Net_practice

In this project we learn the basics of networking, learning what are the main technologies, protocols and devices that go into making networks operational and stuff..

The actual project is solving a series of exercises where we must connect different devices to one another over a network, solving the problems isn't that hard and we can always ask for help but really understanding why a given solution works and even more important knowing all the possible solution and all scenarios where it wouldn't work that is, to me, the real project.

Network_devices

Switch - allows us to make networks

Is networking hardware that connects devices on a computer network by using packet switching to receive and forward data to the destination device. A network switch is a multi-port network bridge that uses MAC addresses to forward data at the data link layer (layer 2) of the OSI model. Some switches can also forward data at the network layer (layer 3) by additionally incorporating routing functionality. Such switches are commonly known as layer-3 switches or multilayer switches.

Router – routes stuff

A router is a computer and networking device that forwards data packets between computer networks, including inter-networks such as the global Internet.

Routers perform the "traffic directing" functions on the Internet. A router is connected to two or more data lines from different IP networks. When a data packet comes in on a line, the router reads the network address information in the packet header to determine the ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey. Data packets are forwarded from one router to another through an inter-network until it reaches its destination node.

TCP/IP

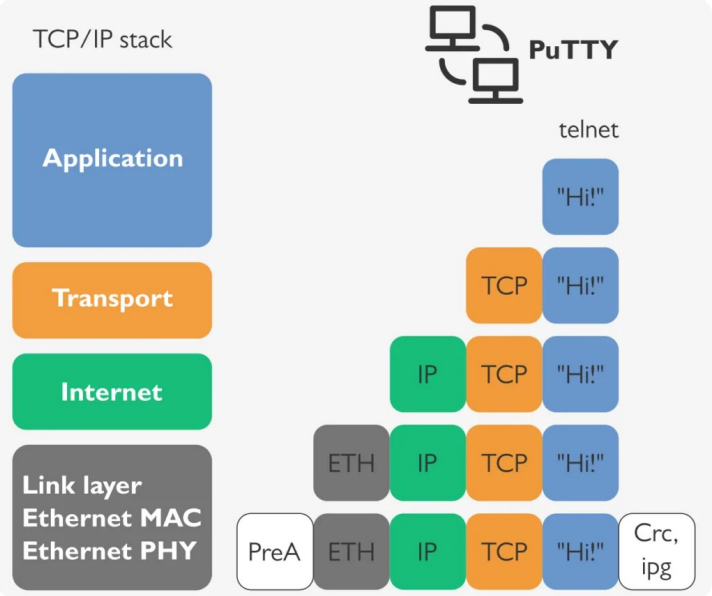
TCP/IP is a set of rules (protocols) that allows computers to communicate with each other over a network. It’s like the "language" devices use to send and receive data reliably and efficiently.

TCP (Transmission Control Protocol): Handles breaking data into packets, sending them, and ensuring they arrive in the right order and without errors.

IP (Internet Protocol): Handles addressing and routing the packets to the correct destination.

Together, they make sure your data (like a webpage or an email) gets from point A to point B.

How Does TCP/IP Work?



Think of TCP/IP like sending a letter through the mail:

TCP is like tearing the letter into pieces (packets), numbering them, and ensuring they’re reassembled correctly at the destination.

IP is like the postal service figuring out the address and delivering the pieces to the right mailbox.

Layers of TCP/IP: TCP/IP is often explained using a 4-layer model:

- **Application Layer:** Where apps like browsers or email clients live (e.g., HTTP, FTP).
- **Transport Layer:** TCP (reliable) or UDP (faster, less reliable) manage data delivery.
- **Internet Layer:** IP handles addressing and routing.
- **Link Layer:** Physical connection (e.g., Ethernet, Wi-Fi).

Handshakes:

TCP uses a "three-way handshake" to start a connection:

Sender says, "Hey, let's connect!" (SYN).

Receiver says, "Cool, I'm in!" (SYN-ACK).

Sender confirms, "Great, we're connected!" (ACK).

Why It Matters

Reliability: TCP ensures no data is not lost or corrupted (great for emails or file downloads).

Flexibility: IP works with any network type (Wi-Fi, Ethernet, etc.).

Scalability: It powers everything from your home router to the global internet.

Here's the step-by-step:

1. Your computer breaks data into small chunks called packets.

2. Each packet gets a header with info like the source IP address (your device), destination IP address (the receiver), and sequence number.

3. The packets are sent across the network, possibly taking different routes based on network conditions and routing decisions.

4. If a packet doesn't reach its destination due to issues like network congestion, hardware failure, or errors, the sender (via TCP) doesn't immediately know. The destination or intermediate devices may drop the packet, and no acknowledgment is sent back.

5. TCP at the destination waits for all packets and checks for missing or corrupted ones using sequence numbers and checksums. If a packet is missing, the destination sends a negative acknowledgment (NACK) or fails to send an acknowledgment (ACK) for that packet.

6. Upon detecting a missing packet (no ACK or NACK received), the sender retransmits the lost packet. This process may involve a timeout mechanism where the sender waits a certain period before resending.

7. Once all packets arrive, TCP at the destination puts them in the correct order using sequence numbers and reassembles the original data.

8. If packets are still missing after several retransmission attempts, TCP may notify the application of a failure, and the connection might be terminated or the data retransmission might be attempted again, depending on the protocol and application requirements.

9. Finally, the reassembled data is delivered to the receiving application, ensuring reliability and order.

IP Addresses:

Every device on a network has a unique address, like a phone number. For example: 192.168.1.1.

Two versions exist:

> **IPv4:** Older, uses 32 bits (e.g., 192.168.0.1), limited to ~4.3 billion addresses (the ones we use in net-pratice).
>
> **IPv6:** Newer, uses 128 bits (e.g., 2001:0db8::8a2e:0370:7334), supports way more devices.

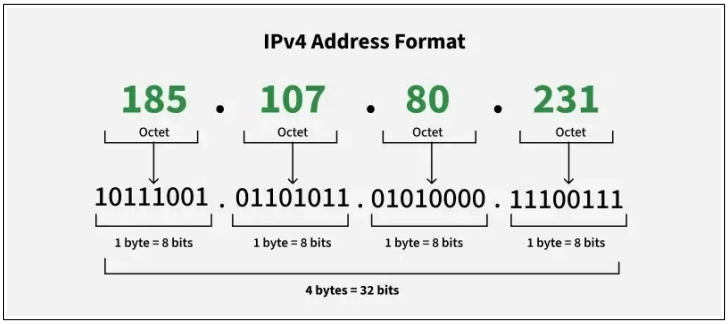
Ports:

Ports are like apartment numbers at the IP address "building." They tell the device which application (e.g., web browser, email) the data is for.

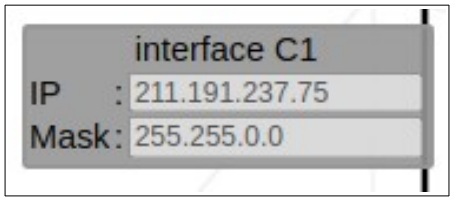
Examples: Port 80 (HTTP for websites), Port 443 (HTTPS), Port 25 (SMTP for email).

OK so how to solve the problems!😤

The first lvls are pretty simple we just have a few devices and maybe a switch, so just a bunch of mostly direct connections that need to be enabled we just need to adjust the IP addresses and sub-net masks of the devices in the network.

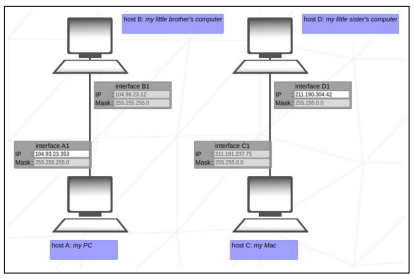


IP - its the 4 number combination where each number is [0-255] meaning it's 4 hexadecimal numbers that identify a device in a network.



sub-net mask - A sub-net mask is defined as a 32-bit address that segregates an IP address into network bits that identify the network and

host bits that identify the host device operating on that network.



So the sub-net mask tells us how many of the bits represent the network and how many represent the host.(ps: I put zeros before so the numbers align and it also works in the net-pratice program)

Eg:

192.168.026.042 = 1100 0000.1010 1000.0001 1010.0010 1010

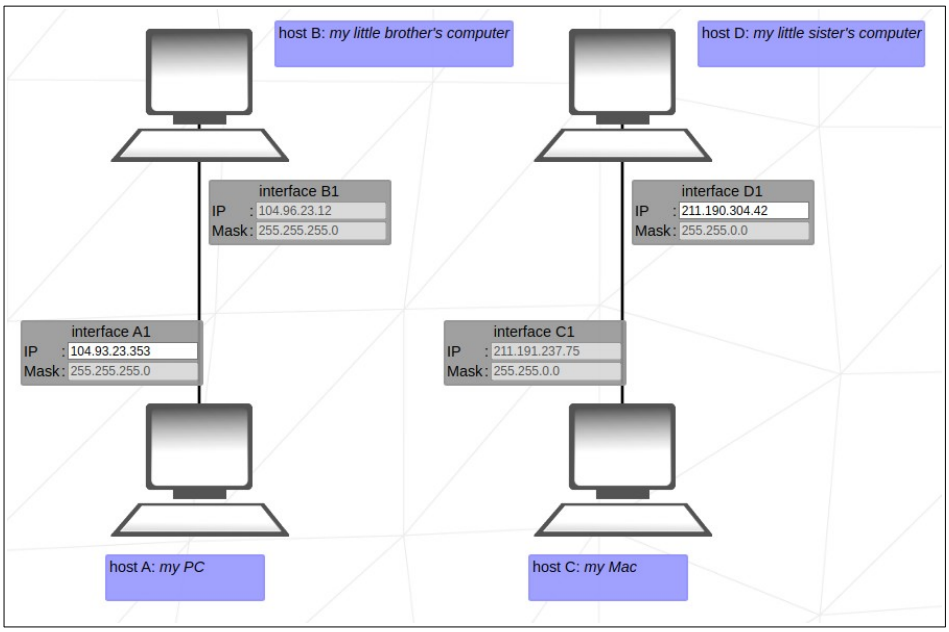
255.255.255.000 = 1111 1111.1111 1111.1111 1111.0000 0000

Sub-net Mask	Binary Representation	Network Bits	Host Bits
000.000.000.000	0000 0000.0000 0000.0000 0000.0000 0000	0	32
255.000.000.000	1111 1111.0000 0000.0000 0000.0000 0000	8	24
255.255.000.000	1111 1111.1111 1111.0000 0000.0000 0000	16	16
255.255.255.000	1111 1111.1111 1111.1111 1111.0000 0000	24	8
255.255.255.255	1111 1111.1111 1111.1111 1111.1111 1111	32	0

Class	Sub-net Mask	# of Usable Hosts
	000.000.000.000	4,294,967,294
A	255.000.000.000	16,777,214
B	255.255.000.000	65,534
C	255.255.255.000	254
D	255.255.255.255	1

LVL 01

The answer lies in the heart of the network



In this one we have to connect two pairs of pcs, just a simple direct connection.

a) for the lil bro pc we have the standard 255.255.255.0 (class c) on both so its pretty simple, since the masks are fixed we need to copy the network part meaning the first 3 octets and then in the IP we can assign any from 1 to 254 so long as its different from the other one different 192.168.000.X onde (1 <= x <= 255 && x != 12).

b) For lil sis both sub-net masks are also fixed and only one IP to edit, in this case since the IP takes two octets it means that we can use any device address from (211.191.0.0) to (211.191.255.255) except those two cuz one they are network and broadcast respectivly, but we can use (0.255), (1.0) (1.255), (255.0),(255.254) and etc.

Subnet mask classes:

We can devide the basic subnet-maks into 4 classes depending on how many octets are used for the network vs host:
Basically we just assign each octect a letter and the last octect used is the class.
We don't have a class for 0.0.0.0 because this is a special case.

This is important but not very necessary por the pratical part of this project.

Class	IP Address	Net ID	Host ID
A	a.b.c.d	a	b.c.d
B	a.b.c.d	a.b	c.d
C	a.b.c.d	a.b.c	d

Eg: 255.255.000.000 (class B)
255.000.000.000 (class A)

LVL 02

Similar to lvl 01 but a little trickier, we can solve it easy but to find all solutions its not so easy.

In comp B the fixed mask's first non 1 filled octect is the last octect, 1110 0000 = 224, we have 5 bits for host so 2^5 = **32** (the number of IP adresses per subnet) and if we devide this by the 256/32 = 8 we get the number of subnetworks the first one starting from [0 - 31] and so on.

To quicly find the start of the subnet just need to make 0 all the host ips (IP & mask):

note: ony the octect where they share bits matters

Step = 32

IP 192.168.148.222 => 1101 1110

mask 255.255.255.224 => 1110 0000

network **192.168.148.192 => 1100 0000**

This means our subnet is [192-223] so we can use any IP from 193 to 221 except 222 because its being used by interface B1.

Ranges

[0 - 31]
[32 - 63]
[64 - 95]
[96 - 127]
[128 - 159]
[160 - 191]
[192 - 223]
[224 - 255]

For the second part we have a CIDR /30 (255.255.255.252) this means we have 30 bits for network meaning we have (8 + 8 + 8 + 6) or minus 2 bits from the 4th octet 1111 1100.

2^2 = 4 ips per sub-net then zero all the host bits to find the sub-net. Como nao temos ip fixo podemos usar qualquer IP valido, eu escolhi 42.42.42.42

Step 4

IP 42. 42. 42. 42 => 0010 1010

mask 255.255.255.252 => 1111 1100

result => **0010 1000** = **[40 - 43]**

Entao podemos usar apenas o 41 e o 42, Usando o escolhido IP mas com qualquer outro o processo e o mesmo, recomendo que teste vc mesmo com outos para verificar.

Sub-net ranges and usable ips

00000000 → 0
10000000 → 128
11000000 → 192
11100000 → 224
11110000 → 240
11111000 → 248
11111100 → 252
11111110 → 254
11111111 → 255

This part is **very important** for this project because much of it is resumed to connecting devices by matching networks and sub-networks

For two devices to be connected they need to have:

1. Physical connection via cable or wireless.
2. Both need to have and IP address in the same network and sub-net.

So we know that the mask tell us how many bits of the IP address are used for the network but that isnt always full octects and when we have partially filled octects the network can be divided into smaller sub-networks like:

192.168.224.042 = 1100 0000.1010 1000.1110 0000.0010 1010

1110 0000 = 224 this means that the first two bits of this octet are used for the network and the rest for the host, thus there are only 8 variations (0/128/192/224/240/248/252/254/255) since these need to be **contiguous** and each using progressively more bits for the network.

The way the subdivision works is by powers of two specifically the 2^(num of host bits) so 1110 0000 has 5 host bits 2^5 = 32 **this is the number of ips of each sub-net** then we divide 256 / 32 = 8 **this is the number of sub-nets** or intervals and the first IP of each sub-net is used for the **network** and the last one for the **broadcast** so in this case we have.

256 / 32 = 8

ID	Range	**Usable**
[1]	[000 - 031]	**[001 - 030]**
[2]	[032 - 063]	**[033 - 062]**
...		
[7]	[197 - 229]	**[198 - 228]**
[8]	[230 - 255]	**[231 - 254]**

Note that the jump is always 32 so the start of a range n is:

(n - 1) * 32

CIDR	Sub-net Mask	Nº of Sub-nets	Nº of Usable
/0	000.000.000.000	1	4,294,967,294
/1	128.000.000.000	2	2,147,483,646
/2	192.000.000.000	4	1,073,741,822
/3	224.000.000.000	8	536,870,910
/4	240.000.000.000	16	268,435,454
/5	248.000.000.000	32	134,217,726
/6	252.000.000.000	64	67,108,862
/7	254.000.000.000	128	33,554,430
/8	255.000.000.000	256	16,777,214
/9	255.128.000.000	512	8,388,606
/10	255.192.000.000	1,024	4,194,302
/11	255.224.000.000	2,048	2,097,150
/12	255.240.000.000	4,096	1,048,574
/13	255.248.000.000	8,192	524,286
/14	255.252.000.000	16,384	262,142
/15	255.254.000.000	32,768	131,070
/16	255.255.000.000	65,536	65,534
/17	255.255.128.000	131,072	32,766
/18	255.255.192.000	262,144	16,382
/19	255.255.224.000	524,288	8,190
...	...	...	...
/30	255.255.255.252	268,435,456	2
/31	255.255.255.254	536,870,912	0
/32	255.255.255.255	4,294,967,296	1

CIDR - Classless Inter-Domain Routing

When we have partially filled bits we need to use another way to represent the subnet-maks, thats where CIDR comes in, this slash then a number (/24).

CIDR (Classless Inter-Domain Routing) is a flexible system for specifying IP address ranges using a prefix length (e.g., `/24`), replacing rigid class-based networks (A, B, C). It’s written as an IP and slash number, like `192.168.1.0/24`, where the number indicates how many bits are for the network (e.g., `/24` = 24 network bits, 8 host bits).

- **Efficiently allocates IP addresses**, avoiding waste (e.g., `/30` for 2 devices, not a whole `/24`).
- **Reduces routing table size** by allowing route aggregation (e.g., grouping /24s into a /22).
- **Offers flexibility** to create sub-nets of any power-of-2 size (4, 8, 16, etc.).
- **Maximizes IPv4 use**, crucial as addresses became scarce.
- **Simplifies network design and scaling** for modern setups like cloud or home networks.

Converting Sub-net Mask to CIDR (/X)

A sub-net mask is a 32-bit number written in dotted-decimal format e.g., 255.255.255.000).

Each 255 means all 8 bits in that octet are network portion, and each 0 means all 8 bits are host portion but when we have partially filled octects we don’t get a 0 or 255 but something in between.

The CIDR number (/X) is the total number of 1 bits in the mask.

When in subnet-mask format the decimals are the converted binary octect, 1000 0000 = 128 as simple as that.

The solutions

Rule 1: same network but different mask cant connect!

LVL 3 - this one is also simple , we are introduced to switches in this one we have one fixed mask and one fixed IP, first we juts make the masks all match and then find the subnet using the one fixed IP.

Now to find our sub-net its the same process as before , just a logic and:

Step 2 ^ (7) = 128

IP 104.198.134.125 => 0111 1101

mask 255.255.255.128 => 1000 0000

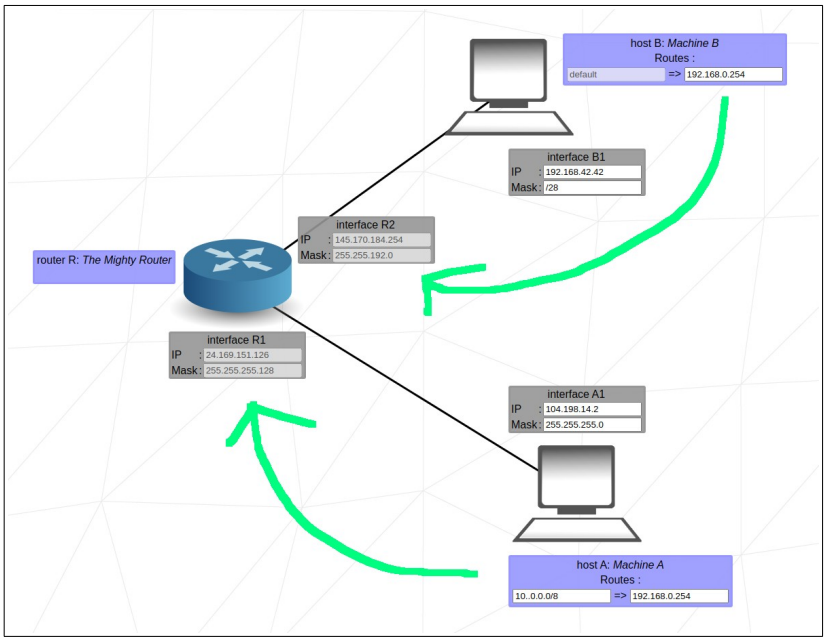
result **=> 0000 0000 = [0 - 127]**

We can use any valid ip in that range.

The key difference in this lvl is that the swith allows us to have multiple devices in this direct connection setup, if we wanted we could add 123 devices since in our current sub-net there are 128 total usable Ips minus the network, the broadcast and the 3 devices already there.

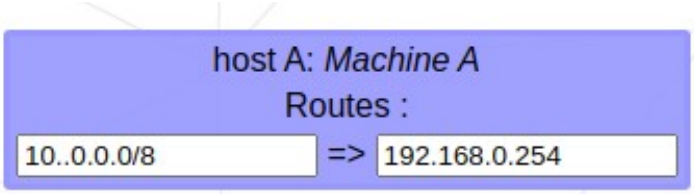
LVL 4 - this time we are introduced to a router, the first part is just two pcs and a switch same as before just match the masks find the range and start, lets use the one already there, its 240 (1111 0000) that means we have 4 bits for the host 2^4 = 16 ips per range and the start is 128 so subnet is **[128 - 143]**, now just match the mask and subnet with the other devices and we are set.

In this lvl the interfaces in the other side of the router are fixed so we don't need to worry about them we here learn that a router allows us to connect devices over different networks cuz even though we used different CIDR and our devices are in different sub-nets we can still connect to interfaces R# and R2 over the router.



LVL 5 – Now we start to use routing tables, these let us tell the router where to send packets b

If it matches the destination IP/CIDR send to net jump this is what these boxes are for, the left side is the destination the device the packet is meant for and the right side is the place it will be sent to if said destination isn’s reachable like say its not in our network.



a) First just like in previous lvls we just need to match the masks and ips of the devices/interfaces in each network so they can communicate.

b) Now since machine B and machine A are in diff networks they need to communicate via the router so lets route the all our traffic from A to its router interface and the same for B. **To route all traffic we use the IP/CIDR (0.0.0.0/0 or default)** so in both networks its just use:

default => router inteface.

Routing

Routing is the process of selecting the best path for data packets to travel across a network from their source to their destination.

In the these lvls we have these new windows that say x routes: y basically ur saying that packets meant for x should go to y, in case u want to say all packets u should use default = 0.0.0.0/0

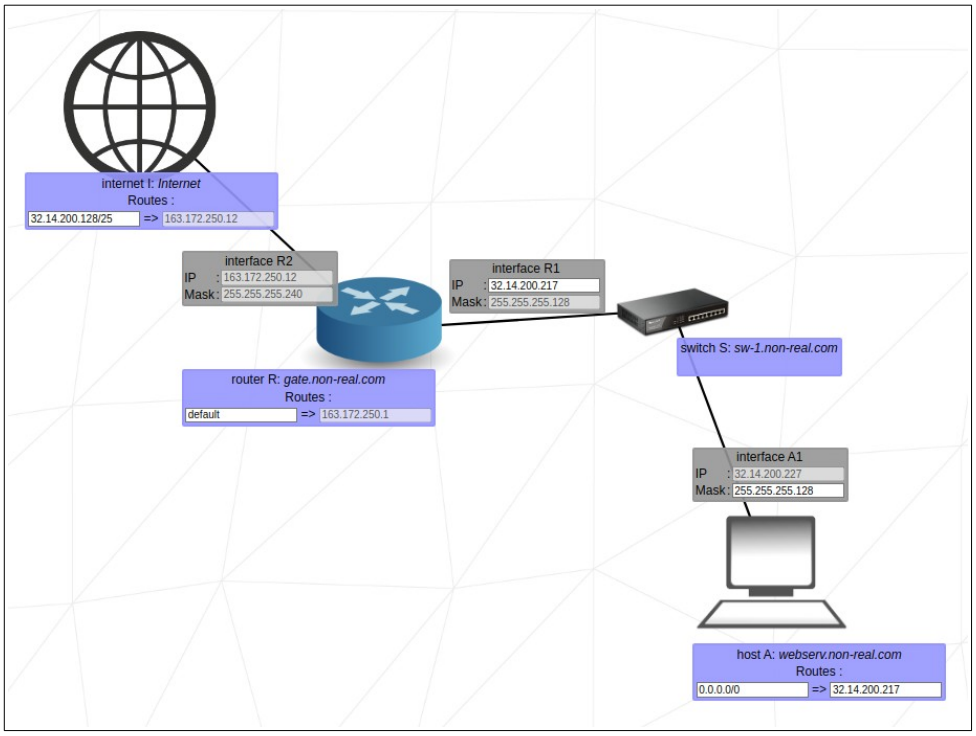
In lvl 5 we just need to say for both pcs that all their traffic should be routed to the router interface that is next to them, thus A routes default => 145.170.184.254 (R2) and B routes default => 24.169.151.126 (R1) this solves the problem now A and B can communicate.

Additionally we could say that:

A routes default 192.168.42.42 => 24.169.151.126 (R1)

B routes default 104.198.14.2 => 145.170.184.254 (R2)

This way we specify that only packets meant for A from B should go to the router and that only packets meant for B from A should go to router, we don’t have to say its all traffic like in the first solution we can specify a single IP, a small or large range of ips.



LVL 6 - Now we have the internet which acts just like a network.

So first we route packets from A1 to R1, then we have two routing tables, the one under the router I was told is to connect to another router across the internet, that's why it has that fixed IP that is nowhere in the problem its an IP that is part of some router across the web, so we just route default to it, meaning all traffic.

In the routing table right next to the internet we are supposed to route all traffic that is going to the IP of our home network (32.14.200.128/25) to R2 and setting it this way does indeed work just set in the internets routing tabel 32.14.200.128/25 => R2. One important thing is that we can vary the CIDR we use in here and it will still work this is because the IP/CIDR only needs to point to an network or subnet that contains the destination.

Lets say we have and ip 192.168.1.1/24 this IP can be specified in a routing table just like this but if we use /23 this will also be acceptable since it still holds in its subnet the unique binary ip we are looking for.

IP 192.168.1.128 → 1100 0000.1010 1000.0000 0001.1000 0001

CIDR /24 → 1111 1111.1111 1111.1111 1111.0000 0000

CIDR /23 → 1111 1111.1111 1111.1111 1110.0000 0000

the gist is

Say we have an IP somewhere from x.x.x.0 and x.x.x.16 if like x.x.x.9 we have many ways to refer to this IP when routing.

The first one is to just use the ip and its CIDR like x.x.x.9/30 or the network IP and the same CIDR.

The second one is using an IP and a CIDR that includes this ip like:

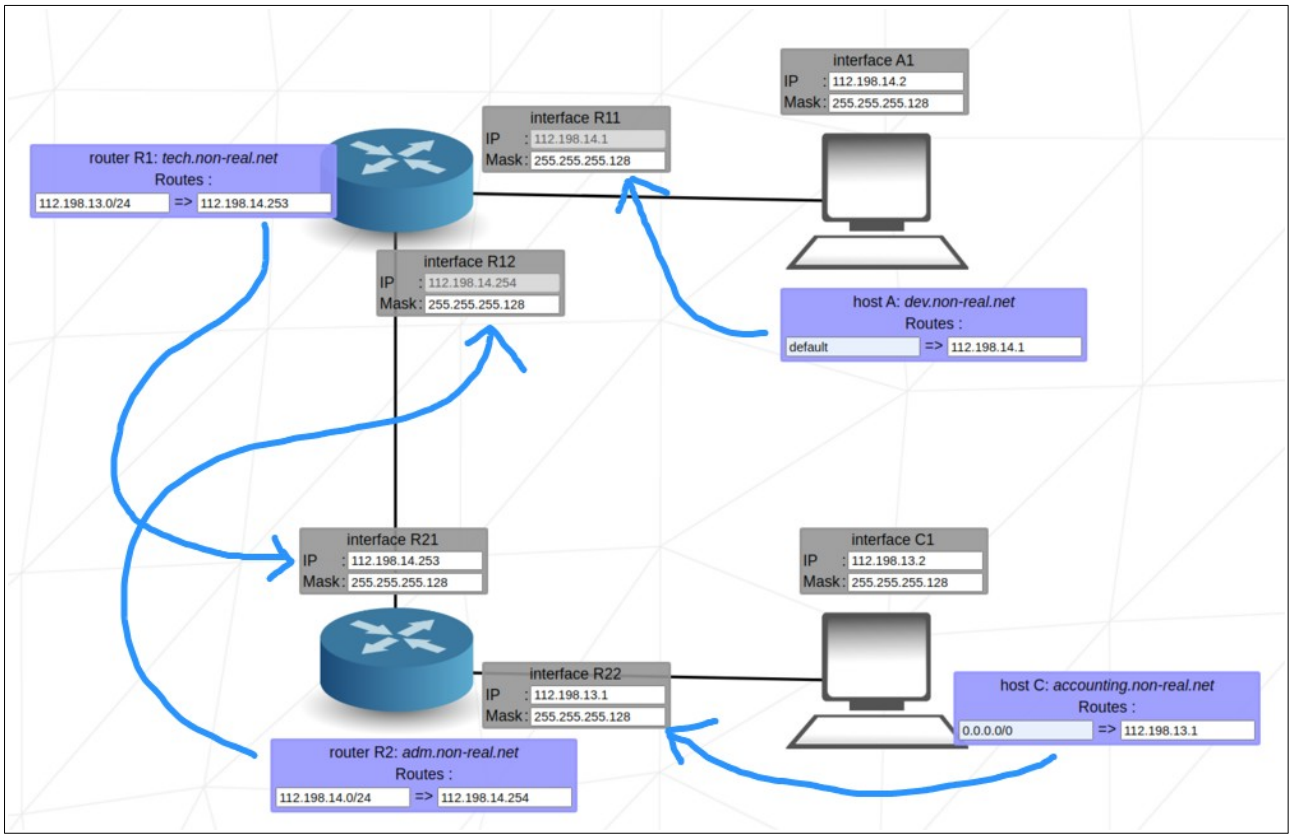
x.x.x.9/29

x.x.x.9/28

In fact so long as we use a CIDR lower then the original one it should work.

LVL 7 - This lvl is similar to the last one, in this case we have two pcs and two router this is essentially how routing is supposed to work at its most basic, it allows us to connects across diff networks.

THIS WAS THE MOST FRUSTRATING ONE YET!! 😫😭😣



So the solution is kinda simple I just got stuck on it for a while until someone helped me out.

> 1 - Like in the previous ones we need to make the neighboring interfaces able to connect to one another by matching the masks and IP ranges for each section in this case we have 3.
>
> 2 - We then set the path forwarding rules, A forwards to its router's interface r1/r11 then in the router we set a forward to the neighboring router's interface r2/r21 then we do the inverse, C forwards to r2/r22 and r2 forwards to r1/r12.

Setting things up like this does work, kind of... it has a major problem, if the IP ranges of the different IPs in one of the 3 networks match then we will have a kind of conflict, if our sender network matches the params set for the receiving network then we will get a loop where the sender is receiving the package it sent over and over, in a loop.

To fix this we need to make it so the sender and receiver networks cant match, we achieve this by using a mask that will divide the IP ranges into intervals that our ips don't overlap. Actually not only should the

sender and receiver networks not match but the same for the middle network the one in between the router we need to separate the 3.

So in this case we have two fixed ips one

in net-A and the other in the routers

network, these are 112.198.14.1 and 112.198.14.254 that means that no matter what one is the very first

IP available for the given mask and the other the very last

now we just need to use any

number of division above 2, so a min of

3, with 3 division or ranges

we can separate our 3 nets.

So at min our mask (use the same in all

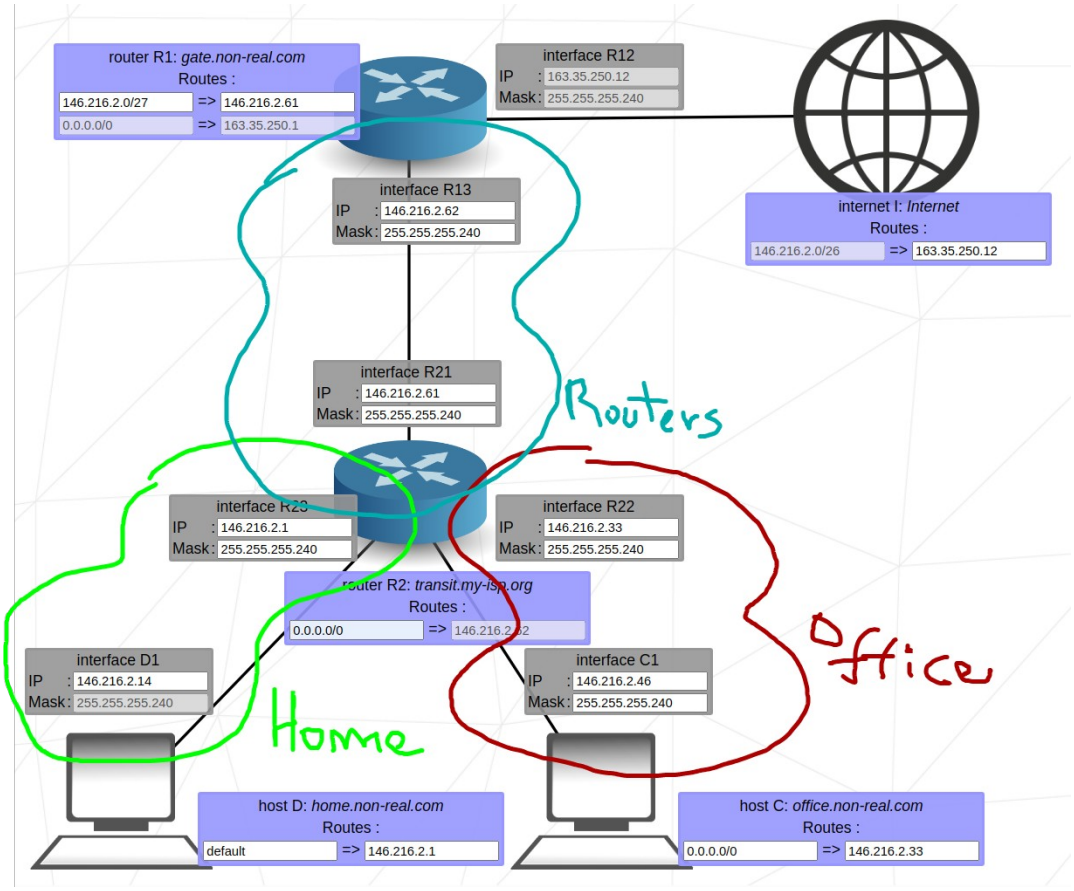
places, for simplicity)needs to be 255.255.255.192

Why this is the min mask ?

So we want to have 3 or more ranges in order to separate our 3 networks, than means we need 255 / 3 = 85, so our number of ips per range needs to be 85 or smaller, What number of bits gets us this ? That would be 2^x = 85 so log2(85) = 6.4 and 2^6 = 64 so our ranges are:

[0-63] [64-127][128-191][192-255]
[net_A] [routers]

So now we just use one of the other ranges for net_c and we are done, we can always use smaller ranges it works as well, the important thing is to separate the ranges for each net.



LVL 8 - in this one we first get an easy one, we just need to connect two pcs in two different nets over a route.

The steps are the same as before:

1. use the same mask in all hosts in the same network. Actually in this case just go ahead and use the same mask everywhere for simplicity.
2. Use diff network parts of the IP for each network.
3. Route trough the router interface in the network.

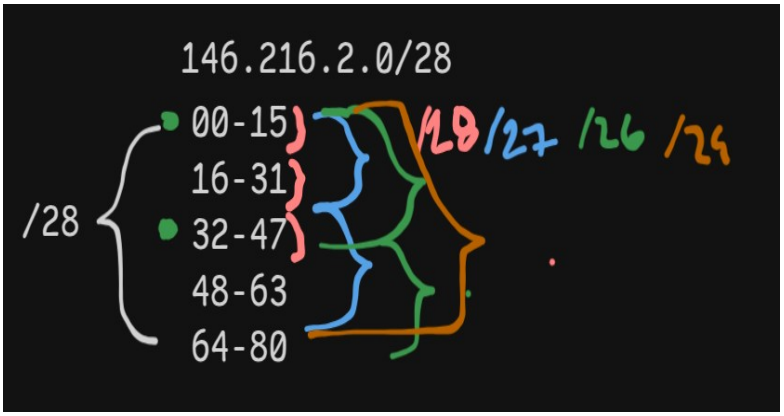
As per specific ips and masks since one of them already has a mask lets just use that for the others as well so now all pcs in both home and office sub-nets have the same mask /28 now for the ips we can use virtually any valid IP ranges given the masks. This last part is not quite true for the entire problem but try anyway it will work for objective 1.

0.0.0.0/0Now for obj 2 and 3 we know that the internet is trying to reach 146.216.2.0/26 meaning this needs to be the address of our final networks (from internet to pc), thus we must change the ips of both the home and office networks to match this, but we cannot use just any ips.

We know our home and office networks must start from 146.216.2.0 and since we have the mask 255.255.255.240 we have 16 intervals of 16 ips each and we must choose one for each network but since the CIDR that the internet is using is /26 starting from 146.216.2.0 (first range) that means we can only choose from [0 - 64] since this the first range so:

[00 15] [16 31]

[32 47] [48 63]



Lets use range [00 15] for home and [32 47] for office, test and obj 1 should still work, then we set the interfaces between the routers since the routing table of R2 is fixed we need to assign routing its next-jump (146.216.2.62) IP to the interface R13 of R1 and then we choose for R21 of R2 a valid IP that is in the same range, in this case [48 63].

Next we set the routing table for the internet (internet I) to pass trough the only interface in the same network: 163.35.250.12.

Finally we set the R1 to route packets meant for 146.216.2.0/28 (yes use /28 not /26) to the interface R21 of R2 with IP 146.216.2.49.

So now it should say that both obj 1 and obj 3 passed but that there is no reverse way in obj 2 this is because we used 146.216.2.0/28 which means that we are looking for the receiver in the 16 ips of the range 146.216.2.[0-15] which only matches the home network and not the office to fix this we need CIDR that includes both the home and office networks.

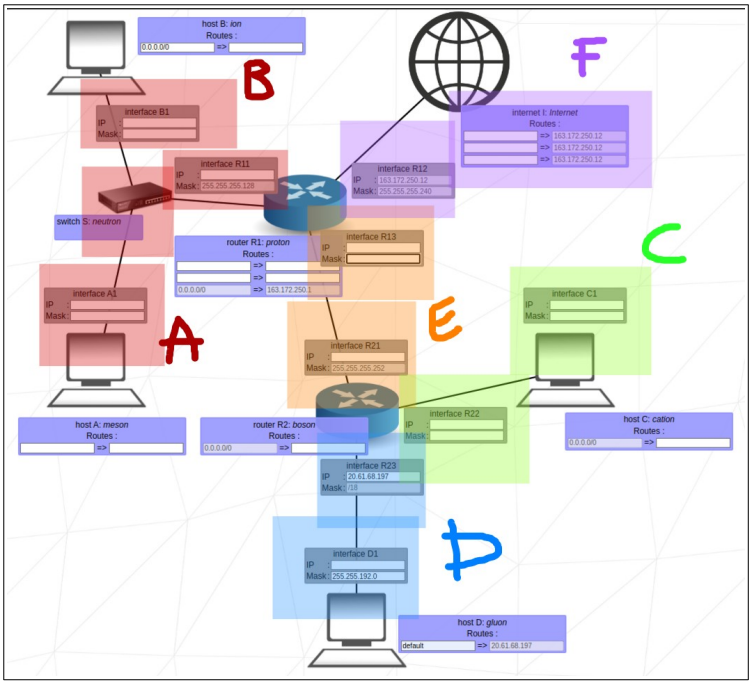
Since home uses the range [00 15] and office [32 47] that means we need a range of 48 ips tp cover both, so 256 / 48 = 5.3(round up) -> 6 which means that our mask needs 6 bits for the host which means our CIDR = 32 - 6 = 26 so our CIDR need to be /26, this way our first range goes from [0 - 47] encompassing both [00 15] and [32 47]. The reason why we can use 5 is cuz 2^5 is only 32 ips which isnt enough since we need 48 ips at least.

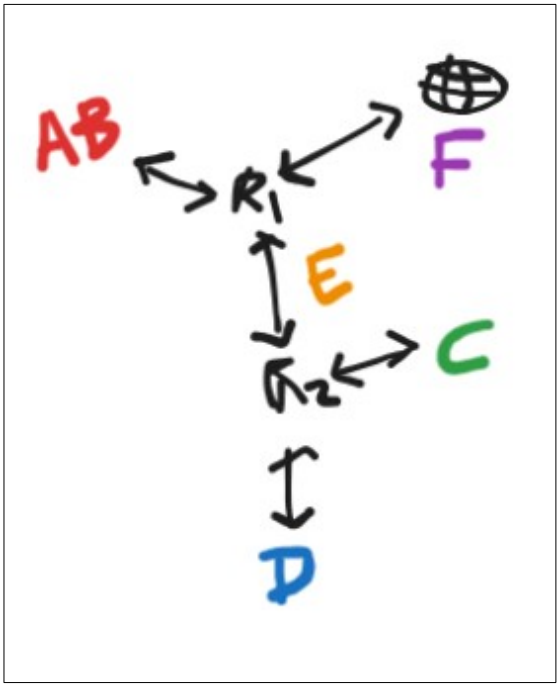
With this we have the solutions the other option is to use the first two usable ranges instead of the first and the third thus we could have used for home and office [00 15] and [16 31] and since they make up a

smaller range of just 32 ips and not 48 like the previous one we can use the CIDR /27.

BONUS_INFO

In case we want to use any other two groups we just need to make sure they are part of the range defined by the CIDR we used in **router R1: gate.non-real.com** but we are also limited by the fixed CIDR in **internet I: Internet** since its fixed at /26 this means that we can only effectively use a CIDR that is bigger or equal, if we try o use one that is smaller that would mean that we are tryng to match with ranges that go outside of the first range the fixed range and so it wont work even if all other aspects are valid.





LVL 9 (Boss fight) - this is the last problem where we need to use all the knowledge we obtained in all previous levels.

Goal-1: for obj 1 we need to connect two machines in the same net, pretty basic same as first lvls, just need to match the masks and use same ip range.

since our mask is 255.255.255.128 our first range is 0 - 127 so using a network like 42.42.42 we get:

```
42.42.42.[0-127]
```

We can use one of the usable ips in this range and we get goal 1.

Goal-2: now we need to connect two machines in two networks over a router but we should use networks that are next to each other so we can latter use the same IP/CIDR to refer to both, basically using an ip start and mask that will encompass both networks.

Since we have the mask 255.255.192.0 (/18) this is a mask where some bits or for the network and some for the host in the same octect this is a **variable-length subnet masks (VLSM),** in this case in order to know how to identify our sub-networks (ip ranges) we can use the formula.

```
[network_variations] + [ip variation].[0-255]

[0/64/128/192]       + [0 - 63].[0-255]
```

Since our net uses 2 bits from the third octect that means that it varies like:

0000 0000 = 0 // 0100 0000 = 64 // 1000 0000 = 128 // 1100 0000 = 192

meaning that in the third octect we only have 4 network variations which makes sence since we only have 2 bits for the network in there.

Now for the host ips they will be the variations of the remaining 6 bits.

We should use the first and the second sub-networks, luckily in we already have a fixed ip (20.61.68.197) (it may change) which is in the second subnet so we just need to use the first subnet in the other network.

Network D = {20.61.64+[0-63].[0-255]/18}

Network C = {20.61.00+[0-63].[0-255]/18}

With this we get goal 2.

Host ip variations

0000 0000
0000 0001
0000 0010
0000 0011
0000 0100
0000 0101
0000 0110
0000 0111
0000 1000
0000 1001
....
0011 1111

Goal-3: now we need to connect A to the internet, at this point it probably says no reverse way which means its getting to the internet destination but isnt coming back, to solve this we need to add a route in the internet routing table. Since the next_jump is fixed to the only interface available we just need to add a destination for our network 2.42.42.0 and since the mask is 255.255.255.128 we use /25.

Now with this route 2.42.42.0/25 => 163.172.250.12 we complete goal 3.

Goal-4/5: this time we need to connect two machines across two routers all we need to do is to add a route in R1 so that packets meant for D or C go to R21 for this we need an IP/CIDR that encompasses both C and D.

Network D = {20.61.64+[0-63].[0-255]/18}

Network C = {20.61.00+[0-63].[0-255]/18}

So we need to go from [20.61.0.0] to [20.61.127.255] this can be done by…

we know we want a first range of 127 * 256 = 32512 ips, this means that 2^bits = 32512 so bits = log2(32512) = 14.98 = 15 thus we need 15 bits for the host portion of the ip leaving 32 - 15 = 17 s our CIDR is /17.

So we need to use a route 20.61.0.0/17 => R21, we need to set the ips for R21 and R13 we can choose any that are valid in the mask 255.255.255.252 so lets just go with:

R21 = 11.22.33.1 // R13 = 11.22.33.2

Now it should say no reverse way, this is because we need a rule in R2 that routes to its only avaliable interface, R13.

With this we have completed both goal 3 and goal 4.

Goal-6: it should say no reverse way since the way we set thigs up already allows for the packet to reach its destination.

We just need to add a rule in the internet routing table so it can send packets to C and D, note that we cant use default in the internet ip cuz it would send packets back to itself creating a loop.

We already have the IP/CIDR for C and D, the one we put in R1: 20.61.0.0/17 so we just put this into the internet routing table and bum! Solved.

There is one more lvl to go but im confident that if u made it this far u can beat it solo, and after that u have finished net_practice and more importantly u understood it.