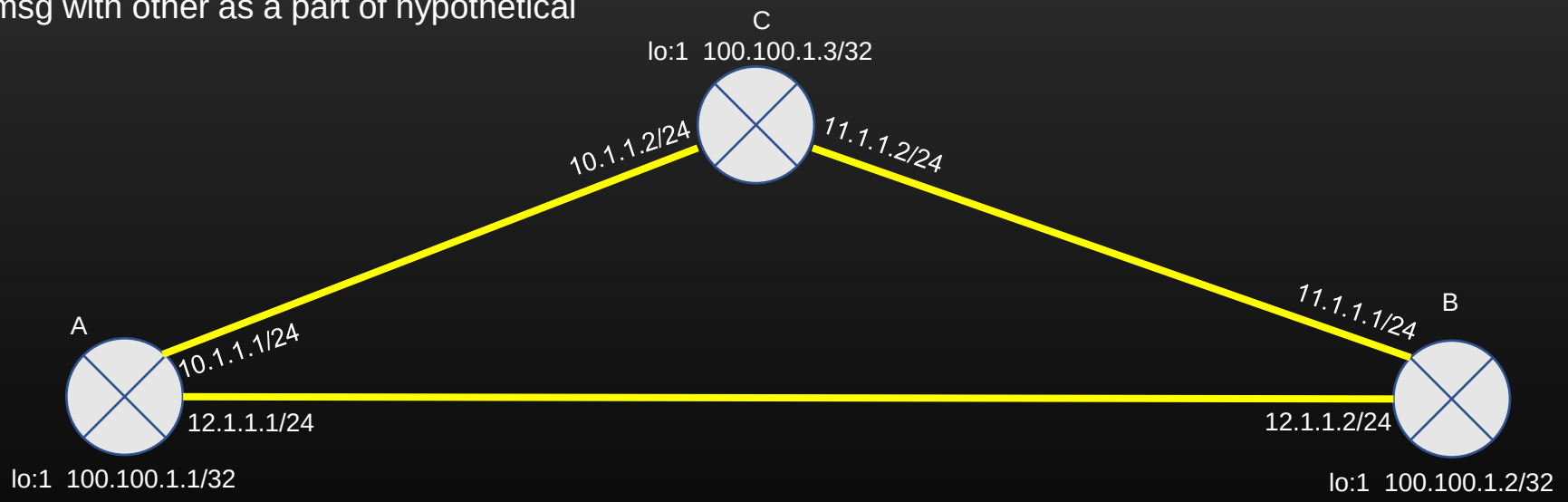
The Concept Of TLVs

Type Length Value

The Concept of TLVs

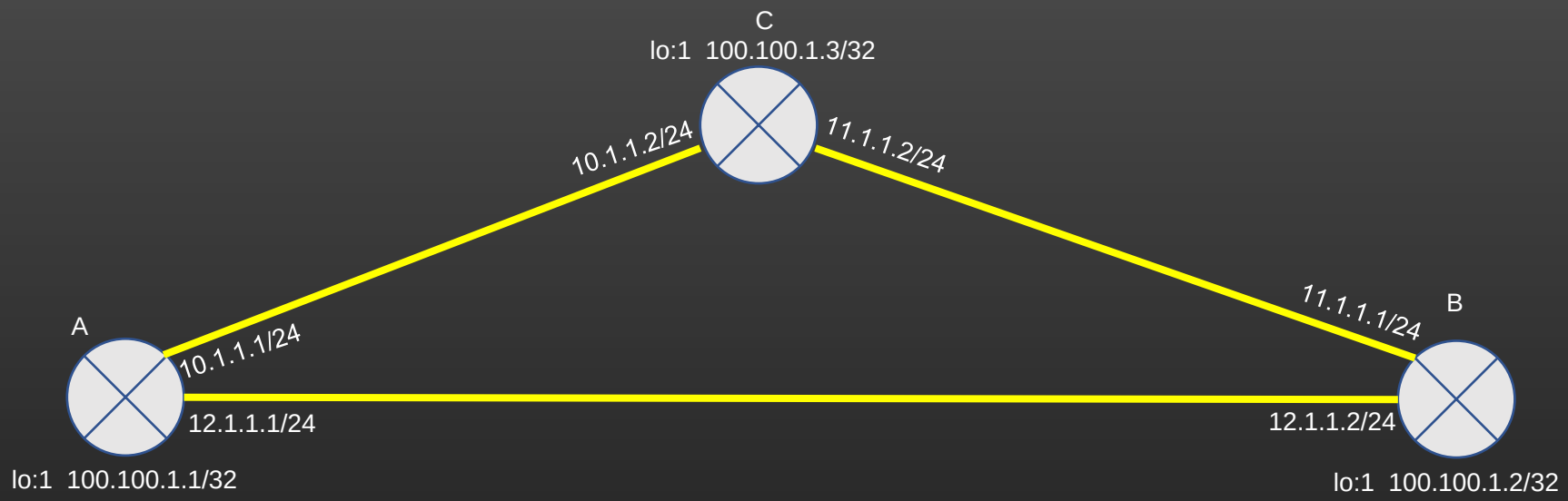
- TLV – Type Length Value
- Let us first try to understand the problem which TLV solves, and then we shall discuss What TLVs are and how they are used
- It is a very common scenario in Networking that Machines often exchange messages with each other. Many Internet routing protocols necessitate Machines to exchange various messages with each other periodically.
- For example, If you remember, Interior Gateway protocols such as OSPF exchange their Link state packets with other routers in the network for their proper functioning.
- To understand the problem, Let's say Machines A, B and C are exchanging the following msg with other as a part of hypothetical functionality P

```
struct xmsg{
        uint loopbck_ip;
        char router_name[32];
        uint if_addr1;
        uint if_addr2;
        uint link1_bw;
        uint link2_bw;
}
```



The Concept of TLVs

```
struct xmsg{
        uint loopbck_ip;
        char router_name[32];
        uint if_addr1;
        uint if_addr2;
        uint link1_bw;
        uint link2_bw;
}
```




```
struct xmsg{
        100.100.1.1
    A
        10.1.1.1
        12.1.1.1
        100
        200
}
```

A

```
struct xmsg{
        100.100.1.2
    B
        11.1.1.1
        12.1.1.2
        110
        220
}
```

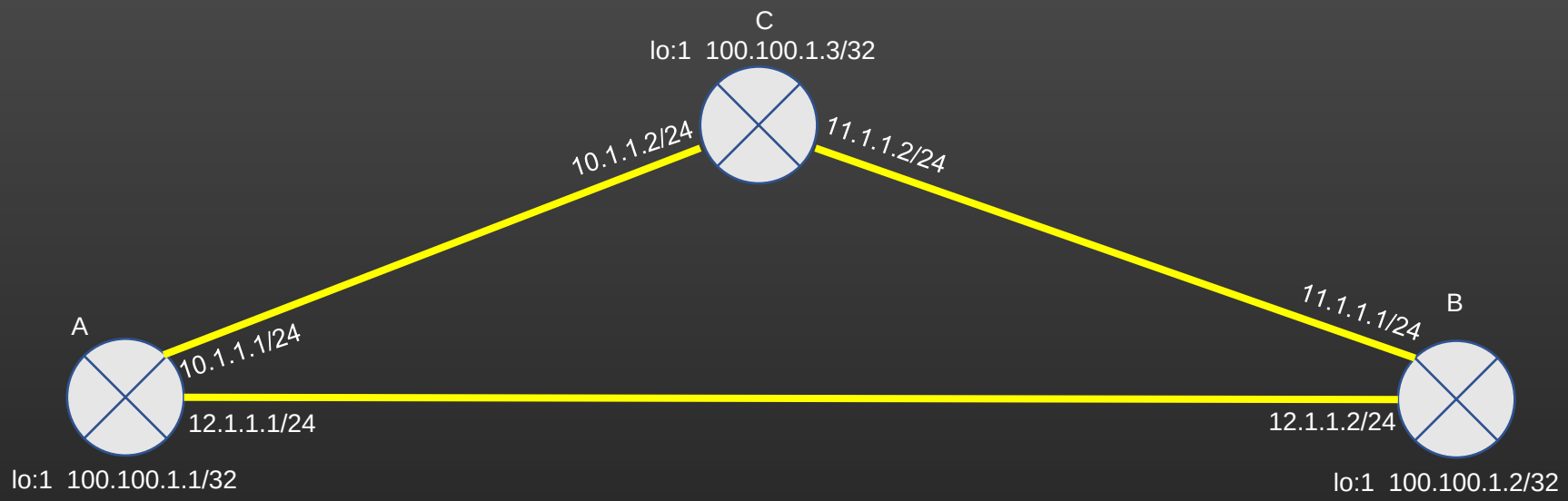
B

```
struct xmsg{
        100.100.1.3
    C
        10.1.1.2
        11.1.1.2
        90
        190
}
```

C

The Concept of TLVs

```
struct xmsg{
        uint loopbck_ip;
        char router_name[32];
        uint if_addr1;
        uint if_addr2;
        uint link1_bw;
        uint link2_bw;
}
```



- So, When machine B/C when receive this msg from machine A over the network, B/C can simply read the msg as as usual :

```
struct xmsg *recv_msg = (struct xmsg *)buffer;
recv_msg->uint_loopback_ip;
recv_msg->router_name;
recv_msg->if_addr1;
recv_msg->if_addr2;
recv_msg->link1_bw;
recv_msg->link2_bw;
```

So ?? What's the problem ?

The Concept of TLVs

- The problem in such exchange of messages arises due to heterogeneity of communicating machines
- Heterogeneity reasons could be mannnyyy ….
  - Different manufacturing vendors
  - Using different Hardware and Technologies
  - Using Different C compilers
  - And so on . . .

- We cannot ask all the vendors around the world to manufacture their network equipment's using Identical technologies and hardware ！

- So let us try to understand the technical glitches that arises due to heterogeneity of the communicating machines in the network

We will discuss two scenarios :

- When machines are distinct and incompatible
- When selective machines in the network are upgraded

The Concept of TLVs

Ok, before going forward , let us revise our C knowledge a bit …

```
struct xmsg{
        uint loopbck_ip;
        char router_name[32];
        uint if_addr1;
        uint if_addr2;
        uint link1_bw;
        uint link2_bw;
}
```

loopbck_ip
router_name
if_addr1
if_addr2
link1_bw
link2_bw

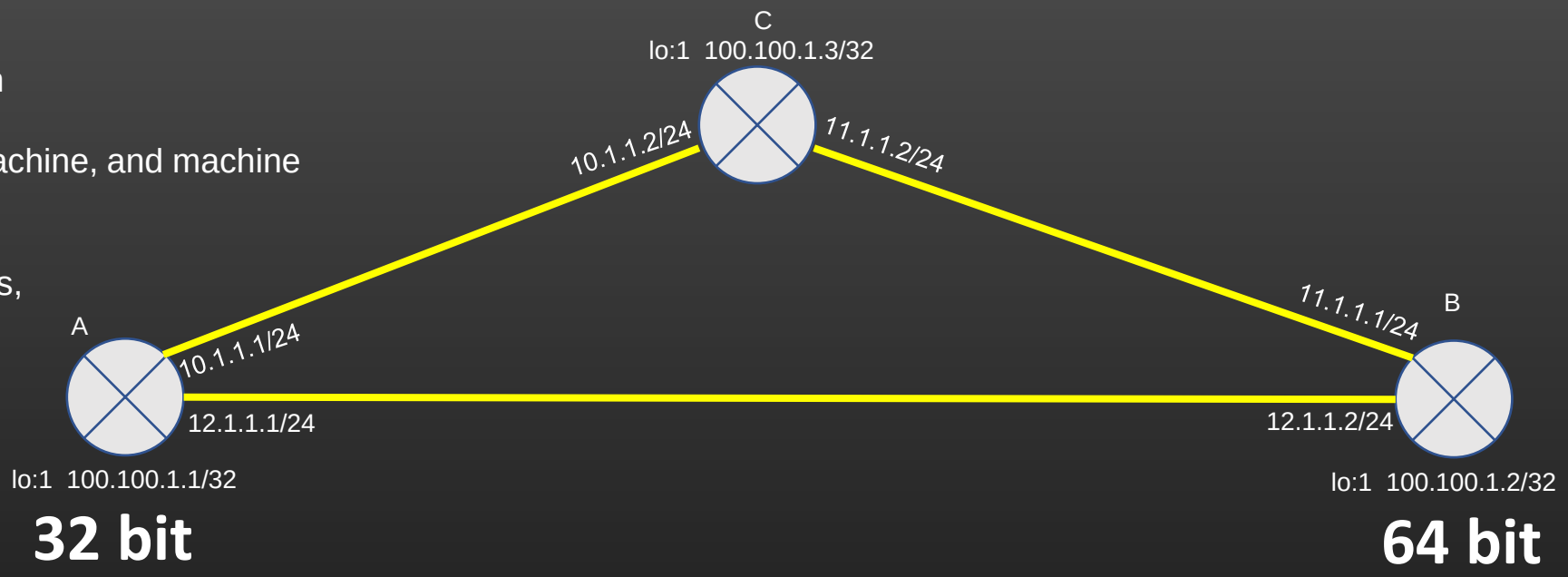
On a 32 bit system

Fields	Size	offset
loopbck_ip	4	0
router_name	32	4
if_addr1	4	36
if_addr2	4	40
link1_bw	4	44
link2_bw	4	48

Struct xmsg *ptr;
ptr->if_addr2 -- reading/writing 4 bytes @40th byte from starting address

The Concept of TLVs

- Ok, now let us see the real problem
- Let us Say Machine A is a 32 bit machine, and machine B is a 64 bit machine.
- It means, sizeof(uint) on A is 4 bytes, whereas it is 8 bytes on B
- Now, let see the xmsg layout on wire when they are generated by machine A and B respectively.




```
struct xmsg{
        uint loopbck_ip;
        char router_name[32];
        uint if_addr1;
        uint if_addr2;
        uint link1_bw;
        uint link2_bw;
}
```

A

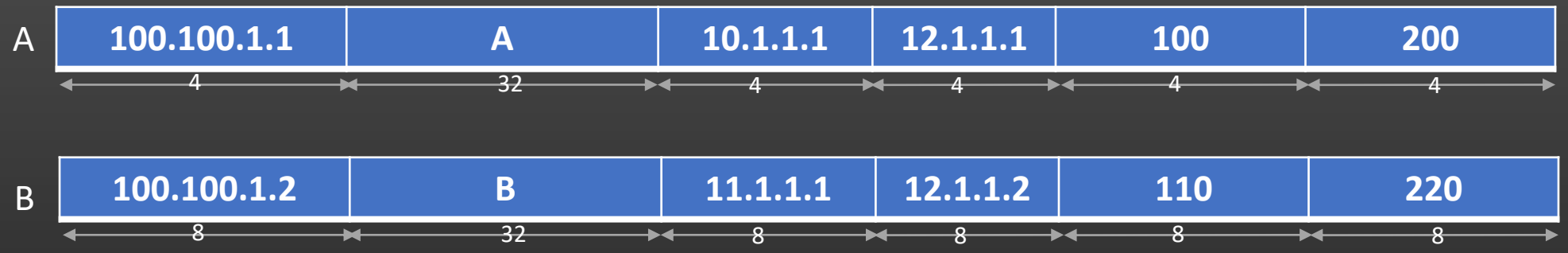
100.100.1.1	A	10.1.1.1	12.1.1.1	100	200
4	32	4	4	4	4

B

100.100.1.2	B	11.1.1.1	12.1.1.2	110	220
8	32	8	8	8	8

The Concept of TLVs

```
struct xmsg{
        uint loopbck_ip;
        char
router_name[32];
        uint if_addr1;
        uint if_addr2;
        uint link1_bw;
        uint link2_bw;
}
```



When A receives xmsg from B, A will typecast the msg according to its belief of definition of xmsg:

- So, When machine A receive the xmsg from machine B over the network, A will type cast the msg according to its own definition of struct xmsg :

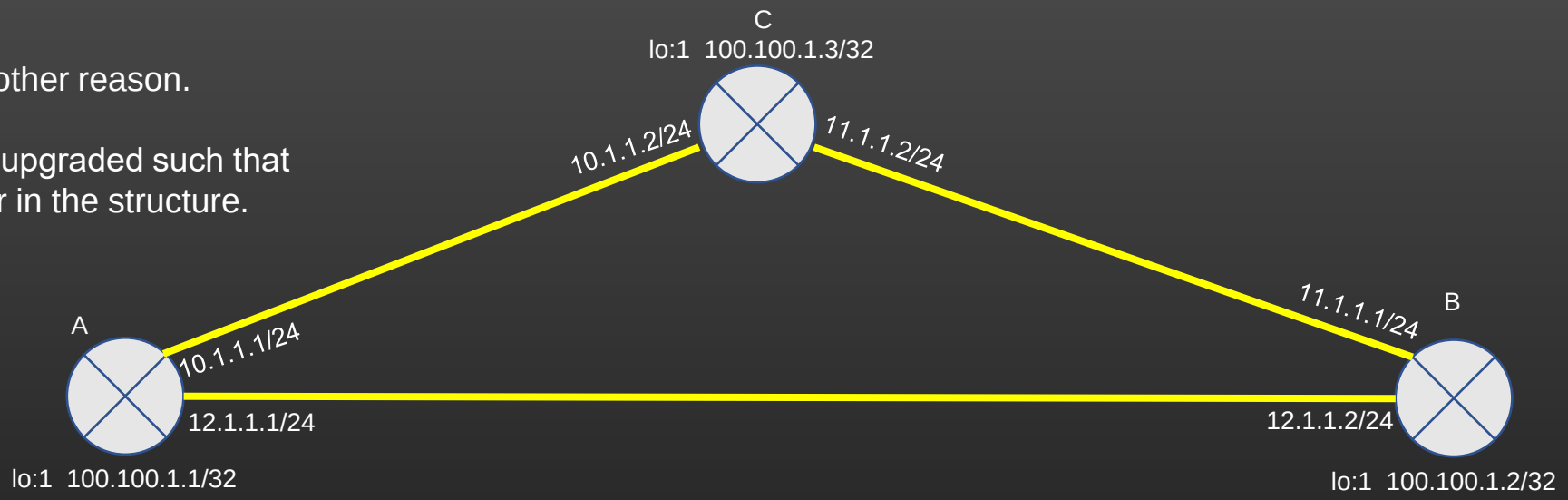
```
struct xmsg *recv_msg = (struct xmsg *)buffer;
recv_msg->uint_loopback_ip;    /*Instead of reading 8 bytes, A will read only 4 bytes*/
recv_msg->router_name;         /*From B's perspective, it is 8th byte from start of msg, from A's perspective it is 4th byte from start of msg*/
recv_msg->if_addr1;
recv_msg->if_addr2;
recv_msg->link1_bw;
recv_msg->link2_bw;
```

So, A ends up in reading a Garbage, leading to Data corruption on A. It happened because size of Data types on B is different from that of A.

The Concept of TLVs

- Lets see the same problem due to other reason.
- Let us Say Machine A's software is upgraded such that it introduces a new member in the structure.

```
struct xmsg{
        uint loopbck_ip;
        char router_name[32];
        uint if_addr1;
        uint if_addr2;
        char if_mac1[6];
        char if_mac2[6];
        uint link1_bw;
        uint link2_bw;
}
```



A

100.100.1.1	A	10.1.1.1	12.1.1.1	100	200
4	32	4	4	4	4

A.1

100.100.1.2	B	11.1.1.1	12.1.1.2	If1_MAC	If2_MAC	100	200
8	32	8	8	6	6	4	4

The Concept of TLVs

A.1

```
struct xmsg{
        uint loopbck_ip;
        char router_name[32];
        uint if_addr1;
        uint if_addr2;
        char if_mac1[6];
        char if_mac2[6];
        uint link1_bw;
        uint link2_bw;
}
```

B and C

```
struct xmsg{
        uint loopbck_ip;
        char router_name[32];
        uint if_addr1;
        uint if_addr2;
        uint link1_bw;
        uint link2_bw;
}
```

A

A.1

Machines B and C, when receives the new msg A.1 generated by machine A, they will try to read the msg according to their own definition of struct xmsg. Again Data corruption !

So many problems !! Vendor manufacturer has invented his new patented technology but he cannot upgrade his software with new technology because other machines in the network wont work with his new version of Software ! Competitors are happy ! Funny !!

The Concept of TLVs

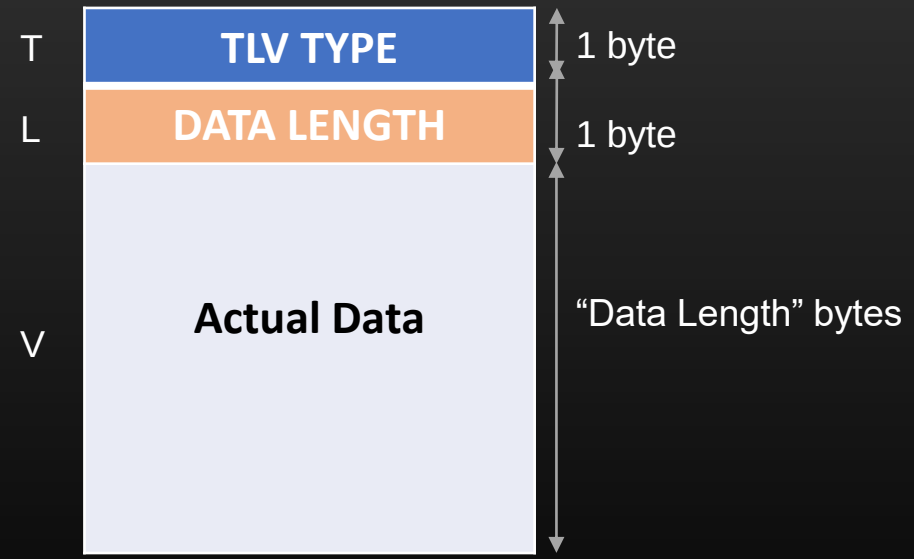
- Networking is a field where various network equipment's being manufactured by various vendors, need to work In complete cooperation and harmony with each other for the network protocol to work.

- Machines need to comply with each other for the network functionality to work correctly, yet at the same time Network Vendors should be free to innovate/upgrade/update their software without breaking the existing compliance with the other Machines deployed in the network.

The concept of TLVs Solves these problems very easily

The Concept of TLVs

- TLV (Type length value)
  - Is a mechanism to encode the data in the format that is independent of
    - Machine Architecture
    - Underlying Operating system
    - Compiler
    - Programming language
  - TLVs has three components :

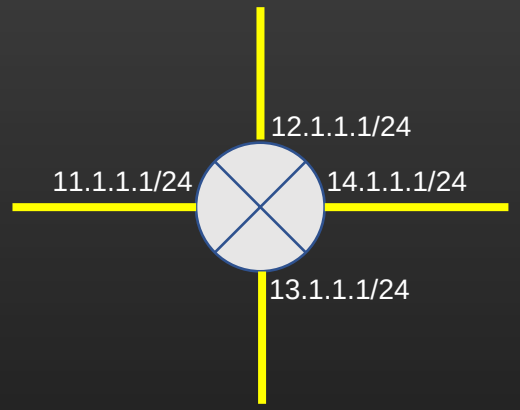


TLV TYPE – TLV UNIQUE IDENTIFER (0-255)

TLV LENGTH – The length of the actual data

The Concept of TLVs

- Example :
  - Suppose Machine A wants to send machine B, the set of all IP addresses which is configured on all its interfaces



132
20
201392385
24
234946817
24
218169601
24
184615169
24

  - We can take any number as TLV type. Let's take it as 132.
  - Next we need to define the definition of TLV 132 :
    - 4 byte integer number (which is ip address)
    - 1 byte mask value
    - This is called *TLV definition*
  - Any machine which is suppose to process this TLV when received, us suppose to be aware of TLV definition.

The Concept of TLVs

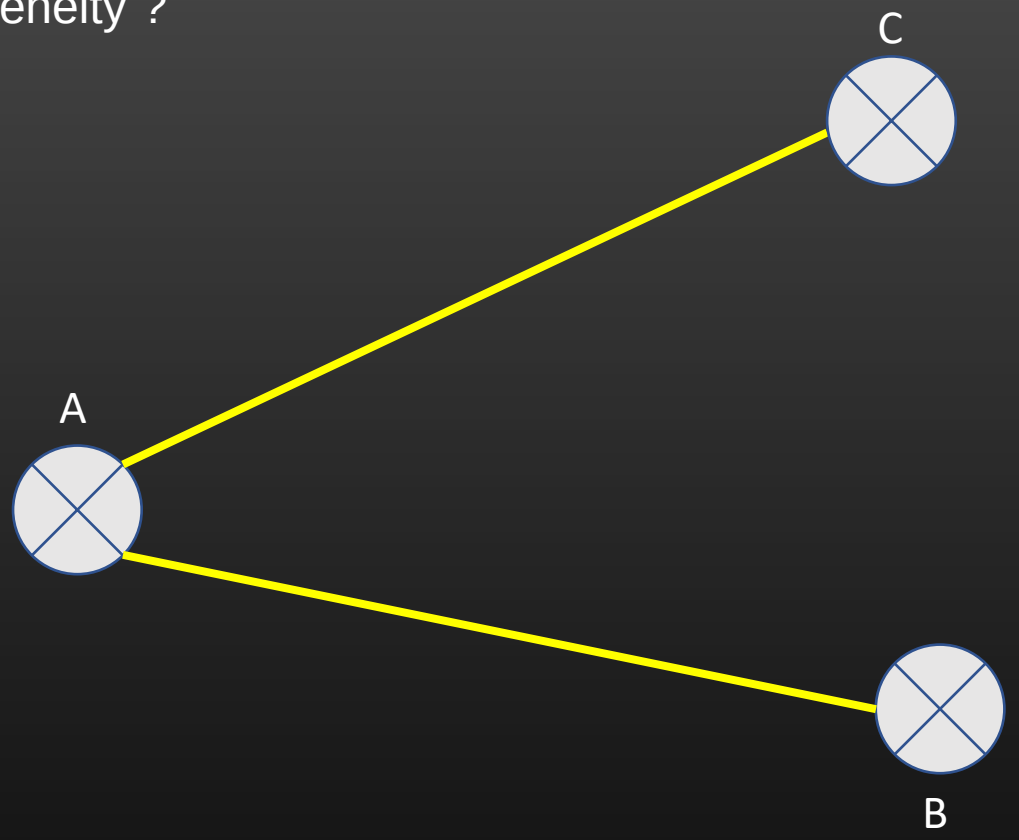
132
20
201392385
24
234946817
24
218169601
24
184615169
24

- When receiving machine receives this TLV :
  - It reads first byte
  - Now it knows that it is TLV 132
  - It means, its unit data size is 5 bytes,
    4 bytes of ip address followed by 1 byte of mask value
  - Read next 1 byte which is 20
  - Divide 20 by 5 = 4
  - Now, machine knows it has four occurrence of unit data type
  - Iterate over rest of the data and read all units of data

The Concept of TLVs

How TLVs solve the problem of machine heterogeneity ?

132
20
201392385
24
234946817
24
218169601
24
184615169
24



Let us suppose, machine A sends this TLV to machine B and C
Let say, machine B is 32 bit machine, and C is 16 bit machine

Let us see, how B and C decode this TLV . . .

The Concept of TLVs

32/16 bit machine

132
20
201392385
24
234946817
24
218169601
24
184615169
24

Code discussion ….

The Concept of TLVs

Had we sent the data as simple C structure on the wire as below :

```
struct tlv132{
        unsigned int ip_address;
        char mask;
}
```

Then it would have been problem for receiving machines, if their hardware architecture differs from sending machine.

For 32 bit machine : structure size is 5 bytes
For 16 bit machine : structure size is 3 bytes

Receiving Machines which are non compliant with sending machine would end up reading garbage !

```
struct tlv132 *ptr = (struct tlv132 *)recv_msg;   /*recv_msg us 4 byte ip address,

ptr->ip_address ;  /*This line would read 4 bytes on 32 bit machine, and 2 bytes
```

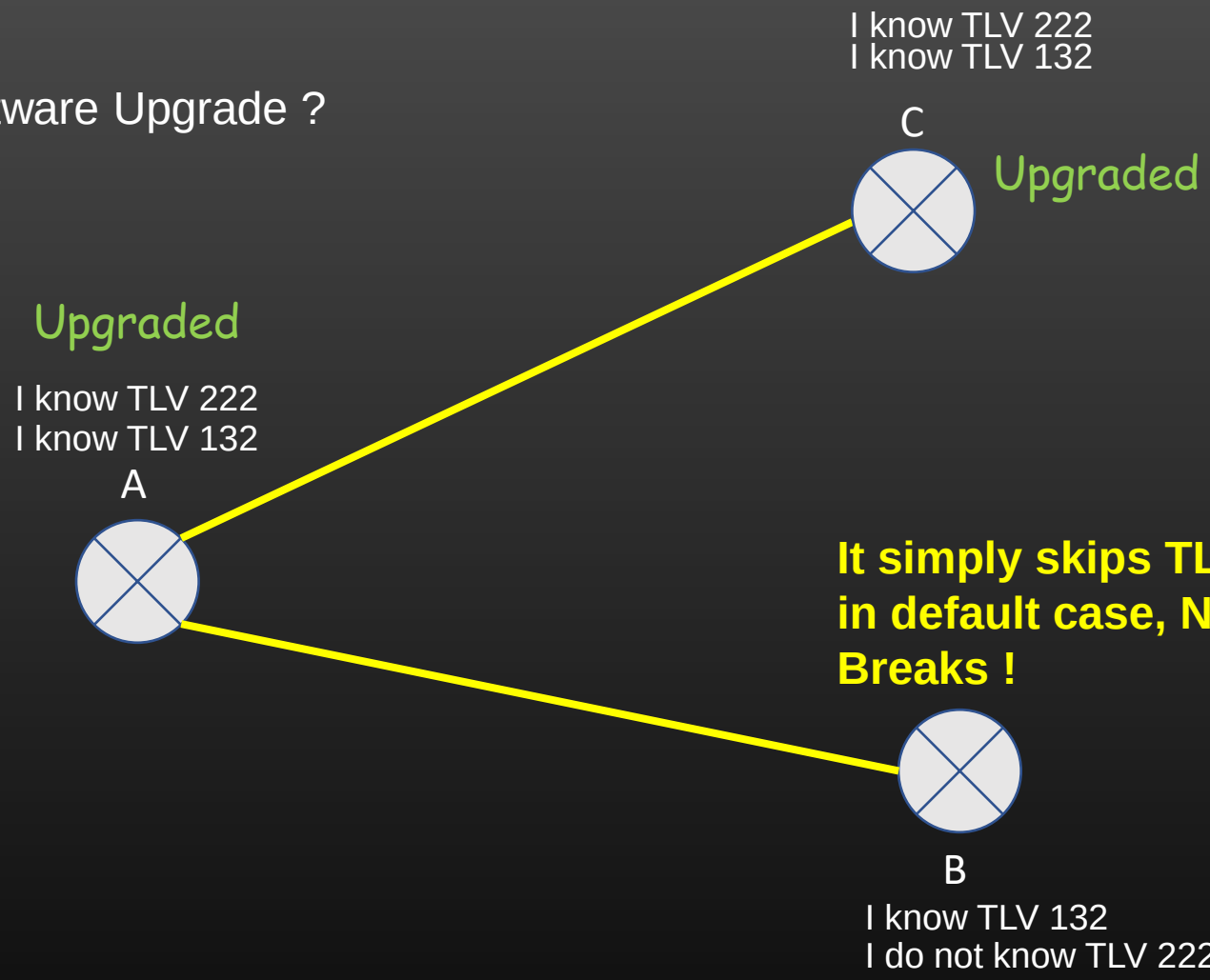
To Sum up : TLVs are all about *Send and Read data byte by byte* , and every machine MUST know TLV TYPE definition

We have just learned, how TLVs solves the problem of machine heterogeneity !!

The Concept of TLVs

How TLVs solve the problem of Software Upgrade ?

132
20
201392385
24
234946817
24
218169601
24
184615169
24
222
12
08:00:27:3e:97:62
08:00:27:ce:90:78



TLV definition :

- 6 bytes (which is mac address)

We have just learned, how TLVs solves the problem of software Upgrade !!

Streams

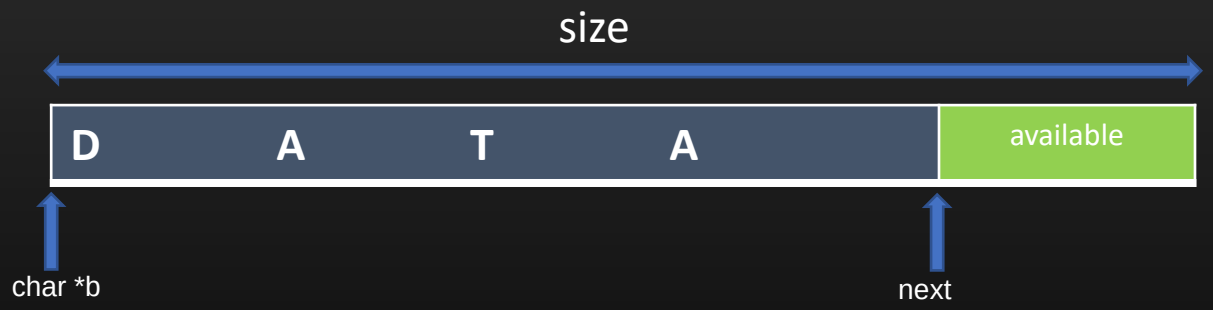
- Design a Data structure to create (serialize) and read (De-serialize) TLVs
- Data structure : *Stream*
- It resembles to type writer – start with the next line when there is no space left in the current line

The Concept of TLVs

Coding Assignment

- Streams (also called serialize-buffer)

```
typedef struct serialized_buffer{

    char *b;
    int size;
    int next;
} ser_buff_t;
```



The Concept of TLVs

Coding Assignment

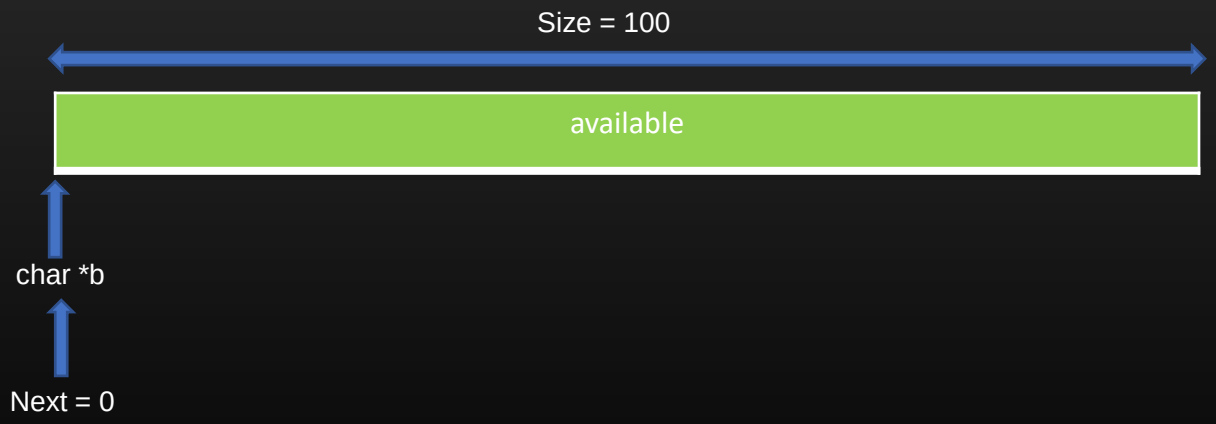
- Streams (also called serialize-buffer)

```
typedef struct serialized_buffer{

    char *b;
    int size;
    int next;
} ser_buff_t;
```

```
void
init_serialized_buffer (ser_buff_t **b){

    (*b) = (ser_buff_t *)calloc(1, sizeof(ser_buff_t));

    (*b)->b = calloc(1, SERIALIZE_BUFFER_DEFAULT_SIZE); /*const , say 100*/

    (*b)->size = SERIALIZE_BUFFER_DEFAULT_SIZE;

    (*b)->next = 0;
}
```

Usage :

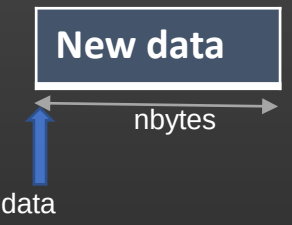
```
ser_buff_t *stream;
init_serialized_buffer(&stream);
```



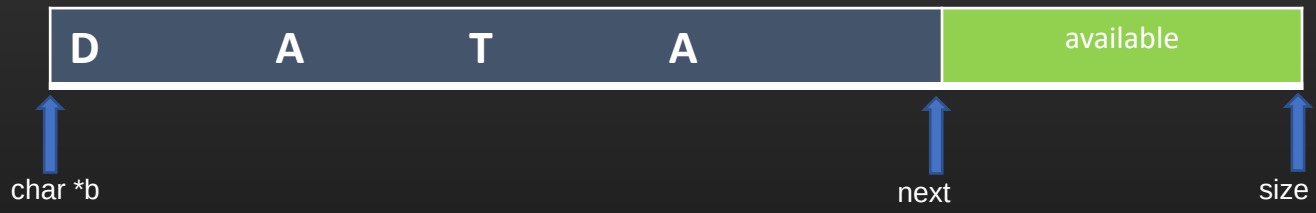
The Concept of TLVs

Coding Assignment

- Streams (also called serialize-buffer)




```
void serialize_data (ser_buff_t *buff, char *data, int nbytes){

    int available_size = buff->size - buff->next;
    char isResize = 0;

    while(available_size < nbytes){
        buff->size = buff->size * 2;
        available_size = buff->size - buff->next;
        isResize = 1;
    }

    if(isResize == 0){
        memcpy((char *)buff->b + buff->next, data, nbytes);
        buff->next += nbytes;
        return;
    }

    // resize of the buffer
    buff->b = realloc(buff->b, buff->size);
    memcpy((char *)buff->b + buff->next, data, nbytes);
    buff->next += nbytes;
    return;
}
```

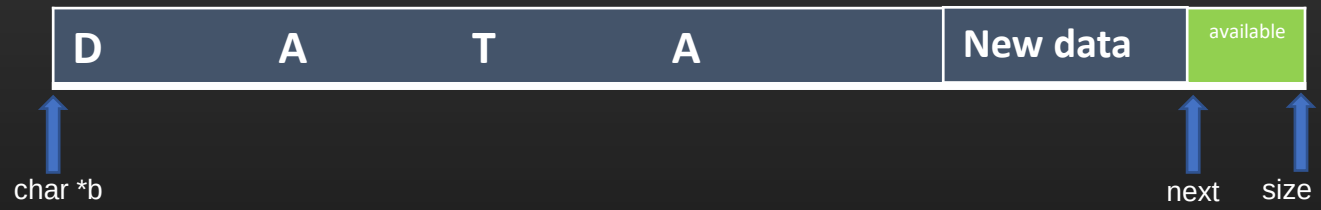


The Concept of TLVs

Coding Assignment

- Streams (also called serialize-buffer)

```
void serialize_data (ser_buff_t *buff, char *data, int nbytes){

    int available_size = buff->size - buff->next;
    char isResize = 0;

    while(available_size < nbytes){
        buff->size = buff->size * 2;
        available_size = buff->size - buff->next;
        isResize = 1;
    }

    if(isResize == 0){
        memcpy((char *)buff->b + buff->next, data, nbytes);
        buff->next += nbytes;
        return;
    }

    // resize of the buffer
    buff->b = realloc (buff->b, buff->size);
    memcpy((char *)buff->b + buff->next, data, nbytes);
    buff->next += nbytes;
    return;
}
```



If there is no space to accommodate new data,
Double the size of entire buffer while preserving the
Exiting content

The Concept of TLVs

Coding Assignment

- Serializing the TLVs

```
ser_buff_t *stream;
init_serialized_buffer(&stream);

char data = 32;
serialize_data (stream, &data, 1 );

data = 20;
serialize_data (stream, &data, 1 );

unsigned int ip =  201392385;
serialize_data (stream, &ip, 4 );

char mask = 24;
serialize_data (stream, &mask, 1 );

ip =  234946817;
serialize_data (stream, &ip, 4 );

mask = 24;
serialize_data (stream, &mask, 1 );

ip =  218169601;
serialize_data (stream, &ip, 4 );

mask = 24;
serialize_data (stream, &mask, 1 );

ip =  184615169;
serialize_data (stream, &ip, 4 );

mask = 24;
serialize_data (stream, &mask, 1 );
```

```
char data = 222;
serialize_data (stream, &data, 1 );

data = 12;
serialize_data (stream, &data, 1 );

char mac1[6]  =  {8, 0, 27, 3e, 97, 62};
serialize_data (stream, mac1, 6 );


char mac2[6]  =  {8, 0, 27, ce, 90, 78};
serialize_data (stream, mac2, 6 );
```

Data to be send as TLV :
stream->b with size stream->next

132
20
201392385
24
234946817
24
218169601
24
184615169
24
222
12
08:00:27:3e:97:62
08:00:27:ce:90:78

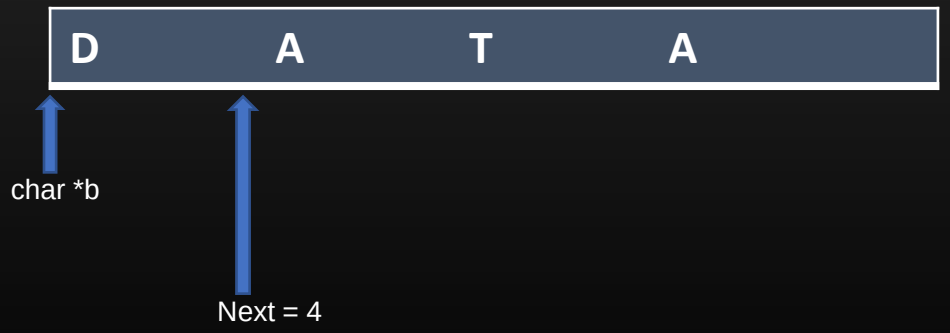
The Concept of TLVs

Coding Assignment

- De-Serializing the TLVs

```
void
de_serialize_data (char *dest, ser_buff_t *b, int size){

    memcpy(dest, b->b + b->next, size);

    b->next += size;
}
```


```
unsigned int dest;
de_serialize_data ((char *)&dest, b, 4);
```




```
de_serialize_data ((char *)&dest, b, 4);
```

The Concept Of TLVs

Type Length Value

Thank you