Computer Networks. Unit 2: IP

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

Llorenç Cerdà-Alabern

March 2, 2021

Contents

2 Unit 2: IP

2.1 IP Protocol RFC791

2.1.1 Who run the protocol

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

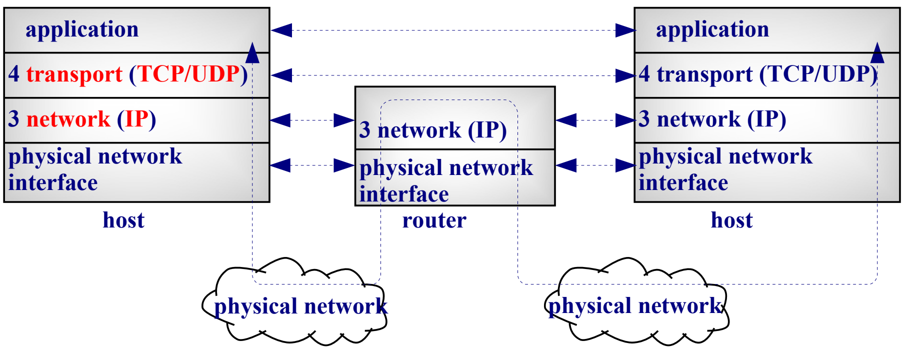

Figure 1: Architecture of the Internet: IP is a network layer protocol.

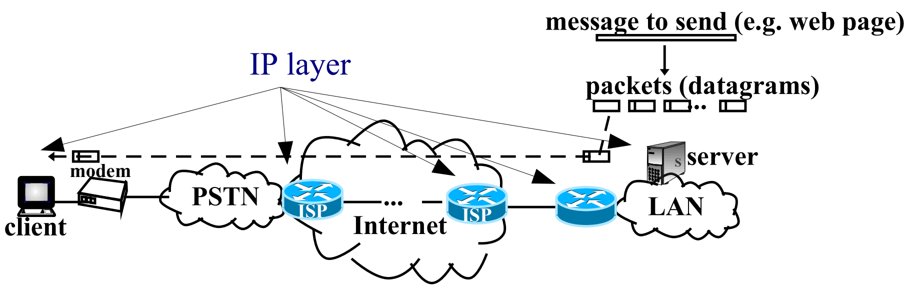

Figure 2: The Internet is an interconnection of hosts and routers, all running the IP protocol to exchange IP datagrams.

2.1.2 IP Service URL

- Properties of the **service offered by IP**
  1. **Connectionless**
  2. **Stateless**
  3. **Best effort**

These properties are a consequence of **how a router works**

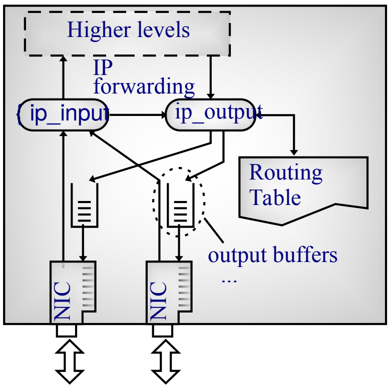

Figure 3: Router architecture: consists of network interfaces, NICs. Arriving datagrams are processed by `ip_input`. If the router is not the final destination, datagrams are forwarded to `ip_oput`, which use a routing table to decide the NIC to reach the next hop. Datagrams pending to be transmitted are stored in the NIC buffer.

- Packets can be discarded if:
  - No space is left on the buffer (congestion),
  - Destination unreachable,
  - Security (firewall),
  - etc.

2.1.3 IPv4 Header RFC791

Datagram (layer 3 packet in TCP/IP)

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

Figure 4: An IP datagram consist of the header followed by the payload, both of variable size.

```
 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      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: The IP header consists of the fields (bits): Version(4), IP header length(4), type of service(8), total length(16), identification(16), flags(3), offset(13), time to live(8), protocol(8), checksum(16), source address(32), destination address(32) and options(variable).

Practical example (bash)

```bash
sudo wireshark
ping 8.8.8.8
```

Figure 6: Observe the IP datagram header with wireshark.

2.2 IPv4 Addresses

2.2.1 netid/hostid

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

```
 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           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: IPv4 address is 32 bits (4 bytes) and consists of a netid followed by a hostid of variable size.

- **netid**: identify a network
- **hostid**: identify a host in a network

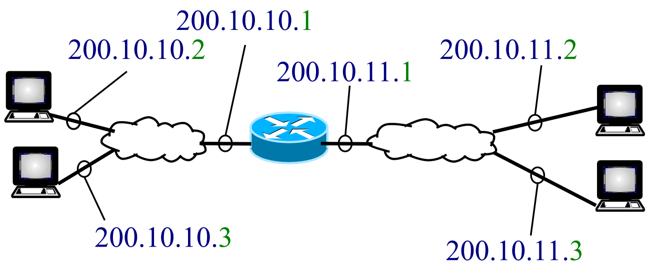

Figure 8: All hosts and router interfaces in the same IP network have the same netid, and different hostid. Routers interconnect different IP networks.

- IPv4 exhaustion: What is IPv4 Run Out?
  - **IPv6** is the long-term solution
- **IPv6** addresses (**128 bits, 16 bytes**) are written in hexadecimal URL
  - eight groups of 2 bytes in hex
  - e.g. **fe80::16fe:b5ff:feee:dd6b**
  - **::** means "blocks of zeros"

2.2.2 Assigment

- IP addresses in **the Internet** must be **unique**
- Internet Assigned Numbers Authority, IANA assign IP addresses to **Regional Internet Registries**, RIR:
  - RIPE: Europe
  - AFRINIC: Africa
  - ARIN: USA
  - APNIC: ASIA and Australia
  - LACNIC: Latin America
- RIR assign IP addresses to **ISPs**, (**Local Internet Registries**, LIR)
- **ISPs** assign IP addresses to their customers

Acronyms:

ISP	Internet Service Provider
LIR	Local Internet Registry
NIC	Network Information Center
RIR	Regional Internet Registry

Practical examples

Practical example (bash)

```
> whois 147.83.34.1
inetnum:        147.83.0.0 - 147.83.255.255
descr:          Universitat Politecnica de Catalunya
source:         RIPE
```

Figure 9: URL whois protocol for querying registered IP databases RFC3912.

Practical example (bash)

```
> ping ftp.au.debian.org # Australia
PING mirror.linux.org.au (150.203.164.37) 56(84) bytes of data.
64 bytes from linux.anu.edu.au (150.203.164.37): icmp_seq=1 ttl=42 time=300 ms
^C
> traceroute 150.203.164.37 # routers to reach the server
> whois 150.203.164.37 # whois
inetnum:        150.203.0.0 - 150.203.255.255
address:        IIS, Australian National University
address:        Canberra ACT 0200 Australia
source:         APNIC
```

Figure 10: Ping around the world, observe the round trip time URL.

2.2.3 IPv4 address classes

- Mechanism defined in the start of the Internet to know the **number of bits of the netid/hostid**
- Every IPv4 address belongs to a particular class
- Number of 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**
- **Most Significant bits** (MSB) identify the class

Number of **bytes** in netid/hostid:

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~

Figure 11: In every row of the table there is one class and the corresponding MSB

Acronyms:
MSB: Most Significant Bits

2.2.4 IPv4 address assignment

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

```
 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            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: IP address consists of a netid prefix, and a hostid

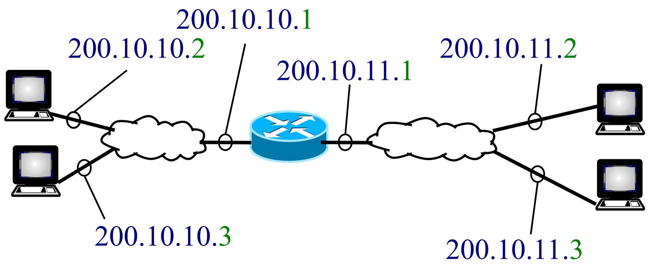

Figure 13: Example of IP address assignment: there are 2 IP networks. All interfaces in the same IP network have the same netid and a different hostid.

Practical example (bash)

```
/sbin/ifconfig wlan0 # IPv4 and IPv6 addresses of the interface
```

Figure 14: The UNIX command `ifconfig` shows the IP address configured in each interface.

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

Figure 15: Table with IP addresses with special meanings which cannot be assigned to an interface.

Note that all IP networks have **2 special addresses** of the network range that cannot be assigned to an interface:

- Network address
- Broadcast address

Example: for the network with netid 200.10.10, the addresses 200.10.10.0 (**network address**) and 200.10.10.255 (**broadcast address**), cannot be assigned to an interface.

Practical examples (bash)

```
ping 127.0.0.1
/sbin/ifconfig wlan0 # and ping to broadcast address
```

Figure 16: Ping to a host loopback address and network broadcast address.

2.2.6 Private IPv4 Addresses RFC1918

- For private usage (not in the Internet)
- Not assigned to any RIR
- Not unique
- Non routable in the Internet

Class	Nets	Hosts	Addresses
A	1	2^{24}	**10**.0.0.0
B	16	2^{16}	**172.16**.0.0 ~ **172.31**.0.0
C	256	2^{8}	**192.168.0**.0 ~ **192.168.255**.0

Figure 17: Table with the private IP addresses.

Acronyms:
RIR Regional Internet Registry

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**

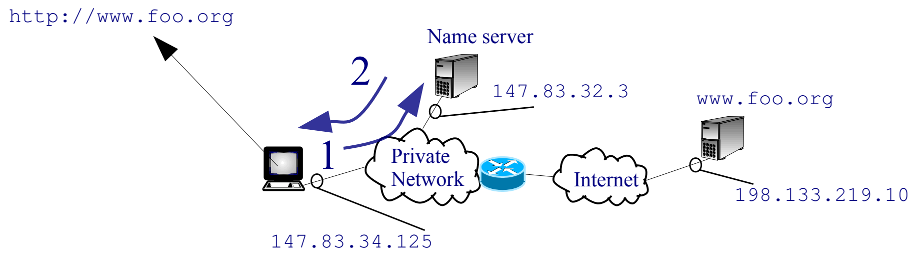

Figure 18: A host exchange a DNS request and DNS reply message with its name server for each name resolution.

Practical example (bash)

```
nslookup # www.upc.edu
tcpdump -ni wlan0 port 53 # capture DNS Request & Reply
```

Figure 19: Capture of the DNS Request & Reply messages generated with the `nslookup` command.

2.3 Subnetting RFC950

2.3.1 Motivation

- Split a large network into smaller ones

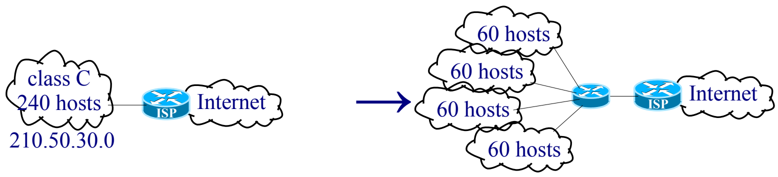

Figure 20: An IP network is split into 4 smaller IP networks connected to a router.

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

Figure 21: Example of an IP address and mask in dotted notation and binary.

Practical example (bash)

```
/sbin/ifconfig wlan0 # observe netmask
```

Figure 22: The UNIX command `ifconfig` shows the IP address, mask and broadcast address.

2.3.3 Variable Length Subnet Mask

- Consists of subnets of different mask length
- **Example**: subnetting a **class C** address:
  - **Base address**: address before subnetting
  - **subnetid**: bits borrowed from base address' hostid
  - /24 ⇒ We have 1 byte for subnetid + hostid
  - Subnetid is green
  - Chosen subnets addresses are underlined

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

Figure 23: Subnetting example: split the base address into 2 subnets /25. Then split one of them into 2 subnets /26. Then split one of them into 4 subnets /28.

  - Base address/24 ⇒ subnets: 1 x /25 + 1 x /26 + 4 x /28

- **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.
  - Define **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.191	B.191	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

Figure 24: Subnetting example. Every row corresponds to a subnet and shows: the subnetid in binary, the IP network address, the IP addresses that belong the the subnet, the broadcast address and the number of IP addresses.

2.3.4 Classless Inter-Domain Routing, CIDR RFC1519

- **Classless** routing: use masks instead of address classes
- Rational **geographical-based** distribution of IP addresses
- Facilitate the router address **aggregation** in the routing table

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
- One aggregation scheme (used in RIP, studied later) is:
- **Summarization:** aggregation at a class boundary

Summarization example Class C addresses are summarized to /24:

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

Acronyms:
RIP Routing Information Protocol

2.4 Routing Table (RT)

2.4.1 Who use the routing table?

- **IP layer** in hosts and routers use a RT
- **ip$_{\text{output}}$**() 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** matches the whole IPv4 address space

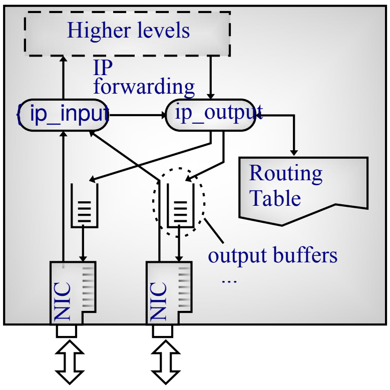

Figure 25: Router architecture: consists of network interfaces, NICs. Arriving datagrams are processed by `ip_input`. If the router is not the final destination, datagrams are forwarded to `ip_oput`, which use a routing table to decide the NIC to reach the next hop. Datagrams pending to be transmitted are stored in the NIC buffer.

2.4.2 What's in the RT?

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

Practical examples

Practical example (bash)

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

Figure 26: Observe the routing table of a UNIX host with the command `route`.

Public BGP route servers. Also known as looking glass servers: allow unauthorized access with the purpose of viewing routing information. List of looking glass servers:

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

Practical example (bash)

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

Figure 27: Observe the BGP routing table in a public BGP route server.

```
route-views>show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is 128.223.51.1 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 128.223.51.1
      1.0.0.0/8 is variably subnetted, 2955 subnets, 17 masks
B        1.0.0.0/24 [20/0] via 202.232.0.2, 1w1d
B        1.0.4.0/22 [20/0] via 114.31.199.1, 2d11h
...
```

Figure 28: Example of the routing table of a public BGP route server.

2.4.3 Datagram Delivery Algorithm

Datagram Delivery Algorithm (c)

```
1. if(IP.destination == address any interf.) {
    send to loopback interface
   }
2. for(RT = each routing table entry
     ordered from longest to shortest netid)
   /* Longest Prefix Match */ {
      if((IP.destination & RT.mask) == RT.destination) {
          return(RT.gateway, RT.interface) ;
      }
   }
3. if(RT.gateway == 0.0.0.0) { /* direct routing */
     send the datagram to IP.destination in RT.interface
   } else { /* indirect routing */
     send the datagram to RT.gateway in RT.interface
   }
```

Figure 29: Algorithm run by the IP layer every time an IP datagram is routed.

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

  Practical examples: adding static entries in the RT (UNIX)

Practical example (bash)

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

Figure 30: The UNIX command `route` allows adding and deleting entries from the routing table.

2.5 ARP protocol RFC826

2.5.1 Motivation

- Physical networks use addresses, e.g. Ethernet

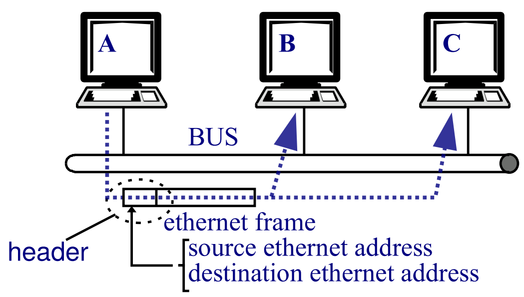

Figure 31: LANs, as Ethernet, use MAC addresses to identify the source and destination network interface card, NIC.

- IP layer pass a **physical address** to NIC driver
- IP calls **ARP** to obtain the physical addresses of a **unicast** IP address
- Physical addresses in LANs are called **MAC addresses**
- A **Broadcast IP** address does not need ARP, it is mapped to a **broadcast MAC** address

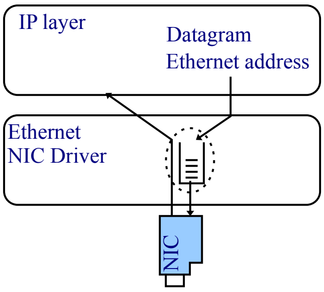

Figure 32: When the IP layer pass down to the NIC driver an IP datagram, it must provide the MAC destination address.

Acronyms:
MAC Medium Access Control
NIC Network Inferface Card

2.5.2 Address Resolution

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

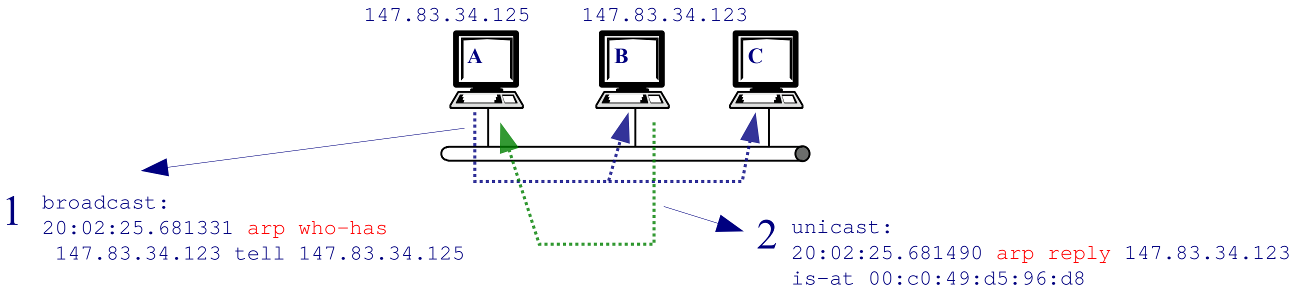

Figure 33: ARP resolution consists of an ARP broadcast request and a unicast reply messages.

ARP Fundamentals

- Encapsulated directly in **L2 frames** (no IP datagrams, no client/server paradigm)
- ARP Request: **broadcast** frame
- ARP Reply: **unicast** frame
- **Both devices** involved add an entry to the **ARP table**
- **ARP table** maps **IP** <-> **MAC** address
- ARP entries are removed after an **aging time**

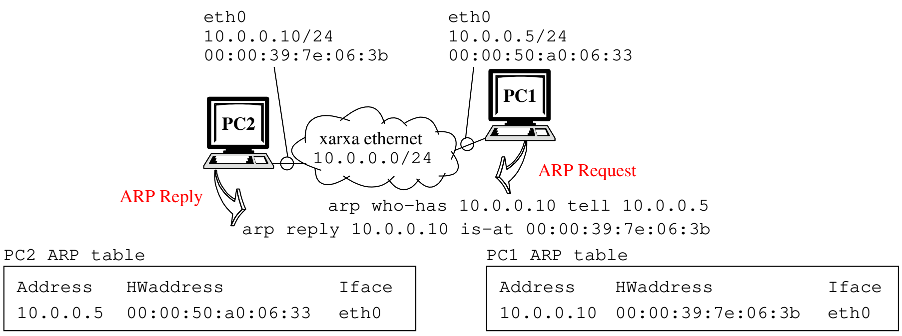

Figure 34: After an ARP resolution, ARP tables of both involved stations are updated with the IP and MAC addresses of the opposite station.

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)                            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 35: Format of ARP messages. Sender and target protocol and hardware addresses fields correspond to IP and MAC addresses of the sender and receiving stations, respectively.

Practical examples

Practical example (bash)

```bash
sudo wireshark
/usr/sbin/arp -n # show ARP table
```

Figure 36: Capture an ARP resolution with wireshark, and observe the ARP table with the arp command.

Exercise: ARP resolution in a ping broadcast.

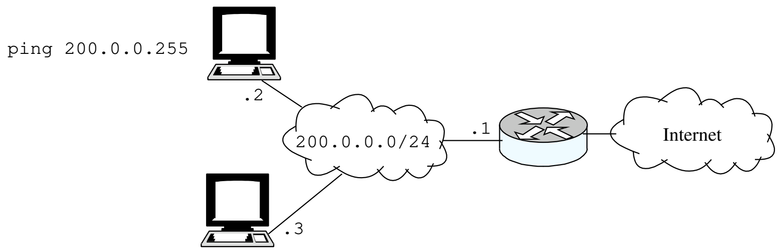

Figure 37: A network has 2 hosts and 1 router. From one of the hosts it is executed a ping to the network broadcast address.

- Show the packets that will be exchanged in the network. Hint: the devices responding the ping message will initiate the ARP resolution.

Solution

packet	eth.src	eth.dst	IP.src	IP.dst	descrip.
1	eth.2	FF	IP.2	200.0.0.255	ICMP echo request broadcast
2	eth.1	FF	-	-	ARP request who as IP.2?
3	eth.2	eth.1	-	-	ARP reply IP.2 is at eth.2
4	eth.1	eth.2	IP.1	IP.2	ICMP echo reply
5	eth.3	FF	-	-	ARP request who as IP.2?
6	eth.2	eth.3	-	-	ARP reply IP.2 is at eth.2
7	eth.3	eth.2	IP.3	IP.2	ICMP echo reply

Figure 38: Table with all packets that will be exchanged, showing MAC and IP addresses. Every row correspond to one transmitted packet.

2.5.3 Gratuitous ARP

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

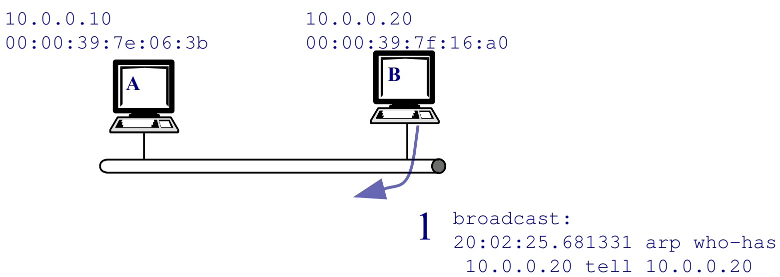

Figure 39: Gratuitous ARP: a host sends an ARP request of his own IP address.

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 condition
- An ICMP error message cannot generate another ICMP error message

2.6.2 ICMP message format

- **type** determines the format of the remaining data
- **code** distinguish messages of equal type
- In **error** messages:
  - 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      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 40: ICMP general message format. Type/Code fields identify the ICMP message. Error ICMP messages have a field with the IP header plus the first 8 bytes of the payload of the datagram generating the error. This information is used to process the error message at the receiving station.

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 port unkown
	4	error	fragmentation needed but 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`

Figure 41: Table with common ICMP messages.

Practical examples (wireshark)

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

2.7 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    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 42: IP header consists of a fixed 20 bytes fields, and a variable size options fields.

- **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

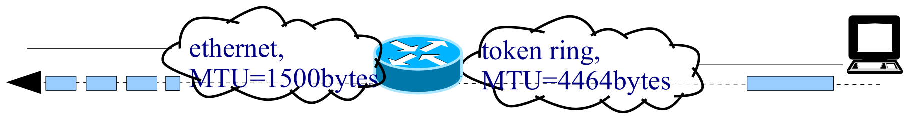

Figure 43: A datagram is fragmented in multiple smaller datagrams.

Fragmentation may occur:

- Router: Fragmentation may be needed when two networks with different **Maximum Transfer Unit (MTU)** are connected
- Host: may be needed if **UDP** datagrams of size larger than the MTU

Practical example (bash)

```
sudo ifconfig lo mtu 1500
./udpserver.rb
sudo wireshark -ki lo
```

Figure 44: Observe the IP fragmentation of an UDP datagram.

Practical example (ruby)

```
require 'socket'

server = UDPSocket.new
server.connect("127.0.0.1", 2000)
server.send("1".ljust(5000, "1"), 0) # send message
print "done"
```

Figure 45: Simple client that sends a UDP datagram of 5000 bytes, which is fragmented.

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)

```
 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    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 46: IP header showing the identification, flags and offset fields.

Example

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

UDP header is 8 bytes Network MTU is 1500 bytes

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

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
  - RFC2132 (options)

2.8.2 DHCP Fundamentals

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

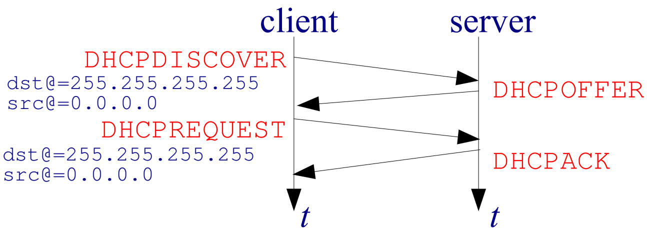

Figure 47: Time diagram showing the exchange of DHCPDISCOVER/DHCPOFFER and DHCPREQUEST/DHCPACK messages.

- NOTES:
  - Client 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

Practical example (bash)

```
sudo wireshark
ps aux | egrep dhclient
sudo dhclient -r # DHCPRELEASE (release configuration)
sudo dhclient -i wlan0 # DHCPDISCOVER-OFFER-REQUEST-ACK
sudo dhclient -i wlan0 # DHCPREQUEST-ACK
```

Figure 48: Capture DHCP messages with wireshark.

2.9 Routing Algorithms

2.9.1 What is a routing algorithm?

Objective: initialize routing tables

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

2.9.2 What is an Autonomous Systems (AS)?

- Internet is organized in *Autonomous Systems* (**AS**):
  - Created to facilitate routing in the Internet
  - Group a set of **IP prefixes** (blocks of public IP addresses)
  - 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

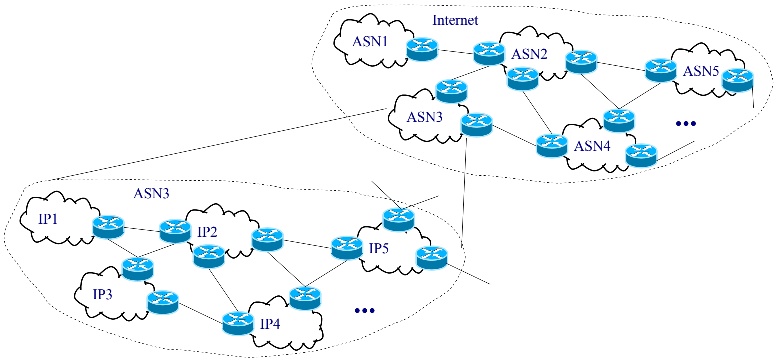

Figure 49: Shows the partition of the Internet in autonomous systems.

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 **commercial agreements**
  - **BGP basis**: routers exchange IP prefixes/AS paths/attributes

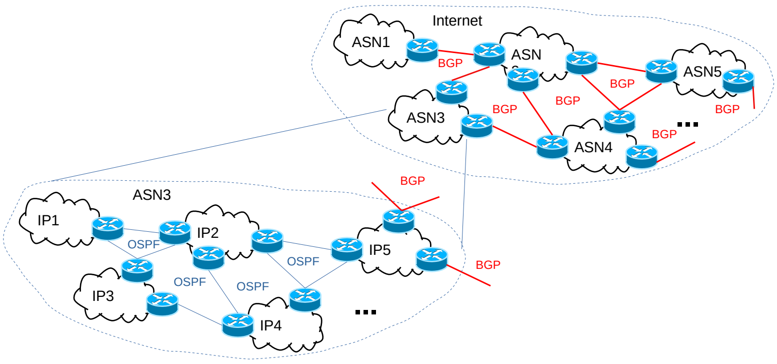

Figure 50: BGP is the routing protocol among ASs. Inside an AS there can be multiple IGP routing protocols.

2.9.4 Routing Information Protocol, RIP RFC2453

(**only routing protocol we will study in detail**)

- **Metric**: number of hops (networks)
- **Metric = 1** for directly connected 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** (destination unreachable) is **16**
- **Route Summarization**: aggregation to class
  - 192.168.0.0/25+192.168.0.128/25->192.168.0.0/24
  - Summarization is done when the update is sent into a network that belongs to a different base address
- RIP **version 2**:
  - allows variable masks
  - multicast dst. 224.0.0.9

Updates

- RIP **updates** include **destinations** and **metrics**
- Upon receiving an update change the routing table if:
  - There is a better path (lower metric)
  - The gateway being used change the metric
  - There is a new route

Example: Routing table of **R1**

Destination	Gateway	metric
N1	-	1
N2	R2	2
N3	R2	3
N4	R3	4

Figure 51: Routing table of R1. Each row is a destination/gateway/metric entry.

- Upon **R1** receiving the update from **R2** (Dest., metric) **[(N2, 3), (N3, 2), (N4, 2), (N5, 2)]**

Destination	Gateway	metric
N1	-	1
N2	R2	**4**
N3	R2	3
N4	**R2**	**3**
N5	**R2**	**3**

Figure 52: Changes on the R1 routing table upon receiving the RIP update [(N2, 3), (N3, 2), (N4, 2), (N5, 2)] from R2.

Count to Infinity

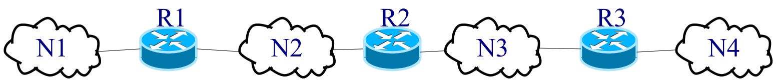

Figure 53: Simple linear network topology where networks N1 to N4 are connected with routers R1 to R3.

- 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

Figure 54: Routing tables when RIP has converged.

- 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

Figure 55: Possible evolution of the entry D=N4 in the routing tables of R1 and R2 when R3 fails.

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

Practical example

- RIP with packettracer
- Notes from the **routing table in IOS**:
  - [120/1]: [administrative distance/route metric]
  - **admin. distance**: reliability of a routing protocol (the lower the distance the more reliable). RIP=120, OSPF=110
  - **route metric of RIP in the IOS RT**: number of hops - 1
    - Note that in the RIP updates metric = number of hops
  - If there are multiple routes to the same destination with the same metric, IOS does **load balancing**

Practical example (shell)

```
Router(config-router)# router rip        ! configure RIP daemon
Router(config-router)# network a.b.c.d   ! export network
```

Figure 56: Basic IOS RIP configuration commands.

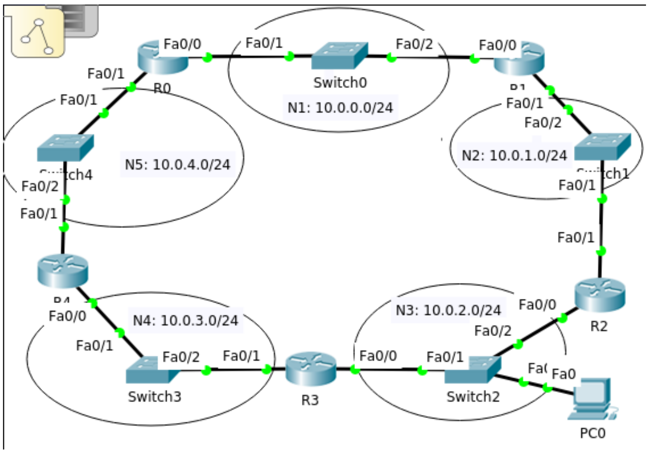

Figure 57: Example of 5 routers and 5 networks with circular topology, configured using RIP with packettracer.

```
R0#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 5 subnets
C       10.0.0.0 is directly connected, FastEthernet0/0
R       10.0.1.0 [120/1] via 10.0.0.2, 00:00:19, FastEthernet0/0
R       10.0.2.0 [120/2] via 10.0.0.2, 00:00:19, FastEthernet0/0
                 [120/2] via 10.0.4.2, 00:00:22, FastEthernet0/1
R       10.0.3.0 [120/1] via 10.0.4.2, 00:00:22, FastEthernet0/1
C       10.0.4.0 is directly connected, FastEthernet0/1
```

Figure 58: Routing table of router R0 when RIP has converged.

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 bitrate, delays etc
- There is no **convergence** (count to infinity) problem

Acronyms:

IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
LS	Link State
LSA	Link State Advertisements
OSPF	Open Shortest Path First

2.10 Network Address Translation NAT URL

2.10.1 Motivation

- Save **public** IP addresses
  - Many private IP address can access the Internet using a single public IP address
- **Security**
  - Internal network using private IP address is not reachable from the Internet

2.10.2 How it works

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

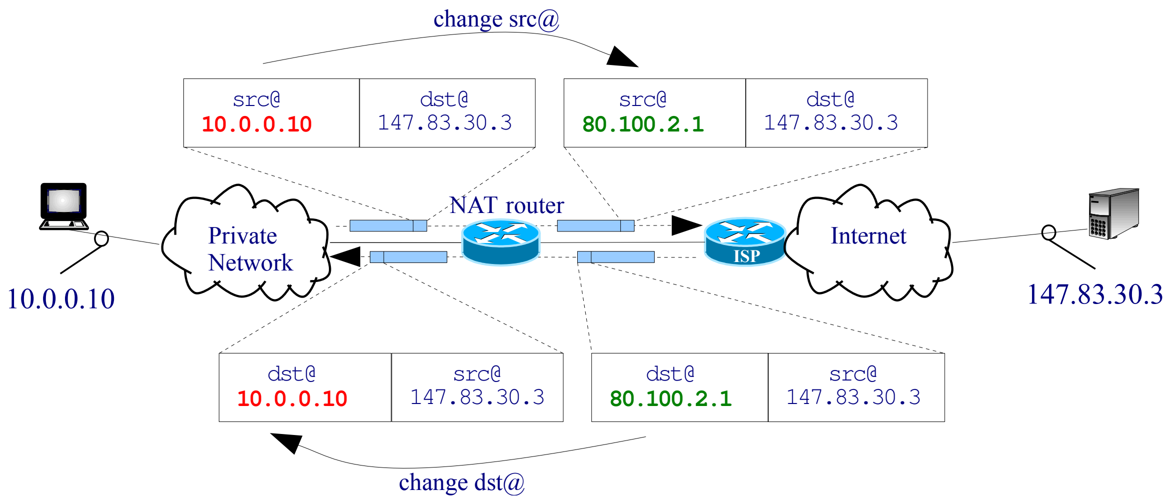

Figure 59: Example of NAT.

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
  - port might need to be **translated** to avoid collisions
  - for **ICMP** echo-request/reply **Query ID** is used instead of port **Query ID** is a 16 bits integer used to match echo-request/reply
  - NAT table must have one entry for **each connection**
- **DNAT**
  - Like NAT, but connections initiated from an external clients
  - Requires **static** configuration

Practical example

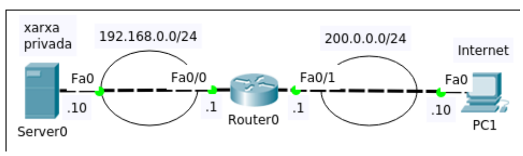

Figure 60: Simple network topology server/router/host with packettracer.

NAT with **packettracer** (IOS):

Practical example (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 192.168.0.10 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     ---                ---
```

Figure 61: NAT configuration in IOS.

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 Virtual Private Network, VPN

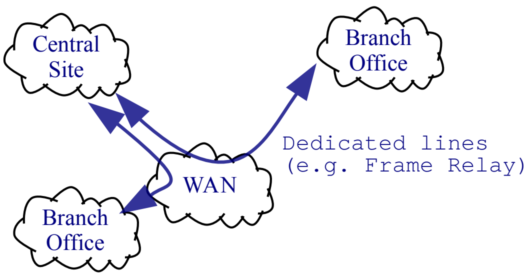

Figure 62: Conventional Private Network

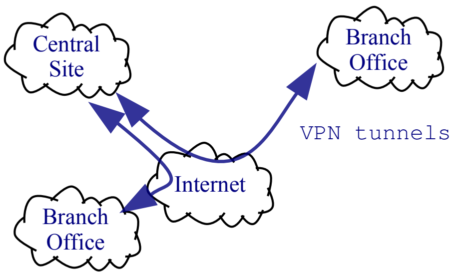

Figure 63: Virtual Private Network

VPN vs **Conventional PN**

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

VPN Ingredients

- **Authentication**: identify authorized parties
- **Encryption**: only authorized parties can access
- **Tunneling**: emulate dedicated lines between networks

How a tunnel works

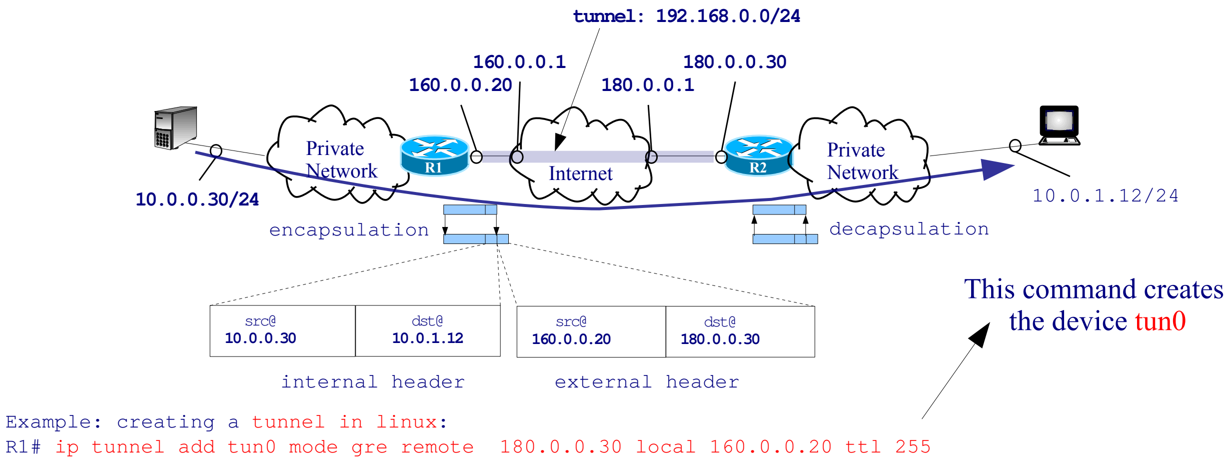

Figure 64: Example of an IPIP tunnel configuration in linux.

Destination	Gateway	Genmask	Iface
10.0.0.0	0.0.0.0	255.255.255.0	eth0
160.0.0.1	0.0.0.0	255.255.255.255	ppp0
0.0.0.0	160.0.0.1	0.0.0.0	ppp0
192.168.0.0	0.0.0.0	255.255.255.0	tun0
10.0.1.0	192.168.0.2	255.255.255.0	tun0

Figure 65: R1 Routing Table

Destination	Gateway	Genmask	Iface
10.0.1.0	0.0.0.0	255.255.255.0	eth0
180.0.0.1	0.0.0.0	255.255.255.255	ppp0
0.0.0.0	180.0.0.1	0.0.0.0	ppp0
192.168.0.0	0.0.0.0	255.255.255.0	tun0
10.0.0.0	192.168.0.1	255.255.255.0	tun0

Figure 66: R2 Routing Table

Practical examples

ip tunnel

Practical example (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
ping 10.1.0.1
sudo wireshark ip.proto==ipip # observe IPIP encapsulation
sudo ip tunnel del tunprova
```

Figure 67: Capture IP datagrams sent over an IPIP tunnel and observe external and internal IP headers with wireshark.

openvpn https://openvpn.net howto

Practical example (bash)

```
sudo openvpn client.ovpn
/sbin/ifconfig
sudo tcpdump -ni tun0
netstat -nat
tcp        0      0 147.83.34.125:39796     147.83.30.75:1194       ESTABLISHED
sudo wireshark tcp.port==1194 # observe the tunnel through an encrypted TCP
↪  connection
```

Figure 68: Observe the TCP connection created as a tunnel by open-vpn, https://openvpn.net.

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

2.11.2 Basic firewalls

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

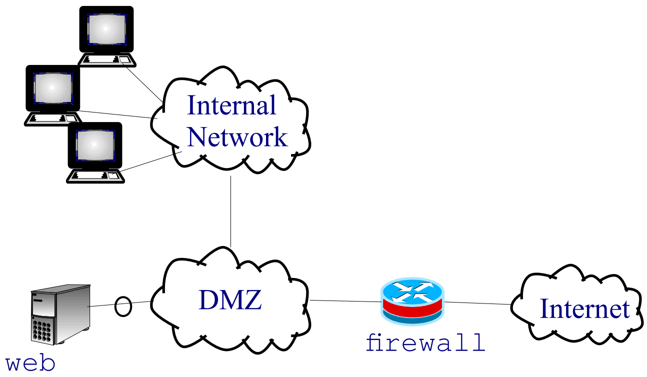

Figure 69: Basic premises network organization with an internal network and DMZ.

Basic Firewall Configuration

- **NAT**
- Access Control List, **ACL**
  - Ordered list of **rules** applied to all packets entering or leaving a router interface
  - Every rule **match** a condition on the IP/TCP/UDP header
  - Upon match apply an **action** (accept/deny) and leave the list
  - Typically there is a last rule **deny all** discarding all packets not accepted by a previous rule

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}

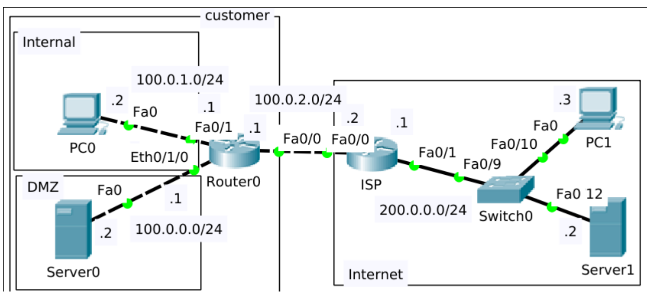

Figure 70: Example of ACL configuration with packettracer.

Practical example (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 any
access-list 100 permit tcp any lt 1024 100.0.1.0 0.0.0.255 gt 1023
```

Figure 71: ACLs configuration with IOS.

2.12 List of Acronyms

ACL	Access Control List
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CIDR	Classless Inter-Domain Routing
DHCP	Dynamic Host Configuration Protocol
DMZ	DeMilitarized Zone
DNS	Domain Name System
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
LIR	Local Internet Registry
LSA	Link State Advertisements
MAC	Medium Access Control
MSB	Most Significant Bits
NAT	Network Address Translation
NIC	Network Inferface Card
NIC	Network Information Center
OSPF	Open Shortest Path First
PDU	Protocol Data Unit
PN	Private Network
RFC	Request For Comments
RIP	Routing Information Protocol
RIR	Regional Internet Registry
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VLSM	Variable Length Subnet Mask
VPN	Virtual Private Network
WAN	Wide Area Network