Computer Networks. Unit 2: IP

Notes of the subject *Xarxes de Computadors, Facultat Informàtica de Barcelona, FIB*

Llorenç Cerdà-Alabern

March 15, 2019

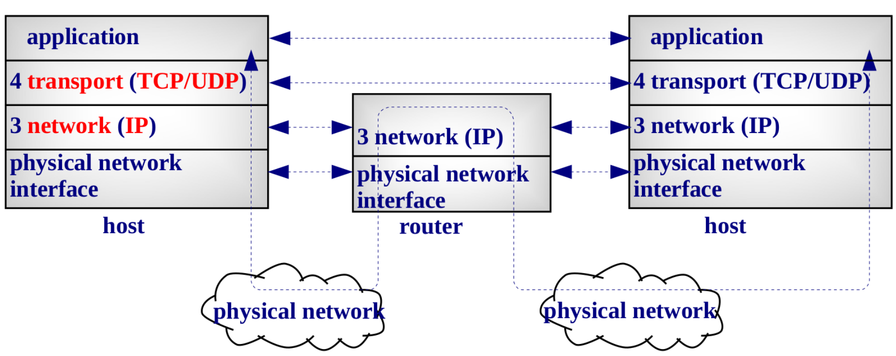
Contents

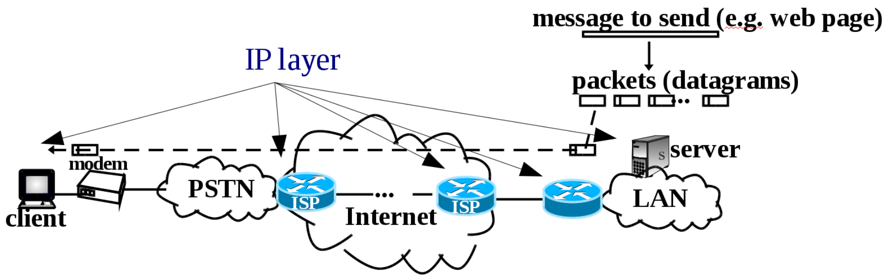
2 Unit 2: IP

2.1 IP Protocol RFC791

2.1.1 Who run the protocol

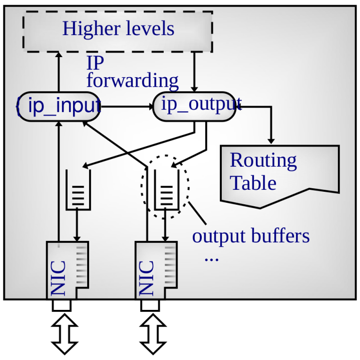
- **Hosts** and **Routers** run the IP protocol





2.1.2 IP Service URL

- Connectionless
- Stateless
- Best effort



Router Arquitecture

2.1.3 IPv4 Header RFC791

Datagram (layer 3 packet in TCP/IP)

```
+--------------------+------------------------+
| header             | payload                |
+--------------------+------------------------+
```

IP Header

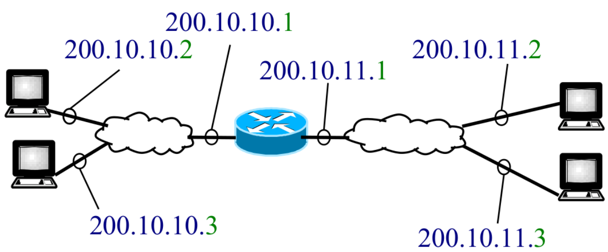
```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 bits
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.2 IPv4 Addresses

2.2.1 netid/hostid

- **32 bits** (4 bytes)
- **Dotted notation** 147.83.24.28

```
 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 bits
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          netid                /              hostid           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



2.2.2 Assigment

- IP addresses must be **unique**
- Internet Assigned Numbers Authority, IANA assign IPs to **Regional Internet Registries**, RIR:
  - RIPE: Europe
  - AFRINIC: Africa
  - ARIN: USA
  - APNIC: ASIA
  - LACNIC: Latin America
- RIR assign IPs to **ISPs**, ISPs to their customers

whois (bash)

```
whois 147.83.34.1
```

2.2.3 IPv4 address classes

- Most Significant bits identify the class
- Bits of netid/hostid varies in classes **A/B/C**
- **D** Class is for **multicast addresses** URL
  - e.g. 224.0.0.2: "all routers"
- **E** Class are **reserved addresses**

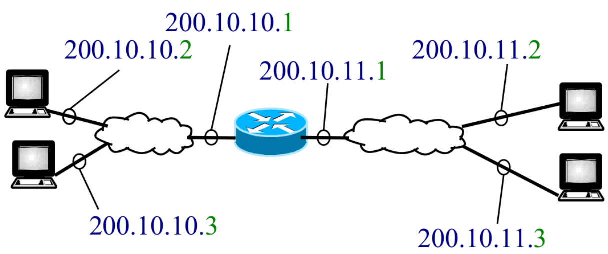
Class	netid	hostid	MSB	range
A	1	3	**0** xxx	**0**.0.0.0~
B	2	2	**10** xx	**128**.0.0.0~
C	3	1	**110** x	**192**.0.0.0~
D	-	-	**1110**	**224**.0.0.0~
E	-	-	**1111**	**240**.0.0.0~

MSB: Most Significant Bits

2.2.4 IPv4 address assignment

- @IP are assigned to **network interfaces**
- **netid** identifies a network
- **hostid** identifies a host

```
 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 bits
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          netid                 /              hostid           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



2.2.5 Special Addresses

netid	hostid	Meaning
any	all 0	Network address Used in routing tables
any	all 1	broadcast address
all 0	all 0	this host in this net. Source IP in DHCP
all 1	all 1	broadcast in this net. Dest IP in DHCP
127	any	host loopback

Practical examples (bash)

```
/sbin/ifconfig eth0
ping 127.0.0.1
```

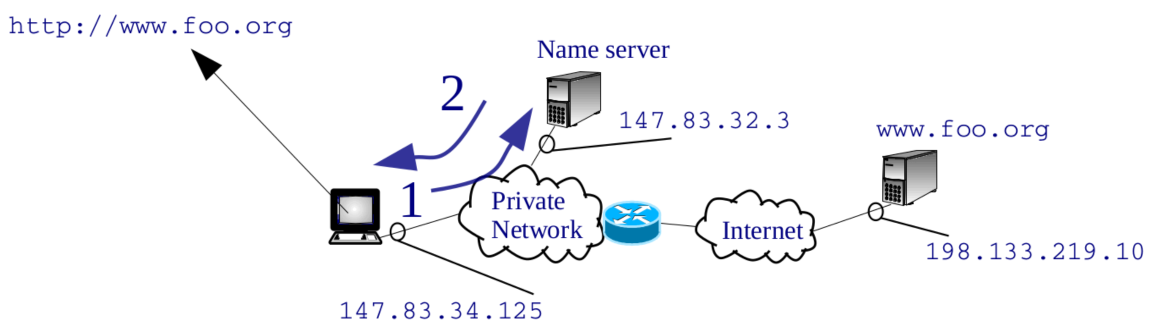
2.2.6 Private IPv4 Addresses RFC1918

- Not assigned to any RIR
- Not unique
- Non routable in the Internet

Class	Networks	Addresses
A	1	**10**.0.0.0
B	16	**172.16**.0.0 ~ **172.31**.0.0
C	256	**192.168.0**.0 ~ **192.168.255**.0

2.2.7 Domain Name System, DNS URL

- **EXPLAINED IN DETAIL IN UNIT 5**
- Convert **names** into **IP** addresses
- **Client-server** paradigm
- Short messages uses **UDP**
- Well-known port: **53**



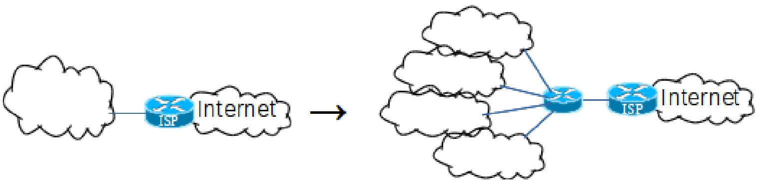
DNS (bash)

```
nslookup
tcpdump -ni wlan0 port 53
```

2.3 Subnetting RFC950

2.3.1 Motivation

- Split a large network into smaller ones



2.3.2 Network Mask

- Allow any number of bits for netid/hostid
- The **mask** identify **#bits of netid**
- Notation in **bits**: 147.84.22.3 **/24**
- **Dotted** notation (traditional): /24 = **255.255.255.0**

example: **147.84.22.3/24**

	dotted not.	binary
address	147.84.22.3	10010011 01010100 00010110 00000011
mask	255.255.255.0	11111111 11111111 11111111 00000000

ifconfig (bash)

```
/sbin/ifconfig wlan0
```

2.3.3 Variable Length Subnet Mask (VLSM)

- Allows subnets of different size
- **Example**: subnetting a class C address:
  - We have 1 byte for subnetid + hostid
  - Subnetid is green
  - chosen subnets addresses are underlined

$$\left.\begin{array}{l}\underline{0000}\\1000\end{array}\right\}\longrightarrow\left.\begin{array}{l}\underline{1000}\\1100\end{array}\right\}\longrightarrow\begin{array}{l}\underline{1100}\\\underline{1101}\\\underline{1110}\\\underline{1111}\end{array}$$

- **Example**
- **Base address** 200.0.0.0/24

Using the previous subnetting scheme, for each subnet show:

1. Subnetid in bits
2. Network address
3. Address range
4. Broadcast address
5. Number of IP addresses

- **Solution**
- **Base address** 200.0.0.0/24. **B=200.0.0**

Subnetid	Net. addr.	Addr. range	Broad.	Num. of IP
0	B.0/25	B.0~B.127	B.127	2^7=128
10	B.128/26	B.128~B.192	B.192	2^6=64
1100	B.192/28	B.192~B.207	B.207	2^4=16
1101	B.208/28	B.208~B.223	B.223	2^4=16
1110	B.224/28	B.224~B.239	B.239	2^4=16
1111	B.240/28	B.240~B.255	B.255	2^4=16

Exercise (subnetting) quiz assessment C1 spring 2018, questions 2,3

2.3.4 Classless Inter-Domain Routing, CIDR RFC1519

- **Classless** routing (use masks)
- Rational **geographical-based** distribution of IP addresses
- Facilitate the router address **aggregation**

Aggregation example:

```
200.1.10.0/24+200.1.11.0/24 -> 200.1.10.0/23
```

- **Aggregation rules** are specified in the routing algorihtm (RA)
- One aggregation scheme (used in the RA called RIP) is:
- **Summarization:** aggregation at a class boundary

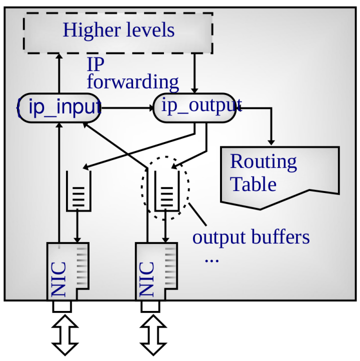
Summarization example (class C address):

```
192.168.0.0/27+192.168.0.128/27 -> 192.168.0.0/24
```

2.4 Routing Table (RT)

2.4.1 Who use the routing table?

- **ip$_{\textbf{output}}$**() use the RT to route each datagram
- **Direct Routing**: Destination directly connected
- **Indirect Routing**: Otherwise. Sent to a **gateway**
- Default route: **0.0.0.0/0**



Router Arquitecture

2.4.2 What's in the RT?

- Routing information:
  - Destinations: **network / mask**
  - How to reach them: **gateway / interface**
- **NOTE**: the gateway is the IP address of a router from a **directly connected network**

Practical examples

```
/sbin/route -n
```

List of public **BGP route servers**

- https://www.bgp4.net/doku.php?id=tools:ipv4_route_servers
- http://www.netdigix.com/servers.html

```
telnet route-views.routeviews.org
# telnet route-server.gblx.net
# telnet route-server.ip-plus.net
# telnet route-server.ip.tiscali.net
```

2.4.3 Datagram Delivery Algorithm

Datagram Delivery Algorithm (c)

```
1. if(IP-dest. == address any interf.) {
    sent to loopback interface
   }
2. for(each routing table entry
     ordered from longest to shortest netid)
   /* Longest Prefix Match */ {
      if((IP-dest. & mask) == Net-dest. RT) {
    return(gateway, interface) ;
      }
   }
3. if(it is a direct routing) {
     send the datagram to the IP-dest
   } else { /* indirect routing */
     send the datagram to the gateway
   }
```

- **NOTE**: the gateway is the IP address of a router from a **directly connected network**

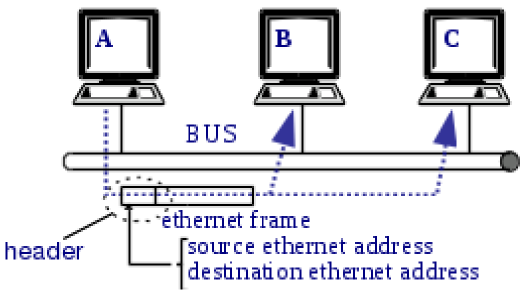
  Practical examples: adding static entries in the RT

```
/sbin/route add -host <IPhost> gw <IPgw>
/sbin/route add -net <IPhost> netmask <IPmask> gw <IPgw>
/sbin/route add default gw <IPgw>
```

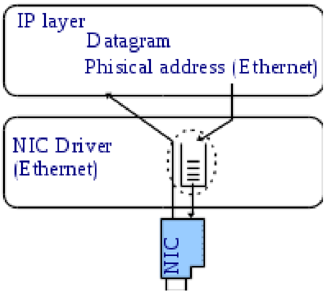
2.5 ARP protocol RFC826

2.5.1 Motivation

- Physical networks use addresses, e.g. Ethernet

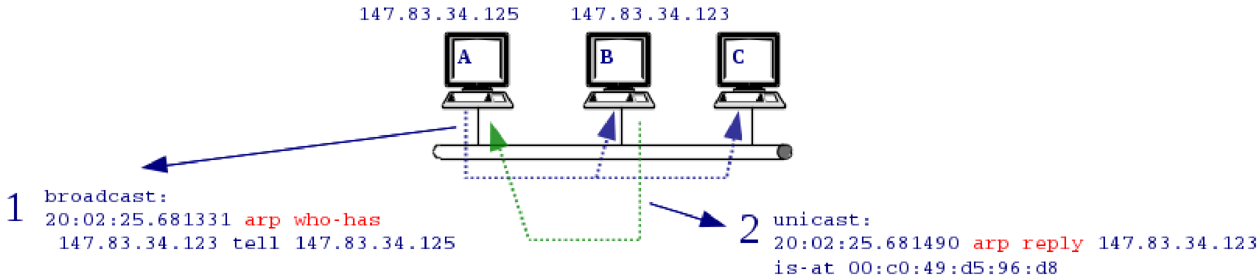


- IP layer pass a **physical address** to NIC driver
- IP calls **ARP** to obtain the physical addresses



2.5.2 Address Resolution

- When IP calls ARP
  - ARP looks the **ARP table**
  - If not found, ARP resolution:



ARP Fundamentals

- Encapsulated directly in L2 frames
- ARP Request: **broadcast** frame
- ARP Reply: **unicast** frame
- ARP table with **IP** <-> **MAC** address
- ARP entries are removes after an **aging time**

ARP Message

```
 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 bits
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Hardware Type (16)            |  Protocol Type (16)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Hard. Length(8)|Prot. Length(8)|  Opcode (16)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Sender Hardware                                         |
+       Address (48)            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               | Sender Protocol Address (32)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sender Protocol Address (cont)|     Target Hardware           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     Address (48)              +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Target Protocol Address (32)                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Practical examples

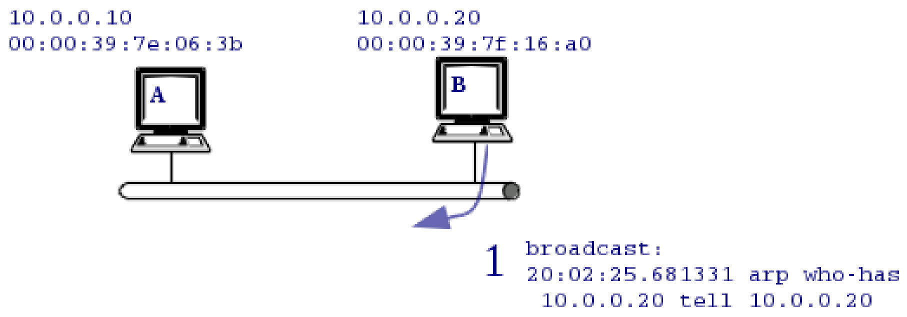
ARP (bash)

```bash
/usr/sbin/arp -n # show ARP table
capture an ARP resolution with wireshark
```

- **Exercise:** ARP resolution in a ping broadcast. The devices responding the ping message will initiate the ARP resolution.

2.5.3 Gratuitous ARP

- A host request its own IP
  - Detect duplicated IP addresses
  - Update MAC addresses in ARP tables



2.6 Internet Control Message Protocol, ICMP RFC792

2.6.1 ICMP Fundamentals

- **Error** or **query** messages
- Can be **generated** by IP, TCP/UDP, and application layers
- **Encapsulated in an IP** datagram (**no UDP/TCP!**)
- **Error messages** are sent to the **source IP address** of the datagram that generates the error condion
- An ICMP error message cannot generate another ICMP error message

2.6.2 ICMP error message format

- IP header + first **8 bytes** of the payload
- Used to identify the **TCP/UDP ports**

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.6.3 Common ICMP messages RFC792

Type	Code	query/error	Name	Description
0	0	query	echo reply	Reply an echo request
3	0	error	network unreachable	Network not in the RT.
	1	error	host unreachable	ARP cannot solve the address.
	2	error	protocol unreachable	IP cannot deliver the payload
	3	error	port unreachable	TCP/UDP cannot deliver the payload
	4	error	fragmentation needed and DF set	MTU path discovery
4	0	error	source quench	Sent by a congested router.
5	0	error	redirect for network	When the router send a datagram by the same interface it was received.
8	0	query	echo request	Request for reply
11	0	error	time exceeded, also known as TTL=0 during transit	Sent by a router when `--TTL=0`

Practical examples (wireshark)

- capture ICMP echo request/reply
- capture ICMP port unreachable

Exercise from collection: problem 1, a,b,c,d

2.7 IP Header

20 bytes

```
 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 bits
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- **Version**: 4
- **IP Header Length** (IHL):
  - Header size in 32 bit words
- **Type of Service**, bits: xxxdtrc0
  - xxx user defined,
  - dtrc: delay, throughput, reliability, cost
- **Total Length**: Datagram size in bytes
- **Identification/Flags/Fragment Offset**: fragmentation
- **Time to Live** (TTL): run by routers

```
if(--TTL == 0) { /* discard datagram */ }
```

- **Protocol**: Encapsulated protocol
  - see /etc/protocols
- **Header Checksum**:
  - Header error detection
- **Options**: (rarely used in practice)
  - Record Route
  - Loose Source Routing
  - Strict Source Routing

2.7.1 IP Fragmentation

- Motivation

Fragmentation may occur:

- Router: Fragmentation may be needed when two networks with different **Maximum Transfer Unit (MTU)** are connected
- Host: may be needed using **UDP**

send a UDP datagram of 5000 bytes (bash)

```
sudo tcpdump -vni wlan0 udp and host 10.0.0.1
```

send a UDP datagram of 5000 bytes (perl)

```perl
use IO::Socket;
use strict;
use Data::Dumper;

my $sock = IO::Socket::INET->new(
    Proto    => 'udp',
    PeerPort => 3555,
    PeerAddr => '10.0.0.1',
) or die "Could not create socket: $!\n";

(my $message = sprintf "%-5000s", "1") =~ tr/ /1/;
print localtime() . ": sending " . substr($message, 0, 10) . " x " . length($message) . "\n" ;
$sock->send($message) or die "Send error: $!\n";
```

Fields:

- **Identification** (16 bits):
  - identify fragments from the same datagram
- **Flags** (3 bits):
  - **D**, don't fragment. Used in TCP **MTU path discovery**
  - **M**, More fragments: 0 only in the last fragment
- **Offset** (**13 bits**):
  - Position of the fragment **first byte** in the original datagram in **8 byte words** (indexed at 0)

20 bytes

```
 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 bits
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example

- What are the fragments generated by a UDP datagram of 5000 bytes?
- Note:

UDP header is 8 bytes Network MTU is 1500 bytes

$$\text{fragment size} = \left\lfloor \frac{\text{MTU} - 20}{8} \right\rfloor$$

Exercise (fragmentation) quiz assessment C1 spring 2012, question 7

2.8 Dynamic Host Configuration Protocol, DHCP

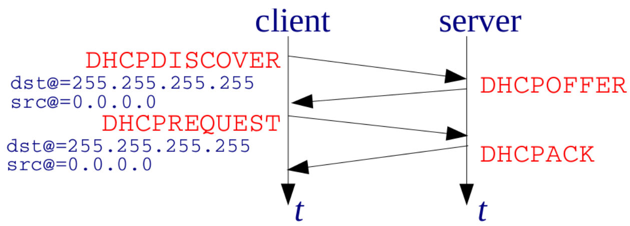
RFC2131 RFC2132 (options)

2.8.1 Objectives

- automatic **network configuration**:
  - Assign **IP** address and mask,
    - **Dynamic**: During a leasing time
    - **Automatic**: Unlimited leasing time
    - **Manual**: to specific MAC addresses
  - Default route,
  - Hostname,
  - DNS domain,
  - Configure DNS servers,
  - etc

2.8.2 DHCP Fundamentals

- **Client server** paradigm
- **UDP**, well known port 67 (client 68)
- Backward compatible with **BOOTP** (bootstrap protocol)
- Messages



- NOTES:
  - Cient messages are always **broadcast**, server messages can be **unicast or broadcast** (requested by the client)
  - If a previous DHCP session has been recorded the client can directly send **DHCPREQUEST**

Practical examples

Capture DHCP messages with wireshark (bash)

```
$ sudo wireshark
$ ps aux | egrep dhclient
$ sudo killall dhclient
$ sudo dhclient wlan0
```

Exercise (dhcp) quiz assessment C1 spring 2014, question 3

2.9 Routing Algorithms

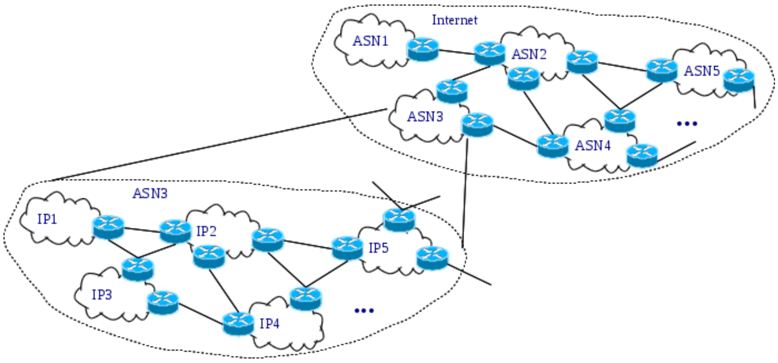
2.9.1 What is a routing algorithm?

- Objective: initialize routing tables

  Static: Manual, scripts, DHCP **Dynamic**: protocol between routers, routing algorithm

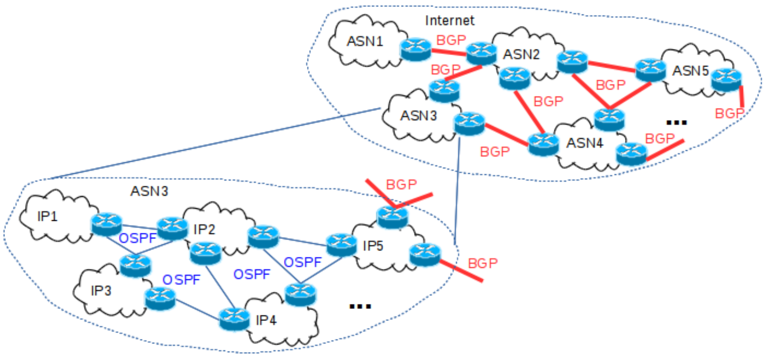
2.9.2 What is an Autonomous Systems (AS)?

- Internet is organized in *Autonomous Systems* (**AS**) RFC1930: An **AS** is a connected group of one or more IP prefixes run by one or more network operators which has a **single and clearly defined routing policy**
- Typically, every **ISP** is a different AS



2.9.3 Routing algorithms classification

- *Interior Gateway Protocols* (**IGP**): **Inside AS**
  - RFC standards: **RIP** (RFC2453), **OSPF** (RFC2328)
  - Proprietary: e.g. CISCO IGRP
  - Routes **minimize a *metric** (cost)
- *Exterior Gateway Prot.* (**EGP**): **Between AS**, **BGP** (RFC4271)
  - Route preferences satisfy **commertial agreements**
  - **BGP basis**: routers exchange IP prefixes/AS paths/attributes



2.9.4 Routing Information Protocol, RIP RFC2453

(only routing protocol we will study in detail)

- **Metric**: number of hops (networks)
- **Broadcast** RIP updates to **neighbors** every **30 seconds**
- **UDP**, src./dst. well-known port = 520
- RIP **updates** include **destinations** and **metrics**
- A neighbor is considered down if no update in **180 s**
- **Infinite metric** is **16**
- **Route Summarization**: aggregation to class
  - 192.168.0.0/25+192.168.0.128/25->192.168.0.0/24
- RIP **version 2**:
  - allows variable masks
  - multicast dst. 224.0.0.9

Count to Infinity

- RT when RIP converge

D	G	M
N1	*	1
N2	*	1
N3	R2	2
N4	R2	3

R1's RT

D	G	M
N1	R1	2
N2	*	1
N3	*	1
N4	R3	2

R2's RT

D	G	M
N1	R2	3
N2	R2	2
N3	*	1
N4	*	1

R3's RT

- Possible evolution of **D=N4** entry when **R3 fails**:

	G	M	R3 fails	G	M	R1 upd	G	M	R2 upd	G	M	R1 upd	G	M		G	M
R1:	R2	3	→	R2	3	→	R2	3	→	R2	5	→	R2	5	...	R2	16
R2:	R3	2	→	R3	16	→	R1	4	→	R1	4	→	R1	6	...	R1	16

Count to Infinity Solutions

- **Split horizon** removes the entries learned from a gateway in the interface where the update is sent
- **Triggered updates** send the update when a metric changes (do not wait 30 seconds)
- **Hold down timer** unreachable routes are in holddown (not updated) during 180 seconds

Exercise (RIP) quiz assessment C1 fall 2016, question 8

Practical example

- RIP with packettracer
- Basic **IOS RIP configuration commands**
  - **router rip** # configure RIP daemon
  - **network** a.b.c.d # export network

2.9.5 Open Shortest Path First, OSPF RFC1131

(**only introduction**)

- IETF standard for **high performance IGP**
- Routers monitor neighbor routers and networks and send this information to all OSPF routers (Link State Advertisements, **LSA**) using **flooding**
- LSA are only sent when changes occur
- Neighbor routers are monitored using a **hello protocol**
- OSPF routers maintain a **LS database**. The **Shortest Path First** algorithm is used to build routing table entries
- The **metric**: computed using link bitrates, delays etc
- There is no **convergence** (count to infinity) problem

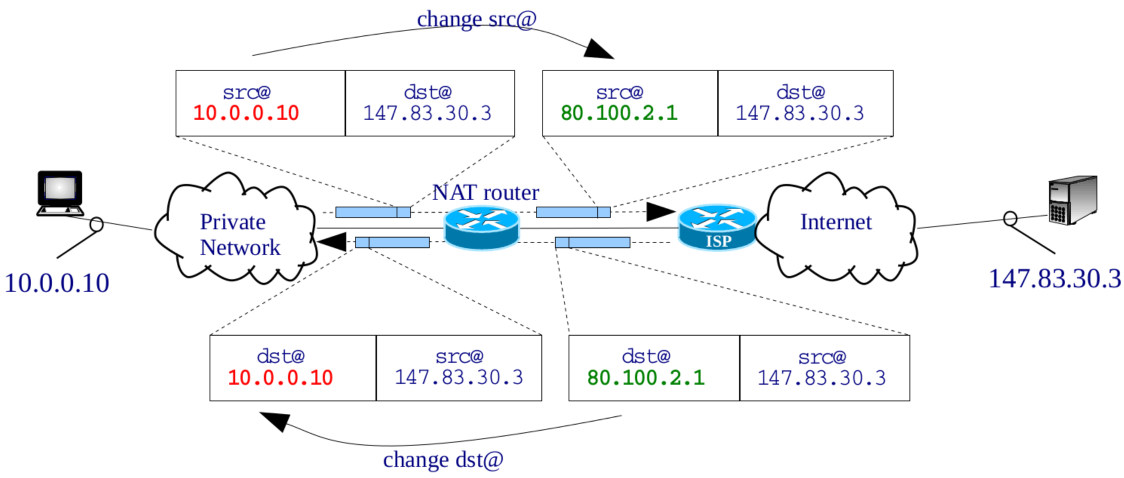
2.10 Network Address Translation NAT URL

2.10.1 Motivation

- Save **public** addresses
- Security

2.10.2 How it works

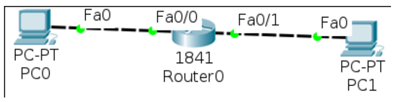
- A **NAT table** is used for address mapping



2.10.3 Types of NAT

- **Basic** NAT
  - public address <-> private address
- **Dynamic** NAT
  - pool of public addresses dynamically allocated
- **Port and Address Translation, PAT** (or **PNAT**)
  - One public address shared by many connections
  - NAT table must store **ports** to distinguish connections
  - NAT table must have one entry for **each connection**
- **DNAT**
  - Like NAT, but connections initiated from an external clients
  - Requires **static** configuration

Practical example



packettracer

NAT with **packettracer** (IOS):

NAT configuration in IOS (shell)

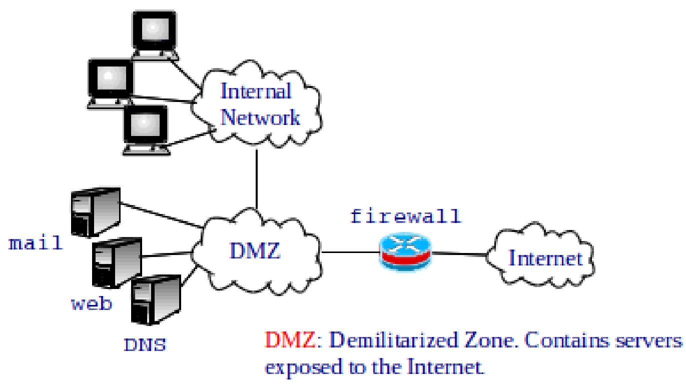
```
Router#sh running-config
interface FastEthernet0/0
ip nat inside
!
interface FastEthernet0/1
ip nat outside
!
! PAT
access-list 1 permit 192.168.0.0 0.255.255.255
ip nat inside source list 1 interface FastEthernet0/0 overload
! DNAT
ip nat inside source static tcp 192.168.0.1 80 200.0.0.1 80

Router#show ip nat translations
Pro  Inside global      Inside local        Outside local       Outside global
tcp 200.0.0.1:80        192.168.0.1:80      ---                 ---
```

2.11 Security in IP

- Objectives
  - **Confidentiality**: Who can access
  - **Integrity**: Who can modify the data
  - **Availability**: Access guarantee
- Basic solutions
  - **Firewalls**
  - **Virtual Private Networks** (VPN)

2.11.1 Basic firewalls

- Packet filtering based on IP/TCP/UDP header rules

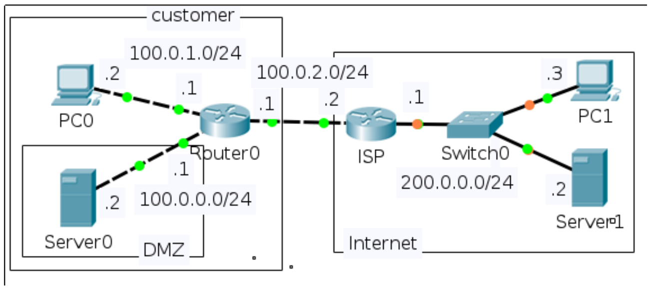


2.11.2 Basic Firewall Configuration

- **NAT**
- Access Control List, **ACL**

Practical example

- Basic **IOS commands**
  - **access-list** #acl {deny|permit} {protocol} {@IP source WildcardMask | host @IP source | any} [operador port source] {@IP dest WildcardMask | host @IP dest | any} [operador port dest] [established]
  - **ip access-group** #acl {in |out}



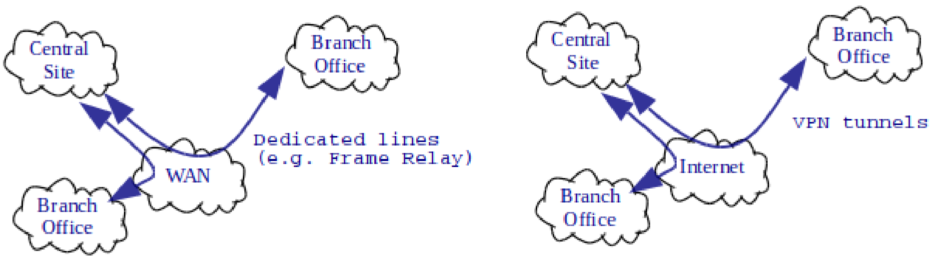
packettracer

ACLs in packettracer **packettracer** (IOS):

ACLs in packettracer (shell)

```
 Router#sh running-config
...
interface FastEthernet0/0
 ip address 100.0.2.1 255.255.255.0
 ip access-group 100 in
!
access-list 100 permit tcp any gt 1023 host 100.0.0.2 eq 80
access-list 100 permit icmp any host 100.0.0.2
access-list 100 permit tcp any lt 1024 100.0.1.0 0.0.0.255 gt 1023
```

2.11.3 Virtual Private Network, VPN

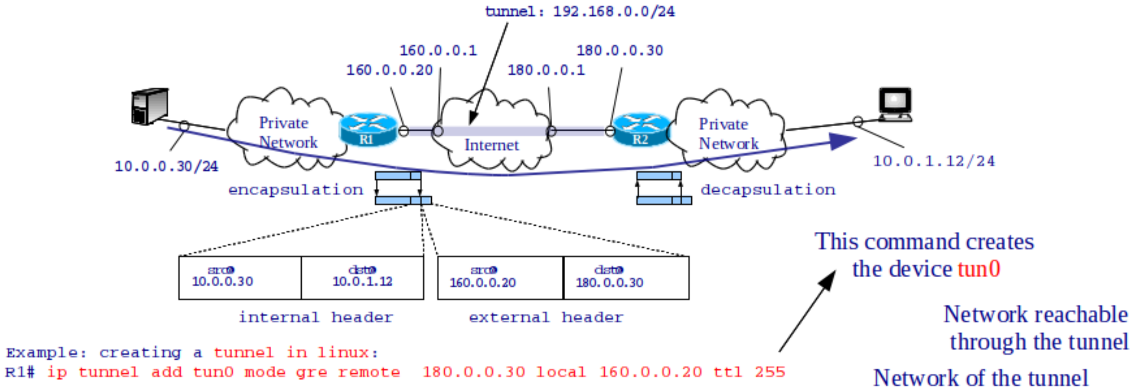


VPN vs **Conventional PN**

- less cost
- more flexible
- simple management
- Internet availability

2.11.4 VPN Ingredients

- Authentication
- Cryptography
- Tunneling



2.11.5 Tunneling issues

- **Fragmentation**: destination in the external header is the tunnel exit, this router should reassemble fragments!,
- Source in the external header is the tunnel entry => **ICMP** messages are set to the tunnel entry => MTU path discovery would not work!
- **Solution**:
  - tunnel pseudo-interface maintains a **tunnel state** e.g. the **tunnel MTU**. **ICMP** messages are sent by the tunnel entry router

2.11.6 Practical examples

ip tunnel

ip tunnel (bash)

```
/sbin/ifconfig
sudo ip tunnel add tunprova mode ipip remote 10.0.0.1 local <ip-wlan0>
ip tunnel show
/sbin/ifconfig -a
sudo /sbin/ifconfig tunprova 192.168.0.1 netmask 255.255.255.0
sudo /sbin/route add -net 10.1.0.0 netmask 255.255.255.0 gw 192.168.0.2
/sbin/route -n
sudo tcpdump -vni
ping 10.1.0.1
```

openvpn https://openvpn.net howto

openvpn https://openvpn.net (bash)

```
sudo openvpn client.ovpn
/sbin/ifconfig
sudo tcpdump -ni tun0
netstat -at
tcp        0      0 192.168.7.2:41446       vpn.ac.upc.es:openvpn   ESTABLISHED
sudo tcpdump -ni wlan0 port openvpn
```