RC

Course Contents

pl

Configure Terminal

in the terminal, type

`gnome-terminal --tab -t "{name}" -- telnet {host} {port}`

and replace the {name} by the desired, the {host}{port} by the indicated. For example,

`gnome-terminal --tab -t "{name}" -- telnet localhost 5000`

teórico-prática

IP Addressing

- ***What is an IP Address?***
  - IP Addresses allow to uniquely identify and communicate with **hosts** (servers, laptops, smartphones, etc) in the internet.

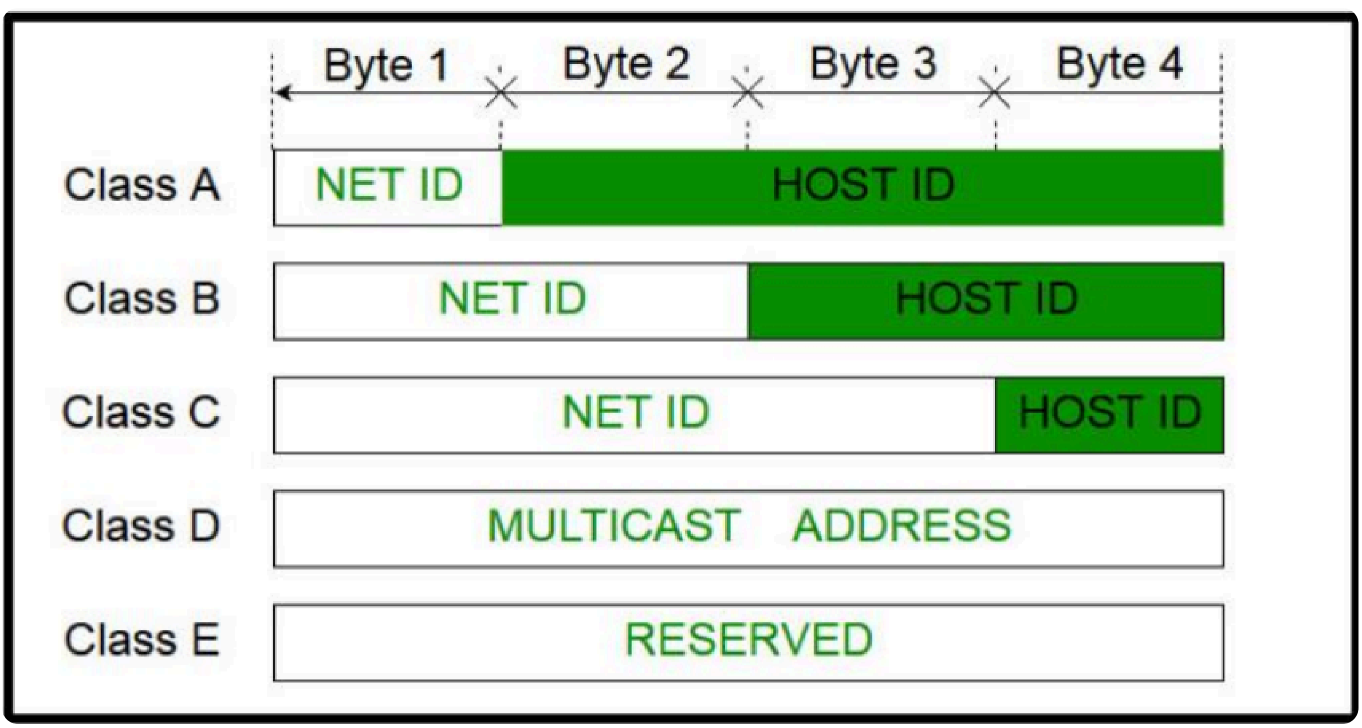
- IPv4 addresses are written as four numbers sepparated by periods ("."), each number can be from 0 to 255
- Alternatively, can be written in hexadecimal notation

IPV4 -> 32 bit used for addresses and provides a total of 4,294,967,296 addresses

IPv6 -> 128 bits used fot addresses and provides a total of 340,282,366,920,938,463,374,607,431,768,211,456 addresses

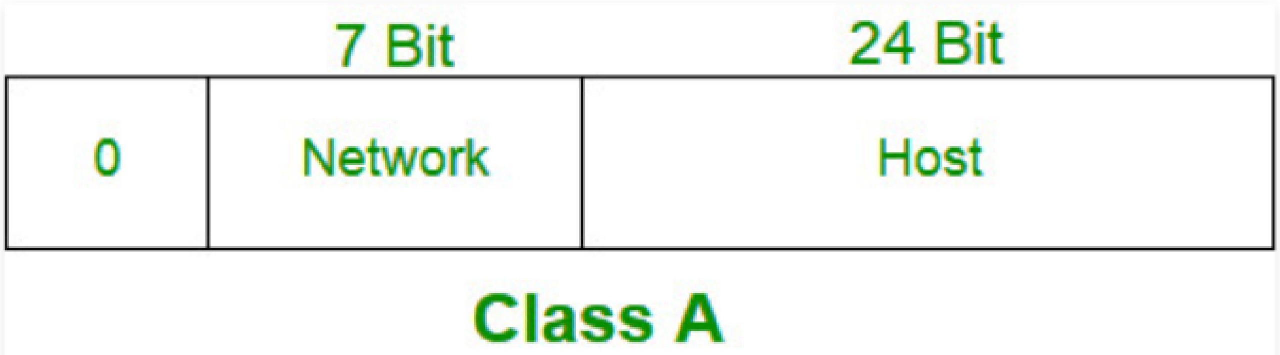
Classful Addressing

```
32 bit are divided into subclasses: A, B, C, D and E
Each class has a valid range of IP addresses
The order of he bits in the first byte / octet determines the classes of the IP addresses
Each IP address is divided in Network ID and Host ID
The class of the IP address determines the number of total networks and hosts possible in that particular class
IP addresses are globally managed by the Internet Registries Numbers Authority (IANA) and Regional Internet Registries (RIR)
Each ISP or Network Administrator assigns IP address to each device that is connected to its network
```

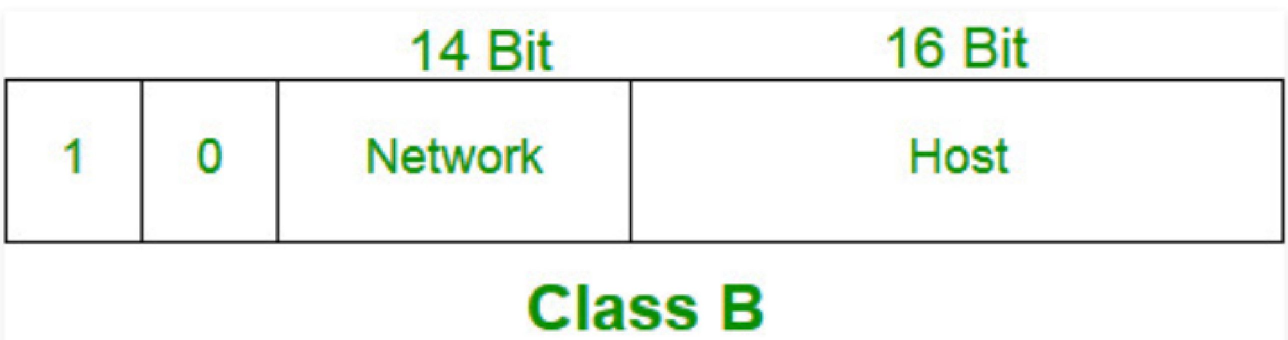


IPv4 classful addressing

- Class A IPv4 addresses
  - *(Nº OF HOSTS)* assigned to networks that contain a *large number of hosts*
  - *(BITS ORGANIZATION)* network ID is **8 bits (1 byte)** long, host ID is **3 bytes (24 bits)** long
  - *(PREFIX BIT)* higher order bit (left to right) of the first byte is always ***0***
  - *(RANGE)* Goes from **1.0.0.1** to **126.255.255.254**

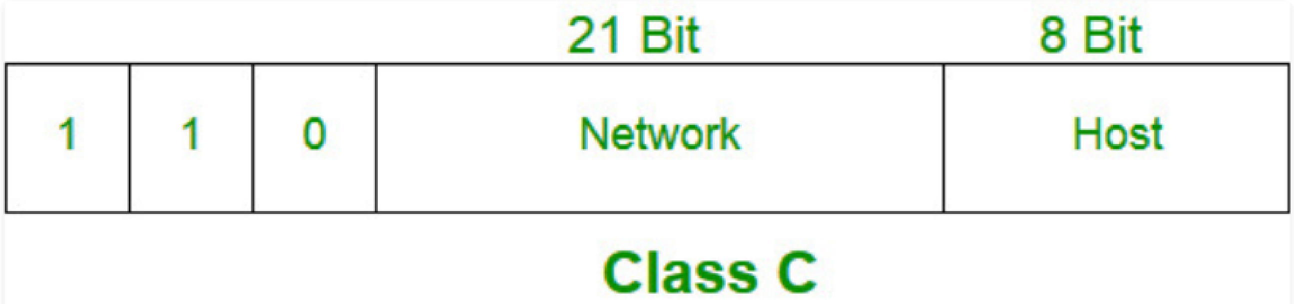


- Class B IPv4 addresses
  - *(Nº OF HOSTS)* Assigned to medium-sized to large-sized networks
  - *(BITS ORGANIZATION)* Network ID is **16 bits (2 bytes)** long, host ID is **16 bits (2 bytes)** long
  - *(PREFIX BITS)* higher order bit (left to right) of the first byte is always ***10***
  - *(RANGE)* From **128.1.0.1** to **191.255.255.254**



- Class C IPv4 addresses:
  - *(Nº OF HOSTS)* Assigned to medium-sized to large-sized networks
  - (BITS ORGANIZATION) Network ID is **24 bits (3 bytes)** long, Host ID is **8 bits (1 byte)** long
  - (PREFIX BITS) Higher order bits of the first byte is always ***110***

- *(RANGE)* From **192.0.1.1** to **223.255.255.254**



- **Class D** IPv4 addresses:
  - () // Multicast uses multiple devices to send data to a single destination through distinct streams or channels.
- *IPV4 vs IPV6*
- *IPV4 classful addressing*
- *IP addresses and netmasks*
- *Special addresses*
- *Reserved IP Addresses*

PL 2

255 . 255 . 252 . 0

Network	**HOST**
11111111 . 11111111 . 111111	00 . 00000000

(that's because the *DHCP* given is *10 . 20 . 192 . 0 / 22* which means that it's *mask* is *22* so the IP address will have *22 bits for the network* and 32 - 22 = *10* bits for the *host*)

Network Address	Mask
10.20.192.0	255.255.255.0

(Dynamic host config protocol)
DHCP -> **00001010 . 000101000 . 110000 | 00 .00000000**

- ***First valid usable network address***
  - -> 00 . 00000001 (don't use 00.00000000 because it is the Network Address)
- ***Last valid usable network address***
  - -> the rest keeps the same | 11.11111110 (don´t use 11.11111111 because it is the *broadcast* address)

PS: **Broadcast address** is when **all HOST bits are 1**, it is used to **send data to all devices** on a specific network

- **Classless Inter Domain Routing**

Range

---

i.e.

subnet	**10.20.192.10**
subnet mask	255.255.255.0 /24
subnet mask (notation)	/24
network address	10.20.192.0
range	10.20.192.1 to 10.20.195.254
broadcast address	10.20.192.255
b.a. binary	11000000 . 10101000 . 00000001 . 11111111

- Now we have the network = 193.136.224.0/20

for the problem we have 2 networks, we need 1 bit to *identify which one* --> **SR**

subNetwork 1	**subNetwork 2**	**SR**	**HOST**
10001000.1110.*0*.000.00000000	10001000.1110.*1*.000.00000000	0 | 1	

At this moment we have **2 subNetworks**: it gets the mask */21* now because **we need the 20 plus the 1 bit borrowed from the HOST** that is used to **identify the subNetwork (SR)**

1. 10001000 . 1110 | 0 | 000 . 00000000
2. *subNetworks* -> 11000001 . 10001000 . 1110 | 0 | 000 . 00000000

until

11000001 . 10001000 . 1110 | 1 | 000 . 00000000

See the IP in the console: **ipconfig**

%falta o T01%

PL 3

> !! Introduction to **NAT**

- Private Networks start by 10. ...
- **SNAT** -> (*Source NAT*) who initiates connection in private network (uses NAT)
- **DNAT** -> (*Destination NAT*)who initiates connection in public network (uses NAT)

GNS3 Tutorial

```
Router> enable
Password:
Router#
```

```
Router# configure terminal
Enter configuration commands, one per line. End with Ctrl-Z
Router(config)#
```

Este modo de configuração tem ainda vários sub-modos como o que permite configurar as diversas interfaces - comando **interface**.

```
Router0>enable                      // para entrar no modo privilegiado
Router0#show running-config         // para ver as interfaces presentes no router
                                    // use a barra de espaço para mudar a
                                    // página de configuração
(...)
Router0#configure terminal          // para entrar em modo de configuração
Router0(config)#interface FastEthernet0/0       // configurar interface
Router0(config-if)#ip address 10.254.0.2 255.255.255.0
Router0(config-if)#no shutdown            // activar a configuração da interface
Router0(config-if)#exit
Router0(config)#exit
```

ROUTES -> Defined for the routers to know there to send the packets. They can be:

Static: Configured manually in the equipment

Dynamic: Added to the router configuration automatically, by referral routing protocols

STATIC ROUTES

defining static routes

```
ip route destination_network subnet_mask default_gateway
```

removing static routes

```
no ip route destination_network subnet_mask default_gateway
```

referral router configuration is shown through

```
Router0#show ip route
```

- **Running Configuration**: is being *executed at the moment* and reflects *all the commands* executed to the moment. Stored in the *RAM* memory. Lost in shutdown.
- **Startup Configuration**: *executed in startup*, stored in *non-volatile router memory*. If the admin wants to *set the running config to permanent router config*, must *copy it from the RAM to the non-volatile memory*.

  Copy the Running Configuration to the Startup Configuration

  ```
  Router0# copy running-config startup-config
  ```

  Para repor a configuração *default* de uma interface (ex: FastEthernet0/0), durante a configuração fazer: `default interface FastEthernet0/0`

  tab completes the commands

  shortcuts:

  `conf t` (ou outras variações) em vez de `configure terminal`

  `sh runn` em vez de `show running-config`

  `sh int` em vez de `show interfaces`

  sh run --> see all the configurations inside the routers

Assignment 1

1

R_2A-)

```
R1 e R2
CIDR: 193.136.224.0
Mask: 255.255.248.0
Broadcast: 193.136.231.255
Gama: 193.136.224.1 - 193.136.231.254

R2 e R3
CIDR: 193.136.232.0
Mask: 255.255.248.0
Broadcast: 193.136.232.0
Gama: 193.136.232.1 - 193.136.239.254
```

R_2B-)

Assignment 2

NAT

- *WHAT* ->Allows to **map IP addresses** in the **communications in between networks**
- *HOW* -> Through **altering the origin or destiny address** in the **heather of the IP packets during its passage through a router**

SNAT is one of the most used techniques

- *WHEN* -> The Source NAT is used in the **communications between private IP addresses and the internet**
- *HOW* -> it **forces** the use of **public IP addresses**
- **used to change the private addresses to the public address**

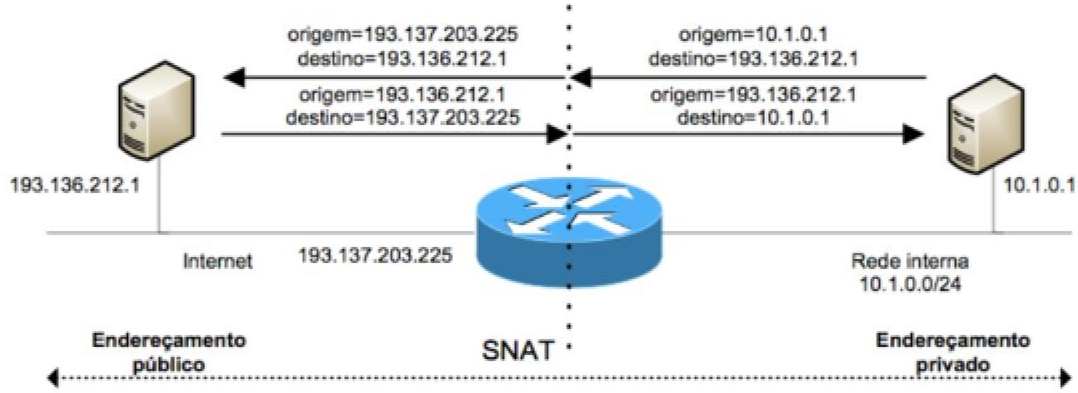


Figura 1 – Exemplo de funcionamento do NAT (SNAT)

DNAT

- *WHAT* -> The Destination NAT allows the translation of the destination addresses of the IP packets.
- *WHY* -> Commonly used for port forwarding or hosting public services on internal servers

key differences

SNAT	DNAT
applied to **outgoing traffic**	applied to **incoming traffic**
changes the **source address**	changes the **destination address**
used for **internet access from private networks**	used for **hosting services accessible from the internet**
hides internal network structure	**exposes** specific internal **services to the outside world**

!! comments

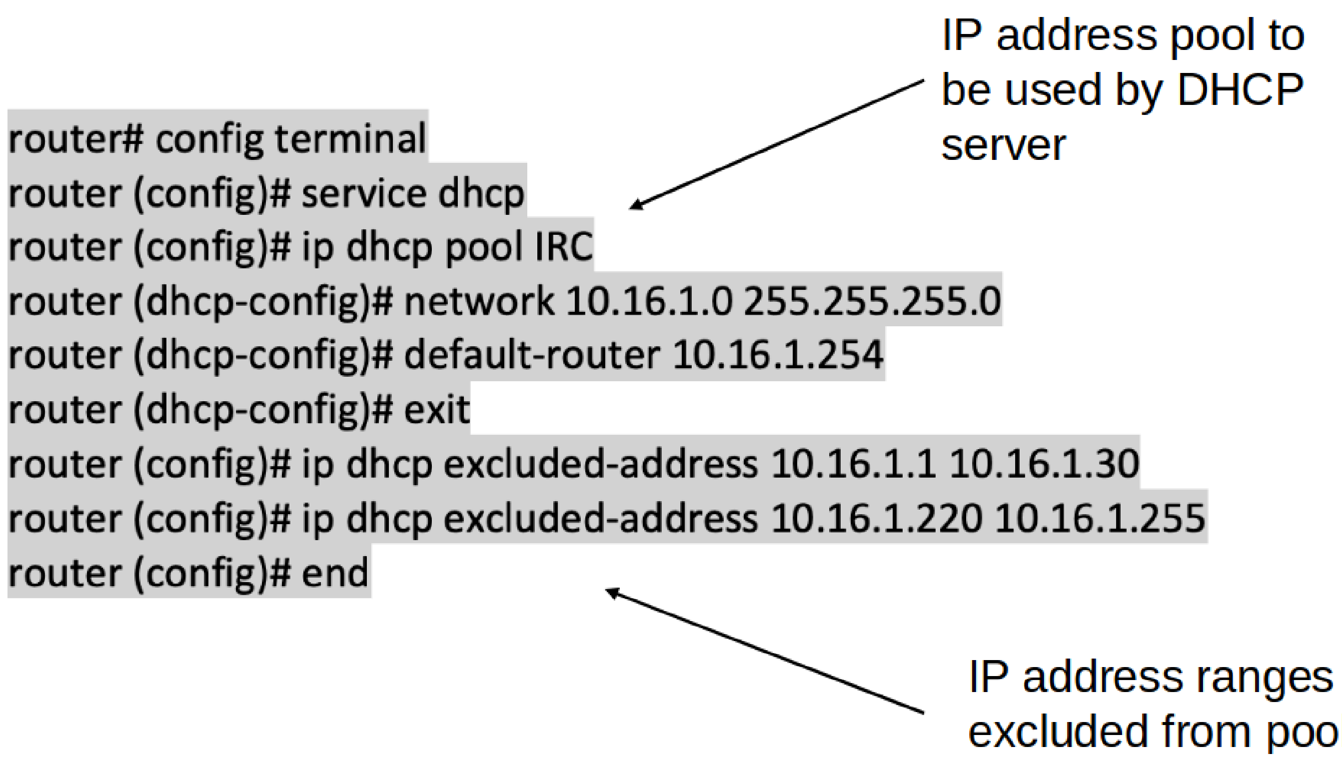
- there are no routs to private networks (private networks don't go to the internet)

TP 3

DHCP

- The own Router can serve as a DHCP server

DHCP Configuration Example



remember

Reserved vs Official IP Addresses

- *Reserved IP Addresses* are those that are set aside for **specific purposes** and **cannot be used for general public internet communication**.

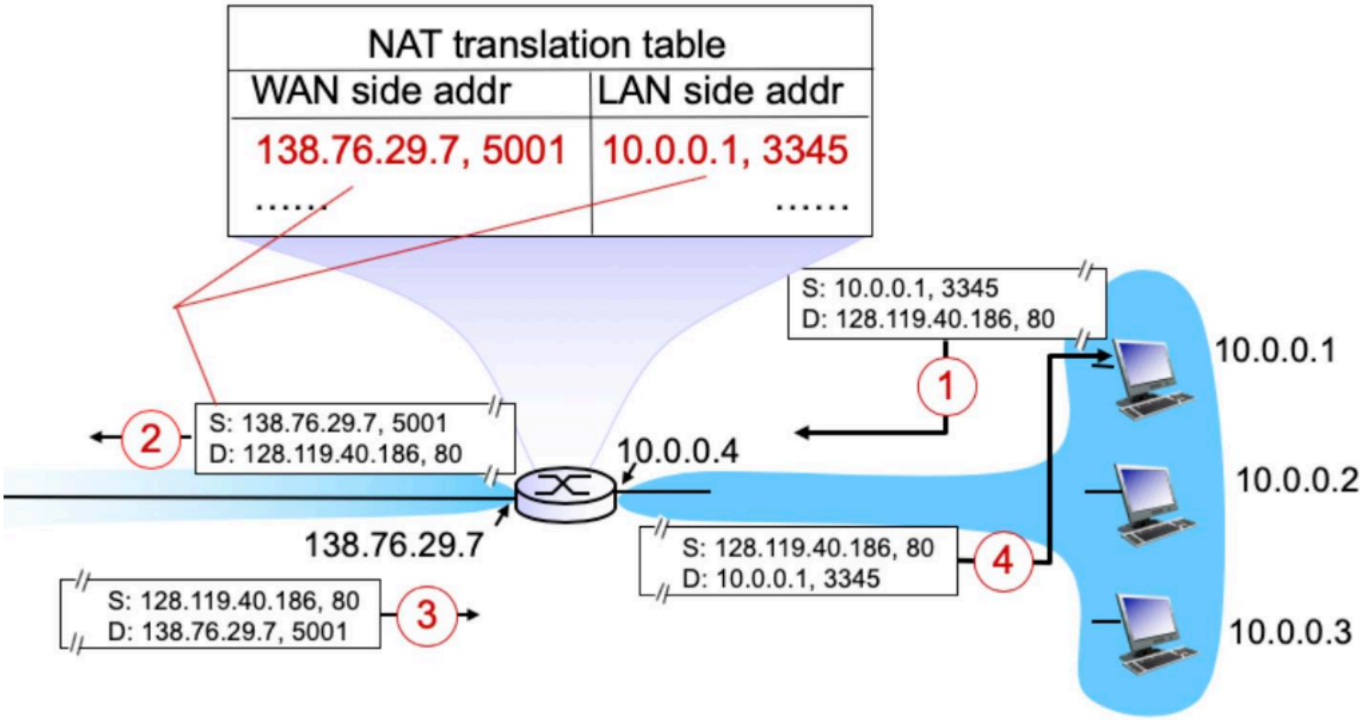
- Designed to be used on a private network behind a NAT (Network Address Translation) device (e.g. firewall or router)
- Cannot be used to communicate directly with other systems over the Internet
- Common usage in home, office and academic networks

Class A IP Range	Subnet Mask
10.0.0.0 – 10.255.255.255	255.0.0.0
172.16.0.0 – 172.31.255.255	255.240.0.0
192.168.0.0 – 192.168.255.255	255.255.0.0

- Official IP Addresses are those allocated by the Internet Assigned Numbers Authority (IANA) or regional internet registries (RIRs) for use on the public internet.

Aspect	Reserved IP Address	Official/Public IP Address
Purpose	Special uses like private networks, testing, etc.	General internet communication
Internet Routing	Not routable on the public internet	Routable on the public internet
Assignment	Predefined by IANA	Assigned by IANA or RIRs
Examples	192.168.x.x, 127.x.x.x	Any globally unique public IP address

NAT (SNAT) translation table



key aspects

- the router 10.0.0.4 knows the PC 10.0.0.1 wants to send a packet to the destination D: 128.119.40.186, port 80 (which is the HTTP address)
- the origin port of the PC is 3345, the origin port of the router is 5001, and it will search for the line in the translation table that matches the address of the router (S: 138.76.29.7) and will switch both the WAN (Wide Address Network) side address (Router S: 138.76.29.7, port 5001) and the LAN (Local Address Network) side address (PC S: 10.0.0.1, port 3345), so that the PC becomes the destination of the receiving packet.
- Then the source becomes the IP of the Server (S: 128.119.40.186, port 80) and the destination becomes the IP of the router (138.76.29.7, port 5001) and after the router, the source becomes the router and the destination becomes the client (PC)

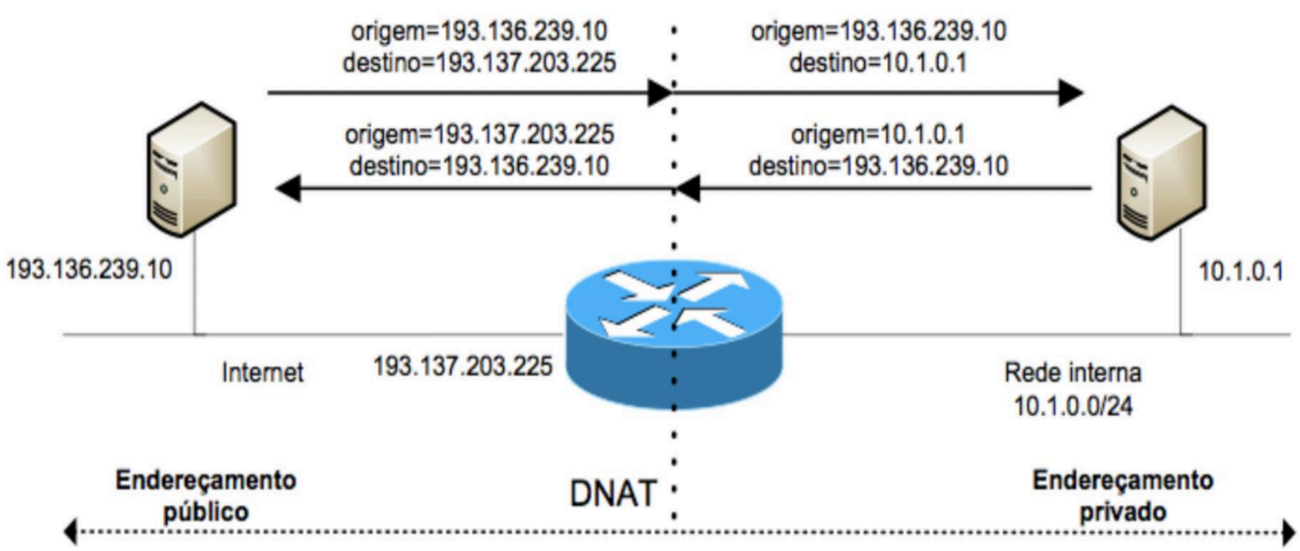
SNAT Configuration Example

Uses IP address of interface for SNAT

```
router# config terminal
router(config)# access-list 30 permit 10.5.0.0 0.0.0.255
router(config)# ip nat inside source list 30 interface Ethernet0 overload
router(config)# interface FastEthernet0
router(config-if)# ip address 10.5.0.1 255.255.255.0
router(config-if)# ip nat inside
router(config-if)# exit
router(config)# interface Ethernet0
router(config-if)# ip address 193.137.203.1 255.255.255.0
router(config-if)# ip nat outside
router(config-if)# end
```

Interfaces declared internal or external for NAT operations

NAT (DNAT) translation table



key aspects

NOTES

- DNAT no R1

- SNAT no R3
  (para a ficha 2)

T 2

Information in the routing tables (Tabelas de Encaminhamento) -> **Routers**

Principles of Network Application

!! **Application Layers**

TCP/IP Protocol Stack

```
An *application* supports many *protocols* in many *layers*
```

Heather

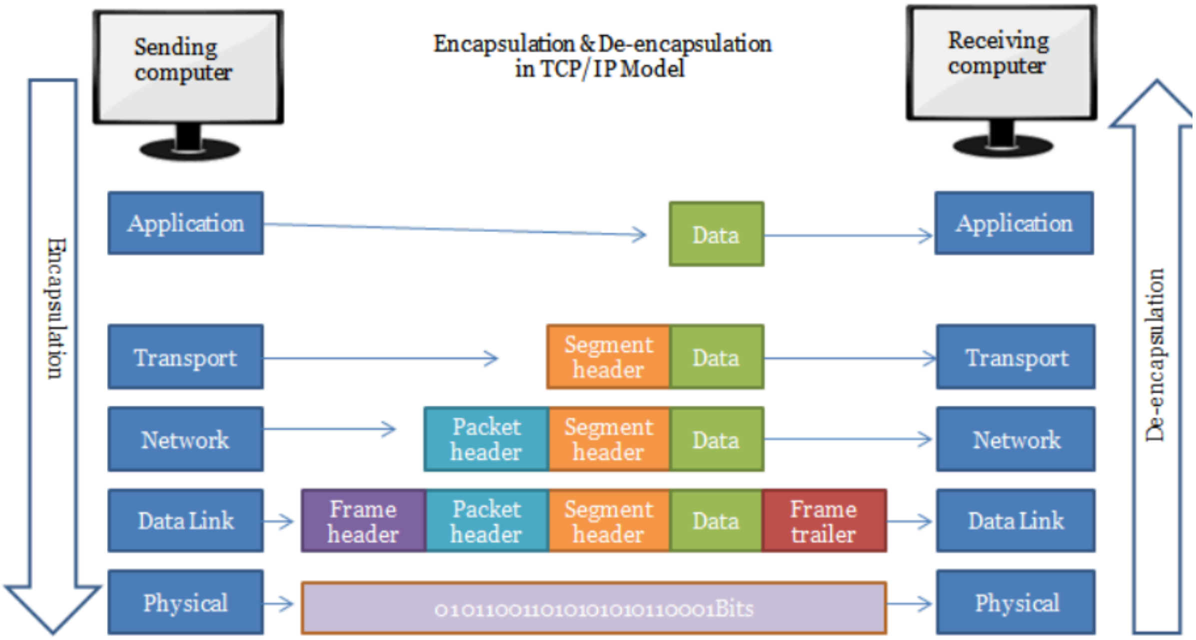
-> the point of a heather is **transport many information from an application by a protocol**, and the **protocol adds information in the heather** (it is control info). Then the *packet* (with the frame heather, packet heather, segment heather, data and frame trailer) arrives in the *physical layer*.

-> Encapsulation

HTTP

TCP

IP



payload -> packet (transported packet info)

- *Application:* supporting network applications
  - FTP, SMTP, HTTP, ...

Application
Transport
Network
Data Link
Physical

Application Layer

-> *Data Generation*

- The process begins at the **application layer**, where user data is **generated by applications** (e.g., HTTP, FTP, SMTP). This data is often referred to as the ***"payload"***

-> *Example*

`A user sends HTTP request to a Web Server`

Client-server Architecture

!! **Server**

Always on-host

Permanent IP address

Data centers for scaling

Client

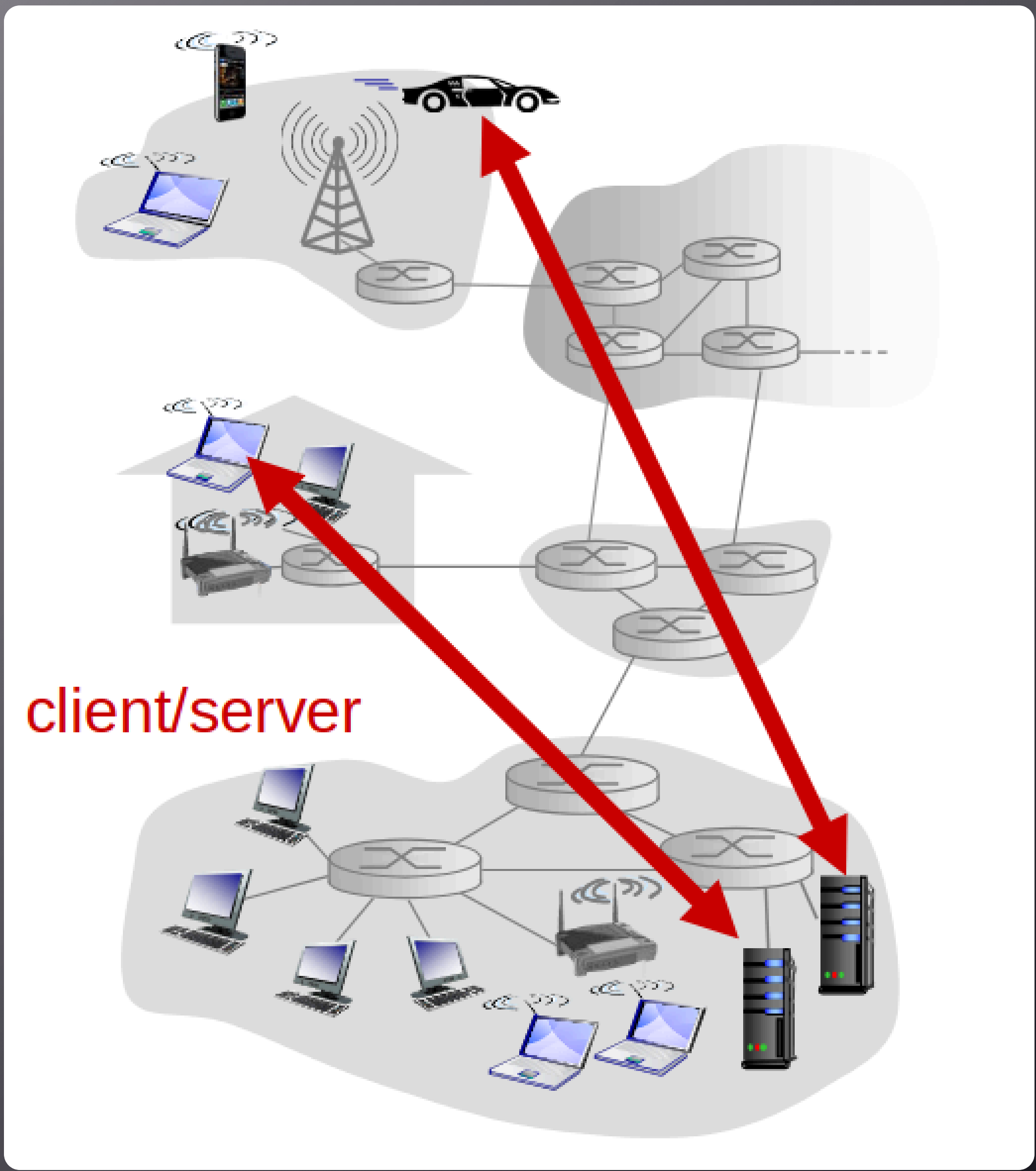
Communicate with the server

May be intermittently connected

May have dynamic IP addresses

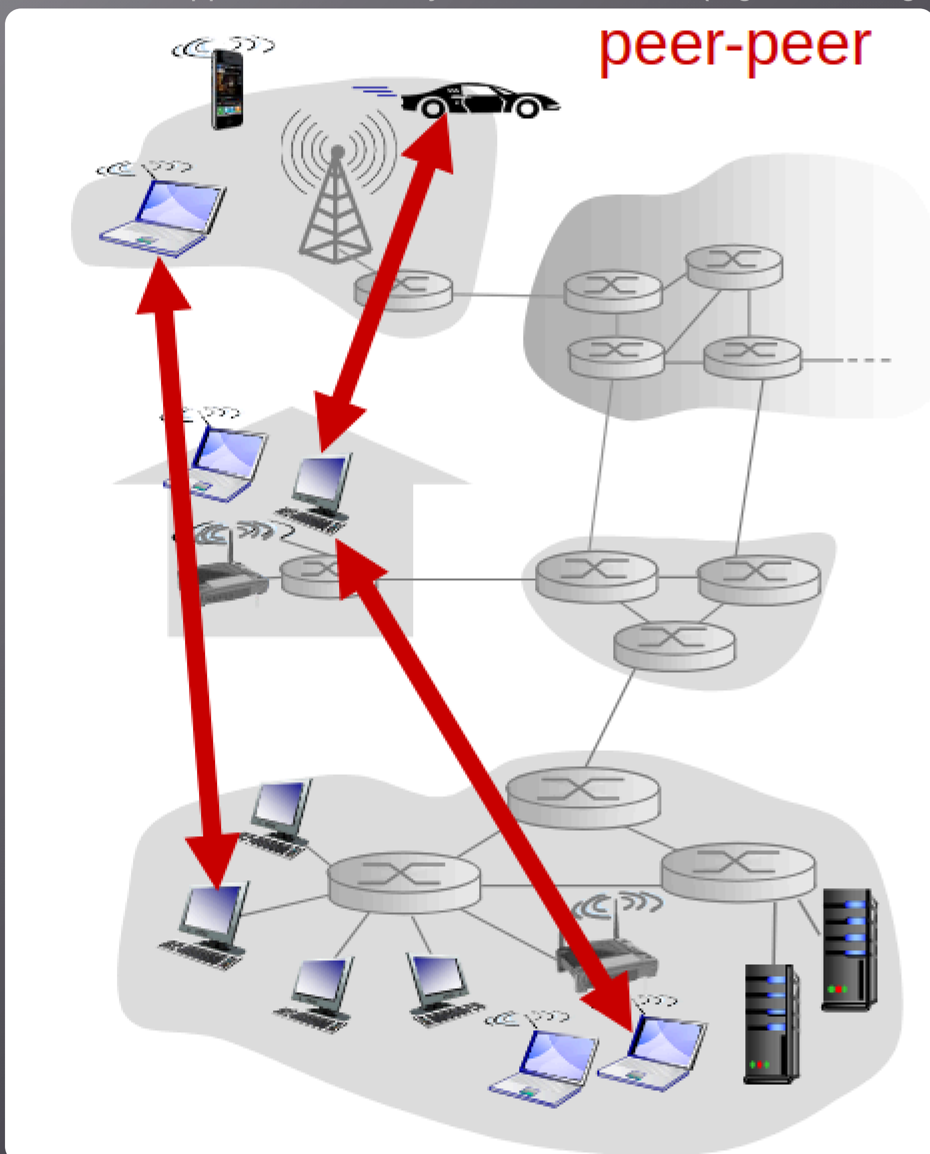
Do not communicate with each other

Examples: WEB ; FTP ; SSH ; E-Mail



P2P Architecture

!! *no* always-on server
arbitrary end systems communicate directly
peers request service from other peers, provide service in return to other peers
self scalability:
new peers bring new service capacity, as well as new service demands
peers are intermittently connected and change IP addresses
Complex Management
Example: BitTorrent, Skype, IPTV
Also, some applications use hybrid architectures (e.g. in messaging)

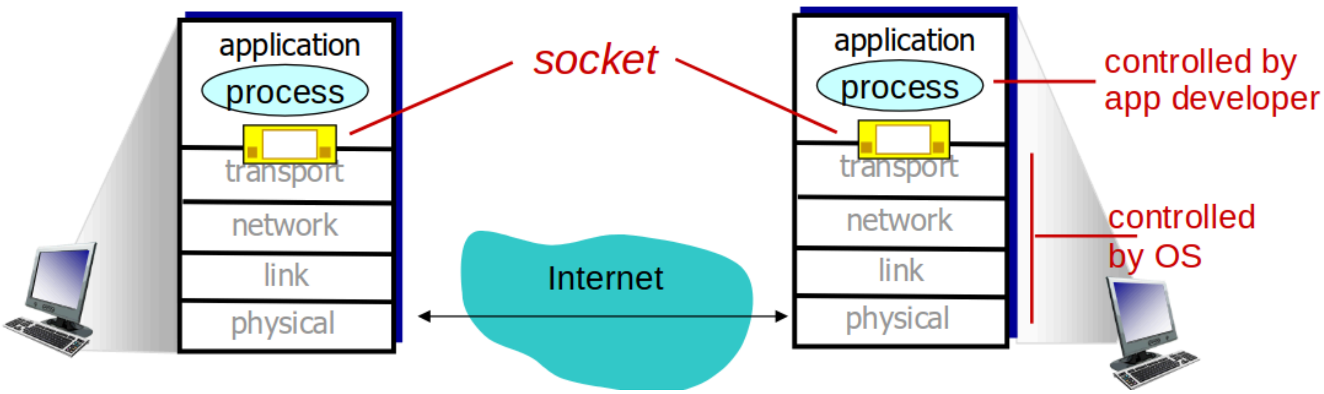


Processes -> program running within a *host*

Client Processes	Server Processes
Process that initiates communication	Process that waits to be contacted

SOCKETS

- process *sends/receives* messages *to/from* its **socket**
- A socket is the **software interface** between the *process* and the *computer network* (between the application layer and the transport layer)
- Also called Application Programming Interface (***API***)
- The application *chooses* the **transport protocol** (e.g. UDP or TCP) to *use the transport-layer services provided* by the protocol



Addressing Processes

- To receive processes, *applications* must have an **identifier**
- Host device has *unique* ***32-bit*** IP address

> !! *Q*: does IP address of host on which process runs suffice for identifying the process?
> *R*: no, many processes can be running on same host

- identifier includes both **IP address** and **port numbers** associated with process (the *socket*) on host.

> !! *Example* of port numbers
> HTTP server: 80
> Mail server: 25

-> Example of addressing processes

Port #	Application Layer Protocol	Type	Description
20	FTP	TCP	File Transfer Protocol - data
21	FTP	TCP	File Transfer Protocol - control
22	SSH	TCP/UDP	Secure Shell for secure login
23	Telnet	TCP	Unencrypted login
25	SMTP	TCP	Simple Mail Transfer Protocol
53	DNS	TCP/UDP	Domain Name Server
67/68	DHCP	UDP	Dynamic Host
80	HTTP	TCP	HyperText Transfer Protocol
123	NTP	UDP	Network Time Protocol
161,162	SNMP	TCP/UDP	Simple Network Management Protocol
389	LDAP	TCP/UDP	Lightweight Directory Authentication Protocol
443	HTTPS	TCP/UDP	HTTP with Secure Socket Layer

!! **Types of messages exchanged**

e.g., request, response

Message syntax

what fields in messages & how fields are delineated

Message Semantics

meaning of information in fields

Rules

for when and how processes send & respond to messages

Open protocols

defined in RFCs (documents)

allows for interoperability

e.g., HTTP (HyperText Transfer Protocol), SMTP (Simple Mail Transfer Protocol)

proprietary protocols

e.g. Skype

What transport service does an app need?

!! **Data integrity**

some apps require 100% reliable data transfer

e.g., file transfer, web transactions

other apps can tolerate some loss

e.g., audio

timing

some apps require low delay to be “effective”

e.g., Internet telephony, interactive games

throughput

some apps require minimum amount of throughput to be “effective”

e.g. multimedia

other apps make use of whatever throughput they get

called “elastic apps”: e.g. e- mail, web, file transfer

Security

Encryption, data integrity, ...

Transport service requirements: common apps

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
“real-time” audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100’s msec
stored audio/video	loss-tolerant	same as above	
interactive games	loss-tolerant	few kbps up	yes, few secs
text messaging	no loss	elastic	yes, 100’s msec
			yes and no

(TCP / UDP) Internet Transport Protocol Services

-> *Segmentation (TCP) or Datagram Creation (UDP)* --> PROTOCOL SERVICES

- The data from the application layer is passed to the **transport layer**, where it is **divided into smaller chunks** called **segments** (for TCP) or **datagrams** (for UDP)

-> *Header Addition*

- ***TCP (segmentation)***

- Adds a *TCP Header* which includes *source and destination port numbers*, *sequence numbers*, *acknowledgment numbers*, and other control information
- ***UDP (datagram creation)***
  - Adds a *UDP header*, which includes *source and destination* and a *checksum*
- ***example***: If using TCP, a **TCP segment is created with a header** containing the **source port** (e.g. 80) and a **destination port** (e.g. 54321)

Internet Layer (IP)

-> *Encapsulation into IP Packet*

- The **segment or datagram** from the transport layer is **passed to the internet layer** where it is **encapsulated into an IP packet**

-> *Header Addition*

- **An IP header is added**, which includes **source and destination IP addresses, protocol type (e.g., TCP, UDP)** and other information such as *time-to-live (TTL)* and *fragmentation control*

-> **Example**: The **TCP segment** is **encapsulated into an IP packet with the source IP address** (e.g., 192.168.1.1) and **destination IP address** (e.g., 192.168.1.2)

IP packets cannot be fully trusted to be sent and received successfully because the Internet Protocol (IP) operates on a **best-effort delivery model**, which means it does not guarantee packet delivery, order, or integrity. There is the possibility of **packet loss**. That is why TCP protocol is used (is reliable and safe, but slower than UDP)

Security Mechanisms