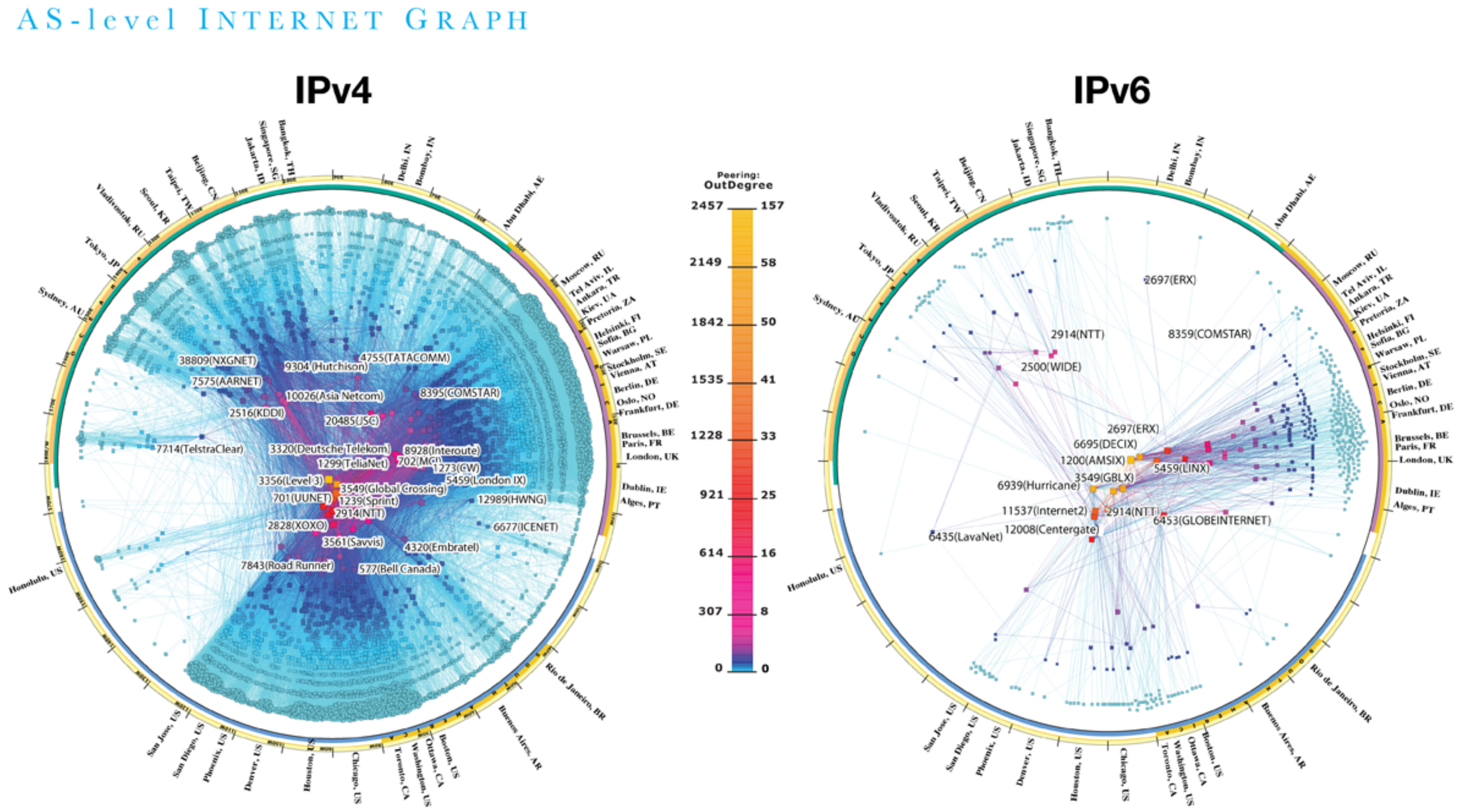
# IP addressing and routing

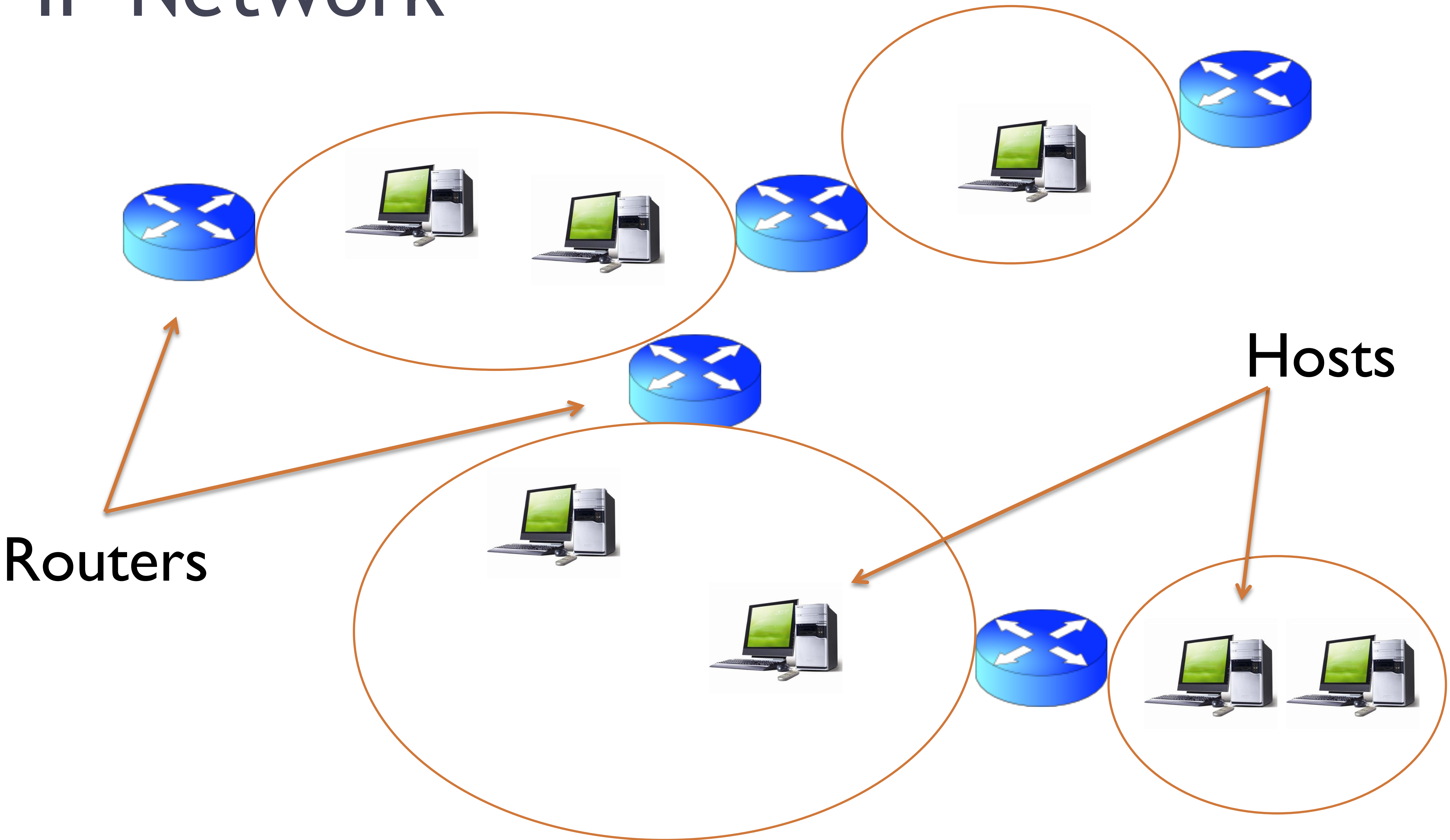

- Lorenzo Bracciale
- Donato Battaglino

# IP Address



AS-level INTERNET GRAPH
IPv4
IPv6
Peering: OutDegree
2457 157
2149 58
1842 50
1535 41
1228 33
921 25
614 16
307 8
0 0
38809(NXGNET)
9304(Hutchison)
4755(TATACOMM)
7575(AARNET)
10026(Asia Netcom)
8395(COMSTAR)
2516(KDDI)
20485(JSC)
7714(TelstraClear)
3320(Deutsche Telekom)
8928(Interoute)
1299(TeliaNet)
702(MCI)
1273(CW)
3356(Level 3)
5459(London IX)
3549(Global Crossing)
701(UUNET)
1239(Sprint)
12989(HWNG)
2914(NTT)
2828(XOXO)
6677(ICENET)
3561(Savvis)
4320(Embratel)
7843(Road Runner)
577(Bell Canada)
2697(ERX)
2914(NTT)
8359(COMSTAR)
2500(WIDE)
2697(ERX)
6695(DECIX)
1200(AMSIX)
5459(LINX)
3549(GBLX)
6939(Hurricane)
11537(Internet2)
2914(NTT)
6453(GLOBEINTERNET)
12008(Centergate)
6435(LavaNet)

# IP Network



# IP Address Anatomy

Each network interface connected to an IP network MUST have an UNIQUE 32-bit identifier called IP Address

Example:
11010001 01010101 10000001 01100011 machine representation

Humans don't like long binary string and prefer to use the "dotted decimal" notation:
Example
209.85.129.99 human representation

Well, also IP address expressed in dotted decimal notation are hard to remember. Names sounds better
Example extra service
209.85.129.99 <---DNS---> www.google.it

# Exercise 1: binary/decimal conversion

224

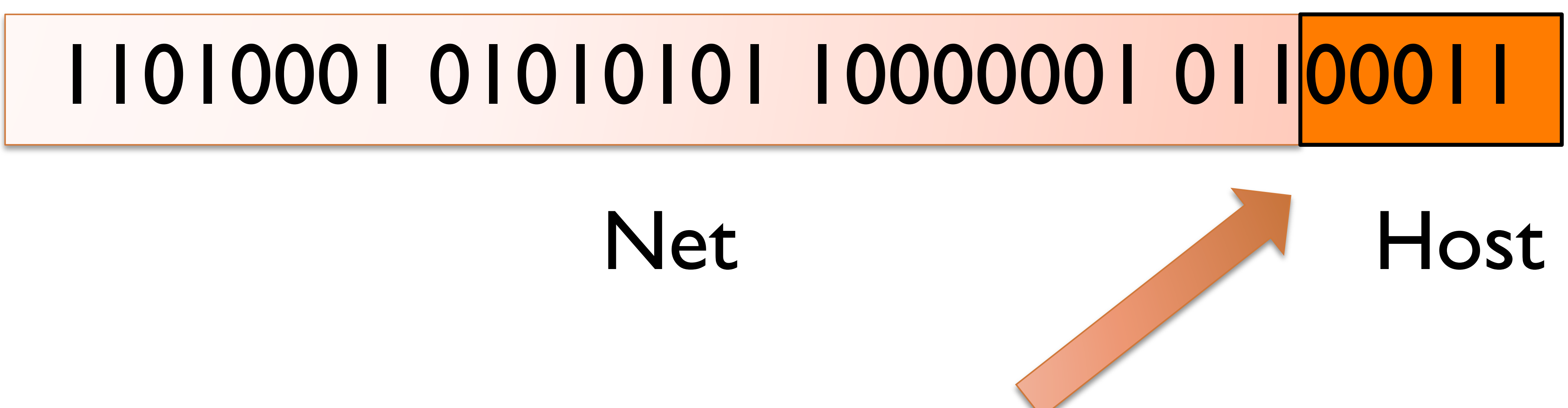
32

192

11010001 01010101 10000001 01100011

# IP Address Anatomy

- IP networks are logically divided **subnet**, so that:
  - Inside each subnet, two hosts must directly communicate using L2 technology (e.g. ethernet, wifi …)
  - Across different subnet, hosts communicate through routers (one or more)
  - IP addresses of the same subnet have same first X bits ("net" part) and a different 32-X bits ("host" part)




How long is the network part of an IP address??

# IP Address Anatomy

- How long is the network part of an IP address??
  - We need some extra information (from 1984, IP address does not give us any hint!)

Net | Host

IP Address

11010001 01010101 10000001 01100011

11111111 11111111 11111111 11100000

Subnet Mask

Alternatively we can use the dotted notation:

Example: 209.85.129.99 , mask 255.255.255.224

Or the "slash" prefix

Example: 209.85.129.99 / 27

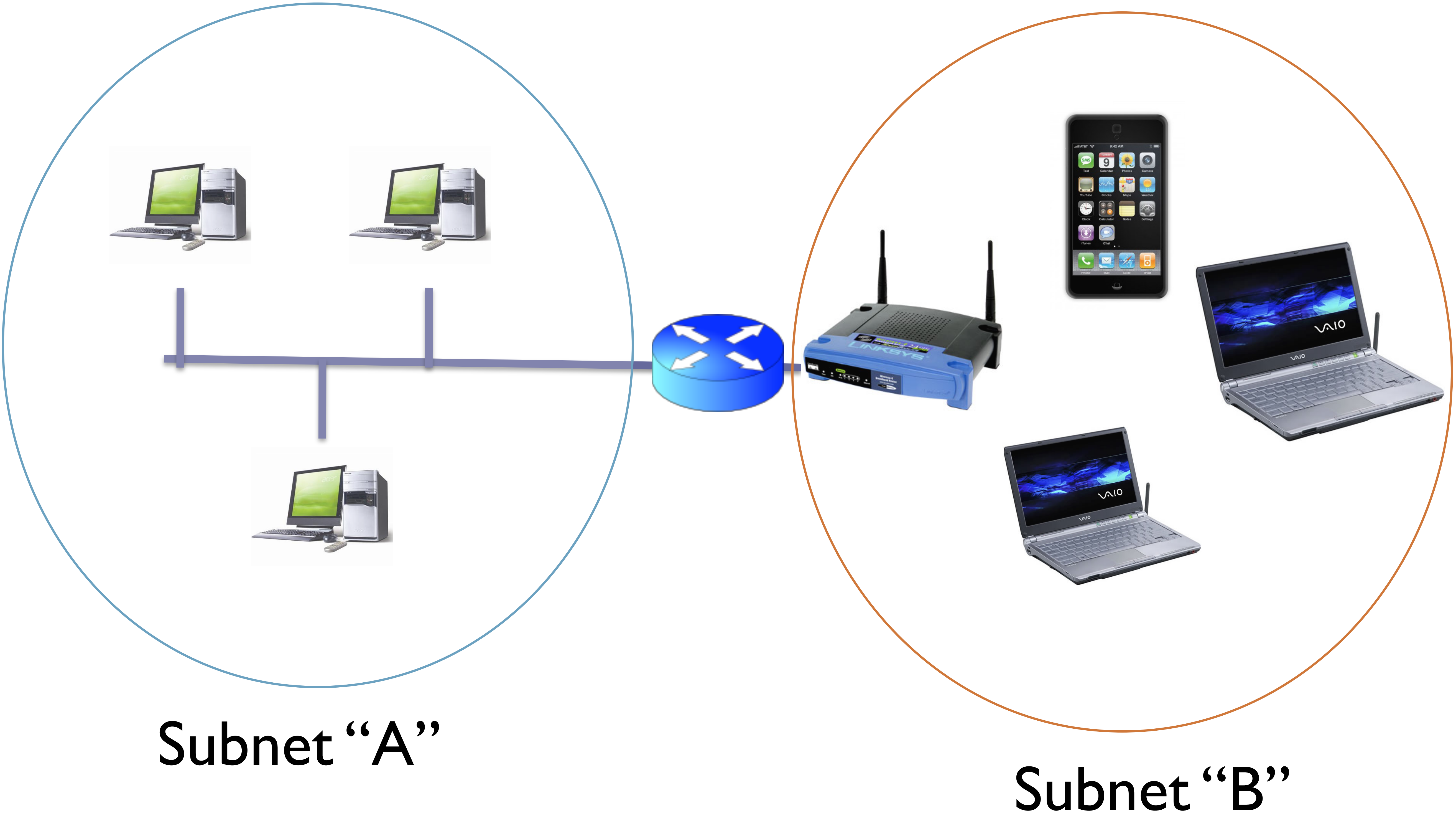
# IP Address Anatomy

- Each subnet has two special IP addresses:
  - Net address (all the bits in the host part are 0)
  - Broadcast address (all the bits in the host part are 1)
- Basically a subnet is identified by the net address and the mask

Example : find the network and broadcast addresses of host 209.85.129.99/27

| | |
|---|---|
| 209.85.129.99 (IP addr host) | 11010001 01010101 10000001 011 \| 00011 |
| 255.255.255.224 (Subnet Mask) | 11111111 11111111 11111111 111 \| 00000 |
| 209.85.129.96 (IP addr network) | 11010001 01010101 10000001 011 \| 00000 |
| 209.85.129.127 (IP addr broadcast) | 11010001 01010101 10000001 011 \| 11111 |

# Subnet: example



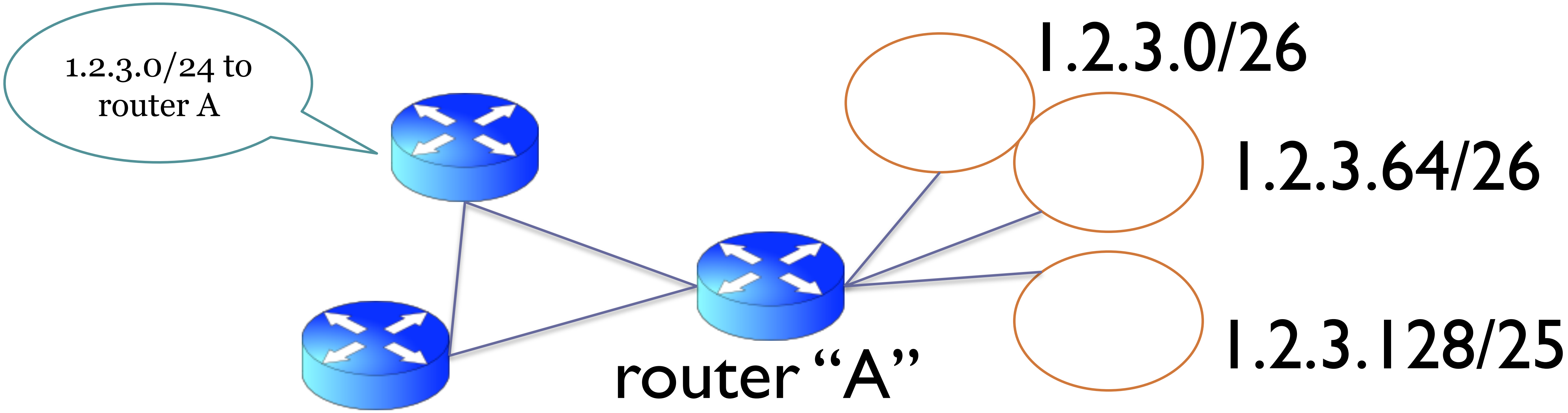
# Exercise 2: network address

One host has the following address network 160.80.103.21/18 :

- Calculate broadcast/network address
- How many host can be in this subnet?

# Subnetting/Supernetting

- It is useful to aggregate different subnet
  - otherwise routers should have information for every subnet in the whole Internet
- This tecnique is called "Supernetting"
  - Example: 1.2.3.0/25 and 1.2.3.128/25 can be aggregated in one supernet: 1.2.3.0/24
- Given a "supernet" we can split it in several subnets (subnetting):
  - Example: 1.2.3.0/24 can be divided in: 1.2.3.0/26, 1.2.3.64/26 , 1.2.3.128/25



# Subnetting: example

Given this address block `160.80.1.128/25`:

Obtain four /27 subnets:

```
160.80.1.128/27    10100000.01010000.00000001.10000000
160.80.1.160/27    10100000.01010000.00000001.10100000
160.80.1.192/27    10100000.01010000.00000001.11000000
160.80.1.224/27    10100000.01010000.00000001.11100000
```

Obtain two/27 and one /26 subnets:

```
160.80.1.128/26    10100000.01010000.00000001.10000000
160.80.1.192/27    10100000.01010000.00000001.11000000
160.80.1.224/27    10100000.01010000.00000001.11100000
```

1 subnet /25 → 2 subnet /26 → 4 subnet /27 ….

*Things getting more complicated with different size of subnetworks (e.g. 1 / 27, 2 /30 , 3 /28)*

# Subnetting: sanity checks!

1. All the subnets have “all 0” in the net part of the address?
2. Subnets do not overlaps?

# Common Orrors

- You have 1.2.3.0/24: place a subnet /30 and a /27

1.2.3.0/30 *from 1.2.3.0 to 1.2.3.3*

1.2.3.4/27 *from 1.2.3.4 to 1.2.3.35*

Error

00000001 00000010 00000011 00000100

This is not a valid net address (host part is not "all 0")

# Common Orrors

- You have 1.2.3.0/24: place a subnet /30 and a /27

1.2.3.0/27 *from 1.2.3.0 to 1.2.3.31*

1.2.3.32/30 *from 1.2.3.32 to1.2.3.35*

Correct!

If you start from the biggest subnet, addesses are correct

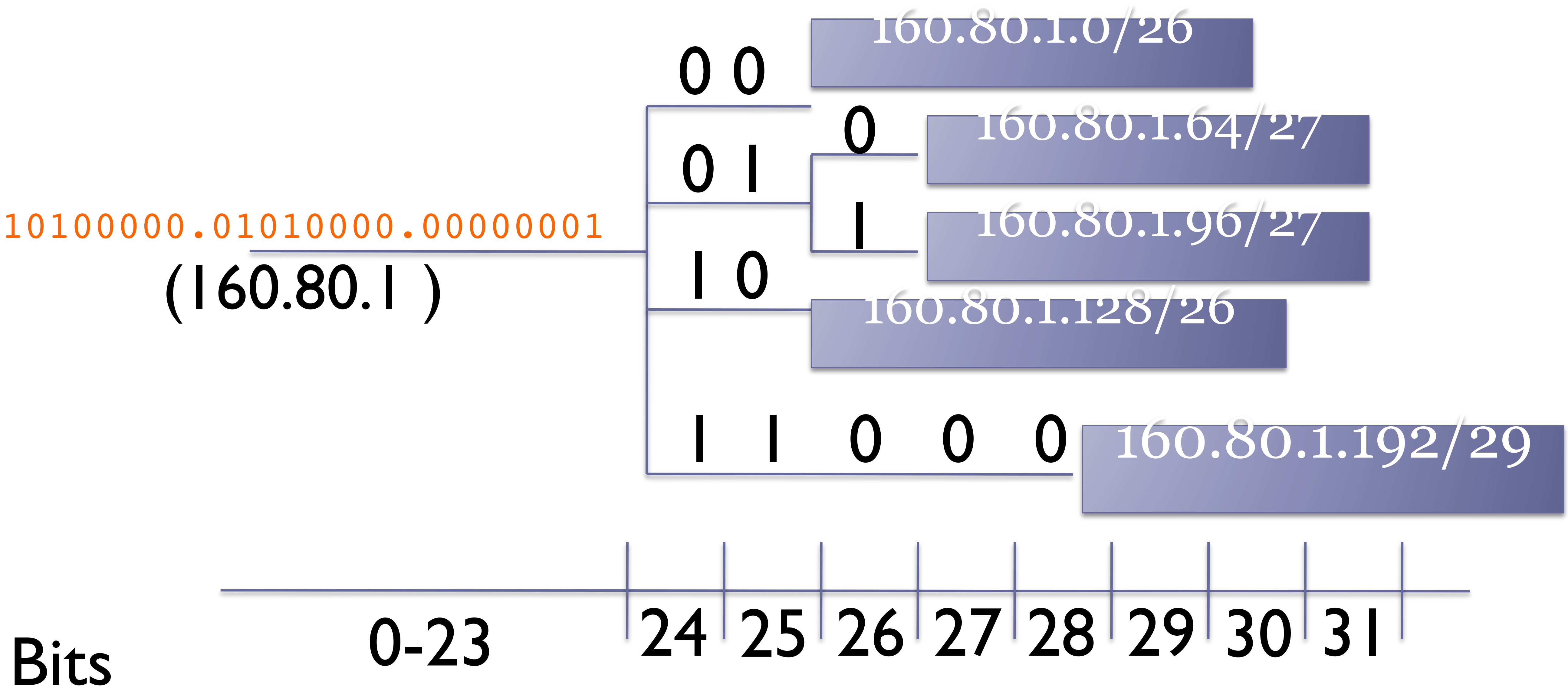
(even if this is not the only correct solution!)

As well as:160.80.64/26 (2^6 host) + 160.80.1.128/26 (2^6 host)

**does not** form a 1.2.3.64/25…think in binary!

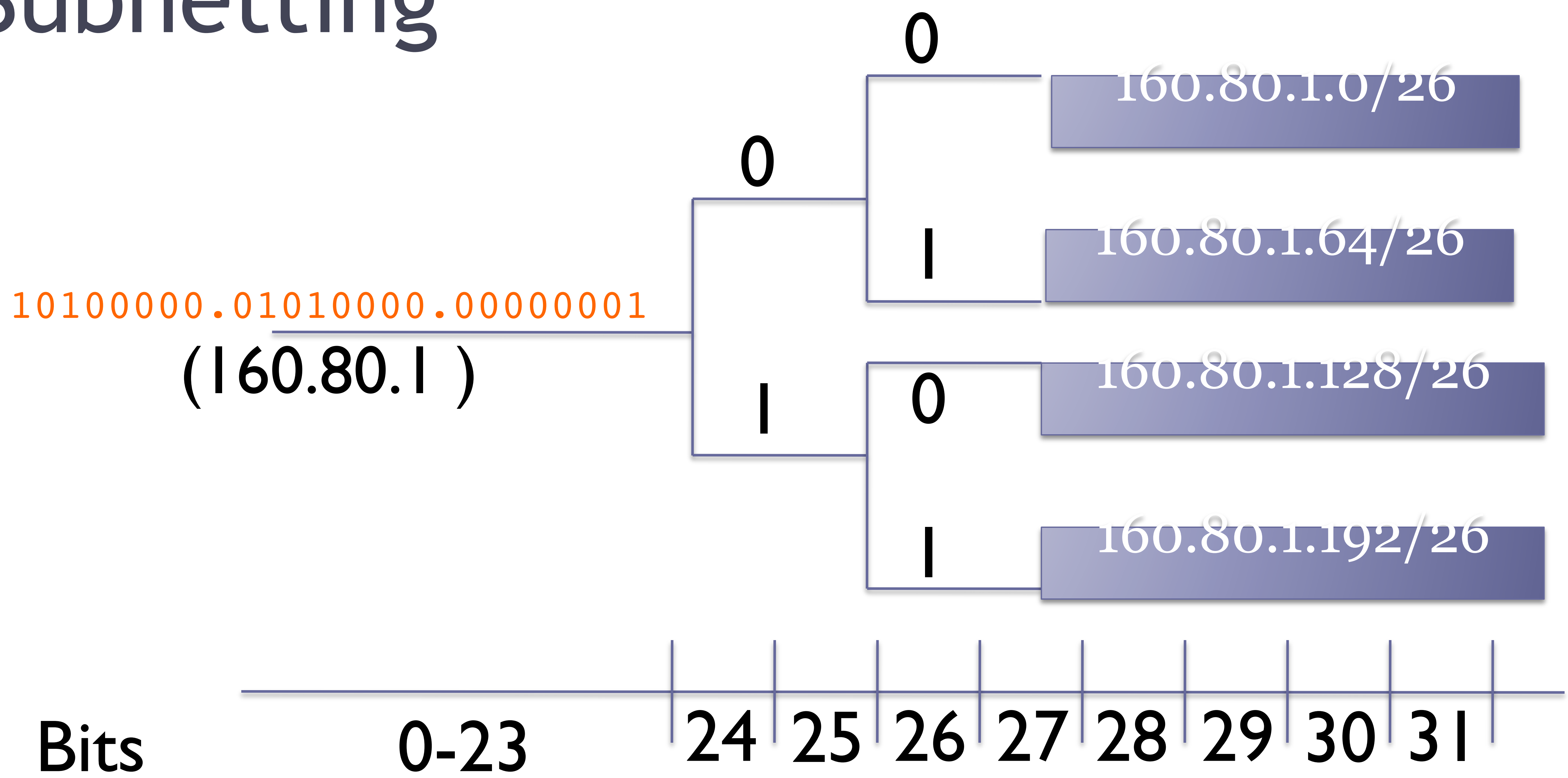
# Subnetting

- Correct subnetting form an addressing tree:




- When subnetting, we can leave some unused space

# Subnetting




It's clear that we can't combine 160.80.1.64/26 and 160.80.1.128/26 in a single /25
(even if they are consecutive)

# Exercise 3:

You have this range of ip address 10.3.4.0/24 :
Subnet the address space to have:

- One subnet /27
- Two subnet /30
- Three subnet /28

# Routers


D-Link
ADSL Router

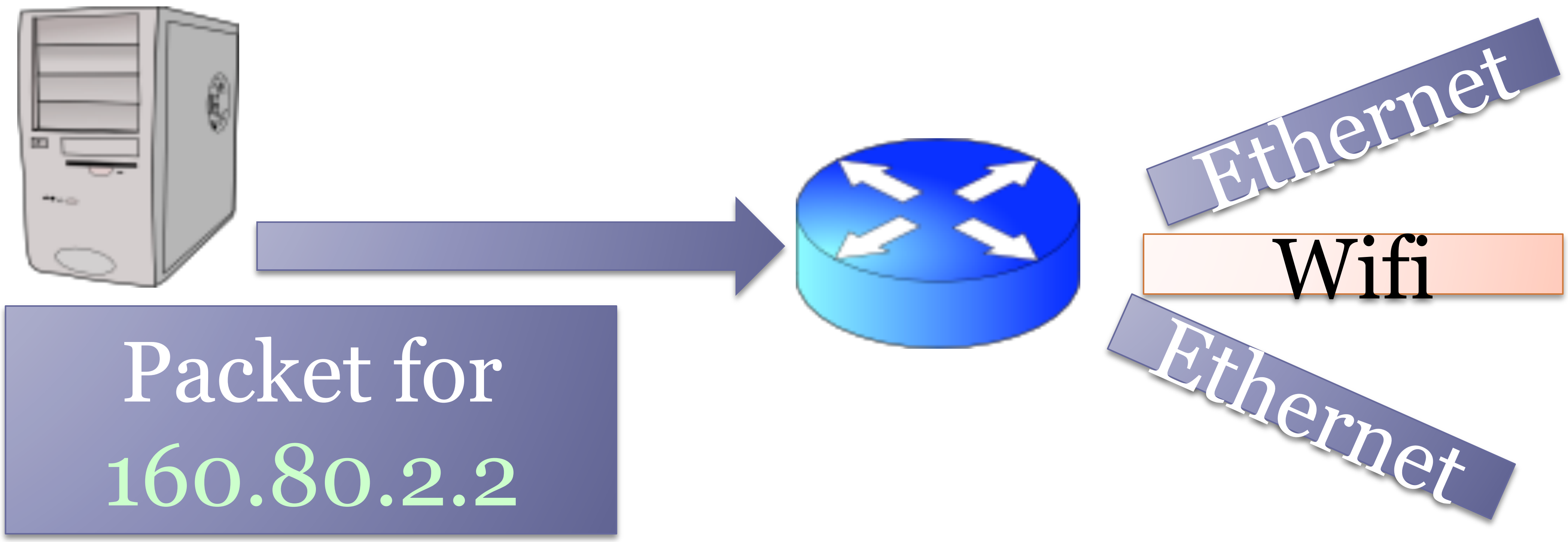

cisco
ROUTERS
Cisco 2851
Cisco 3825
Cisco 7603
Cisco 2821
Cisco 3845
Cisco 7604
Cisco 2811
Cisco 7201
Cisco 2801
Cisco 7204 VXR
Cisco 7606
Cisco 7206 VXR
Cisco 7609
Cisco 7613


Juniper
T320


LINKSYS

# Routers

- Routers have 1 interface (and one IP address!) for each network they connect.
- Basically they:
  - Receive IP packets and read the destination address
  - Lookup in a **Routing Table** and decide which is the next hop (another router or the final node)
  - Send the packet on right interface the the next hop

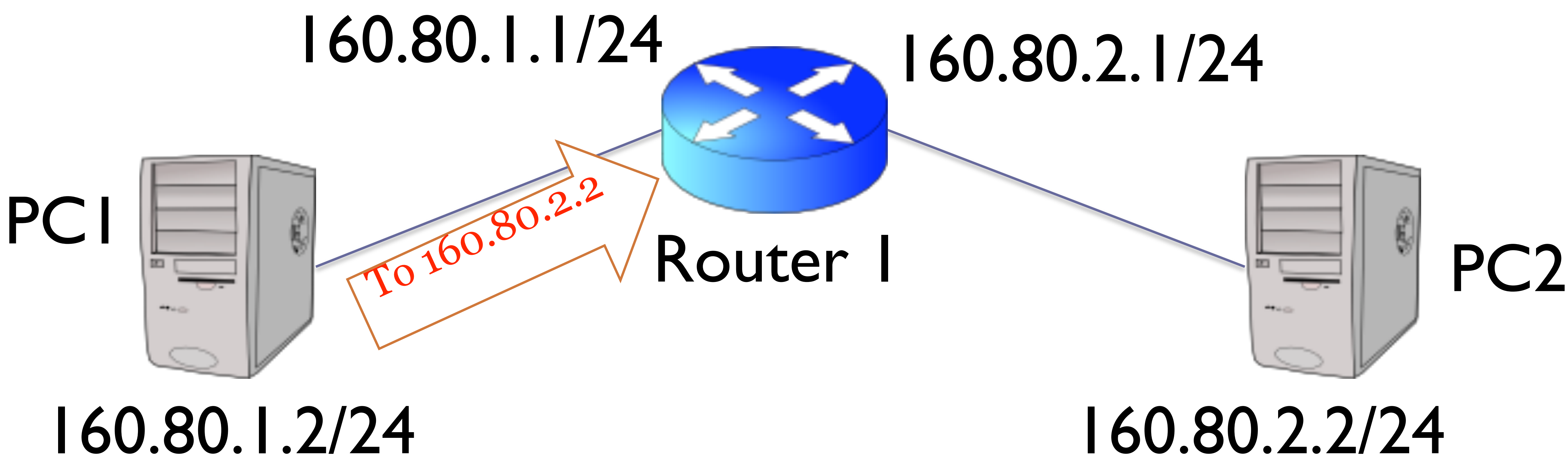


# Routers

| 0 | 4 | 8 | 16 | 19 | 31 |
|---|---|---|---|---|---|
| Version | IHL | Type of Service | Total Length | | |
| Identification | | | Flags | Fragment Offset | |
| Time To Live | | Protocol | Header Checksum | | |
| Source IP Address | | | | | |
| Destination IP Address | | | | | |
| Options | | | | Padding | |

- “basic” routers are interested basically only in destination address for what concern routing decision
- Not all routers are “basic”: different kind of routing are possible (e.g. based on source address!).
- We will always refer to “basic” routers.

# Router Behaviour




Routing Table Router 1

| Destination Address | Mask | Next HOP | Interface |
|---|---|---|---|
| 160.80.1.0 | 255.255.255.0 | local | eth0 |
| 160.80.2.0 | 255.255.255.0 | local | eth1 |

# Forwarding Decision (Router1) 1/2

| Destination Address | Mask | Next HOP | Interface |
|---|---|---|---|
| 160.80.1.0 | 255.255.255.0 | local | eth0 |
| 160.80.2.0 | 255.255.255.0 | local | eth1 |

| | | | | |
|---|---|---|---|---|
| 160.80.2.2 | 10100000 | 01010000 | 00000010 | 00000010 |
| AND 255.255.255.0 | 11111111 | 11111111 | 11111111 | 00000000 |
| = **160.80.2.0** | 10100000 | 01010000 | 00000010 | 00000000 |

160.80.2.0 is equal to Net address ? (160.80.1.0) → NO

Go to the next entry of the routing table

# Forwarding Decision (Router1) 2/2

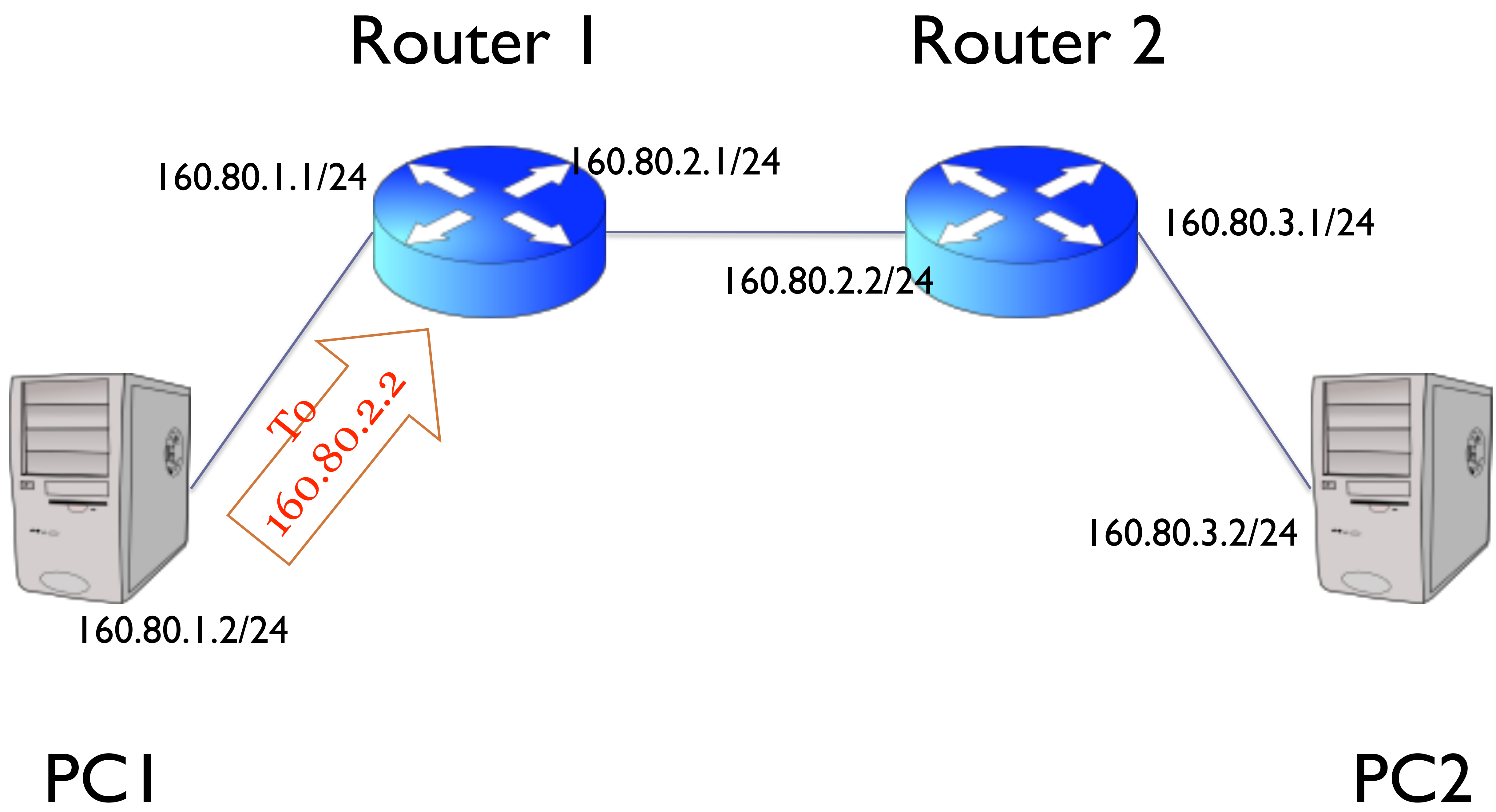
| Destination Address | Mask | Next HOP | Interface |
|---|---|---|---|
| 160.80.1.0 | 255.255.255.0 | local | eth0 |
| 160.80.2.0 | 255.255.255.0 | local | eth1 |

| | | | | |
|---|---|---|---|---|
| 160.80.2.2 | 10100000 | 01010000 | 00000010 | 00000010 |
| AND 255.255.255.0 | 11111111 | 11111111 | 11111111 | 00000000 |
| = 160.80.2.0 | 10100000 | 01010000 | 00000010 | 00000000 |

160.80.2.0 is equal to Net address ? (160.80.2.0) → YES

Forward to next hop (local forwarding) using interface eth1

# Scenario 2



Router 1
Router 2
160.80.1.1/24
160.80.2.1/24
160.80.2.2/24
160.80.3.1/24
To
160.80.2.2
160.80.3.2/24
160.80.1.2/24
PC1
PC2

# Forwarding Decision (Router1)

| Destination Address | Mask | Next HOP | Interface |
|---|---|---|---|
| 160.80.1.0 | 255.255.255.0 | local | eth0 |
| 160.80.2.0 | 255.255.255.0 | local | eth1 |
| 160.80.3.0 | 255.255.255.0 | 160.80.2.2 | eth1 |

If behind R2 there are 100 different subnets,
we need 100 other entries…

| 160.80.4.0 | 255.255.255.0 | 160.80.2.2 | eth1 |
|---|---|---|---|
| 160.80.5.0 | 255.255.255.0 | 160.80.2.2 | eth1 |
| 160.80.6.0 | 255.255.255.0 | 160.80.2.2 | eth1 |
| …. | … | 160.80.2.2 | eth1 |

Can we aggregate these entries?

# Forwarding Decision (Router1)

| Destination Address | Mask | Next HOP | Interface |
|---|---|---|---|
| 160.80.1.0 | 255.255.255.0 | local | eth0 |
| 160.80.2.0 | 255.255.255.0 | local | eth1 |
| 160.80.0.0 | 255.255.0.0 | 160.80.2.2 | eth1 |

**Both the entries are verified....what happens???**

Longest prefix match

entry with the longer netmask wins!

**This entry aggregates all the subnet from 160.80.3.0/24 to 160.80.255.0/24**

# Default gateway

- So this entry:

| Destination Address | Mask | Next HOP | Interface |
|---|---|---|---|
| 0.0.0.0 | 0.0.0.0 | 160.80.1.1 | eth0 |

  - is always verified
  - but used only when there are no other more specific route

# Exercise 4:

- Where to route (i.e. next hop) a packet directed to 160.80.50.4, according to the following routing table?

| Destination Address | Mask | Next HOP | Interface |
|---|---|---|---|
| 160.80.1.0 | 255.255.255.0 | local | eth0 |
| 160.80.2.0 | 255.255.255.0 | local | eth1 |
| 160.80.64.0 | 255.255.192.0 | 160.80.1.4 | eth0 |
| 160.80.0.0 | 255.255.192.0 | 160.80.1.4 | eth0 |
| 160.80.0.0 | 255.255.0.0 | 160.80.2.1 | eth1 |
| 0.0.0.0 | 0.0.0.0 | 160.80.2.1 | eth1 |

- Are all the entry necessary?

# Compact routing tables

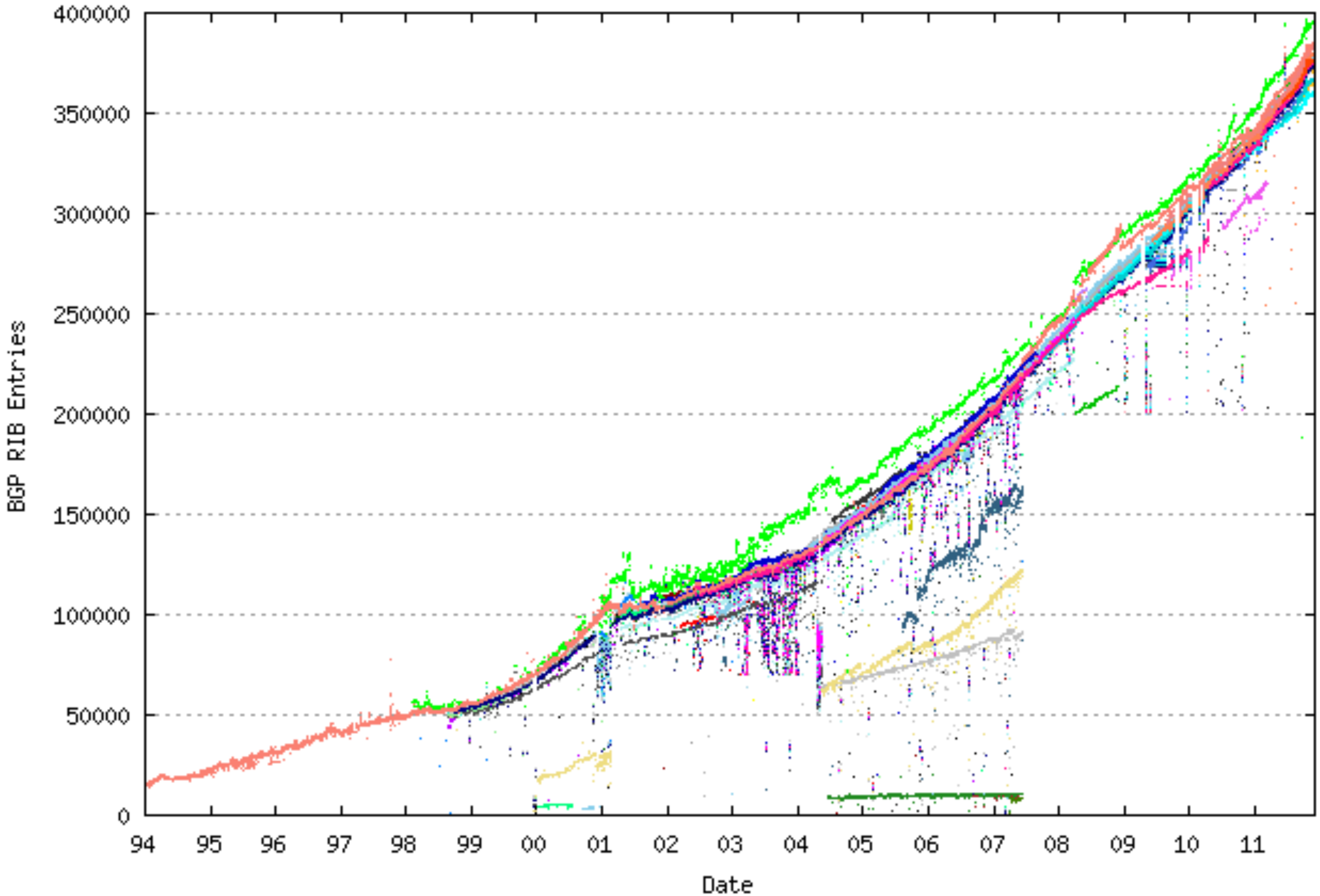
| Destination Address | Mask | Next HOP | Interface |
| --- | --- | --- | --- |
| 160.80.1.0 | 255.255.255.0 | local | eth0 |
| 160.80.2.0 | 255.255.255.0 | local | eth1 |
| 160.80.64.0 | 255.255.192.0 | 160.80.1.4 | eth0 |
| 160.80.0.0 | 255.255.192.0 | 160.80.1.4 | eth0 |
| 160.80.0.0 | 255.255.0.0 | 160.80.2.1 | eth1 |
| 0.0.0.0 | 0.0.0.0 | 160.80.2.1 | eth1 |

Useless!

(redundant with the default gateway entry)

Can be “zipped” in 160.80.0.0/17

# Growth of the BGP Table - 1994 to Present



# Exercise 5:

- Solve last exams (subnetting exercises)

# Managing routing tables, linux case

How to?... IProute2 !!! ☺
The command: **ip route <options>**

Show the routing table: ip route show

```
ip route show
10.0.0.0/30 dev sl0  proto kernel  scope link  src 10.0.0.1
192.168.100.0/24 dev eth0  proto kernel  scope link  src
192.168.100.41
160.80.0.0/16 dev eth1  proto kernel  scope link  src 160.80.103.105
default via 160.80.80.1 dev eth1
```

Enabling packet forwarding in linux kernel (enabled in netkit):

# echo 1 > /proc/sys/net/ipv4/ip_forward

# Setting up static routes

- Local forwarding: routes automatically recorded in the routing table at the address configuration of a network interface…why?
- Static routes:

```
# ip route add ADDRESS/MASK via NEXT_HOP
```

- Example:

```
# ip route add 192.168.1.0/24 via 10.0.0.1
```

# IPROUTE2 Summary

| Comando | Significato |
|---|---|
| ip link set dev eth1 up | Abilita l'interfaccia eth1 |
| ip address add 192.168.1.1/24 dev eth1 | Assegna l'indirizzo 192.168.1.1 all'interfaccia di rete eth1. Imposta una regola della tabella di routing per il forwarding in locale dei pacchetti destinati ad host appartenent alla rete 192.168.1.0/24 |
| ip route add 10.0.0.0/24 via 192.168.1.1 | Imposta una rotta verso la sottorete 10.0.0.0/24 |
| ip link show | Visualizza lo stato di tutte le interfacce |
| ip address show | Visualizza gli indirizzi associati alle varie interfacce della macchina |
| ip route show | Visualizza la tabella di routing |