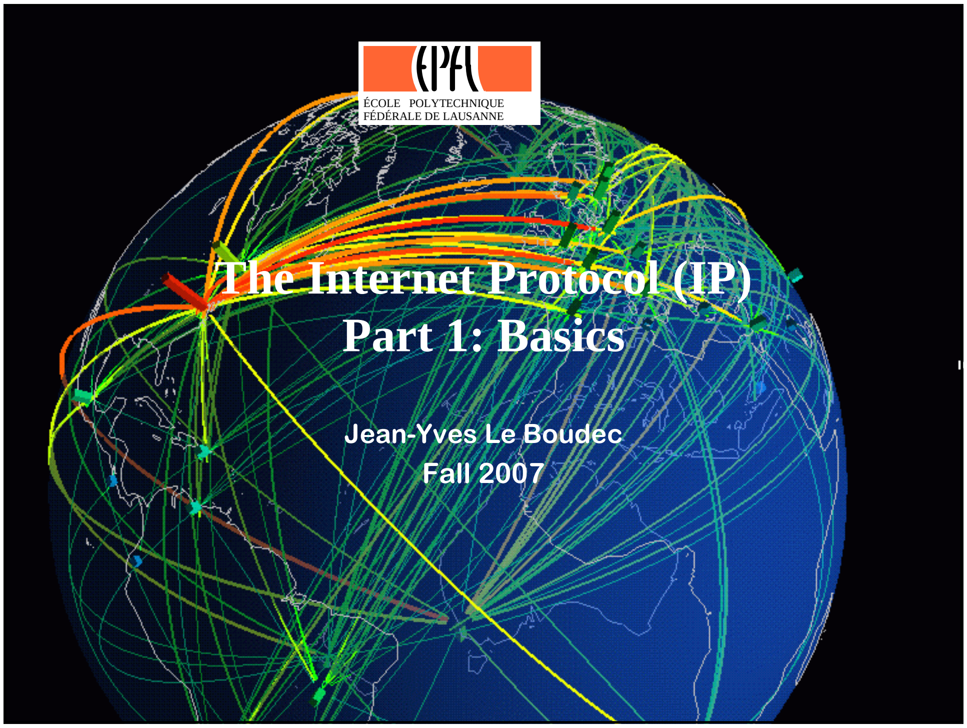


ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# The Internet Protocol (IP) Part 1: Basics

Jean-Yves Le Boudec

Fall 2007

## Contents

# 1. Why a network layer?

- We would like to interconnect all devices in the world. We have seen that we can solve the interconnection problem with bridges and the MAC layer. However this is not sufficient as it does not *scale* to large networks.

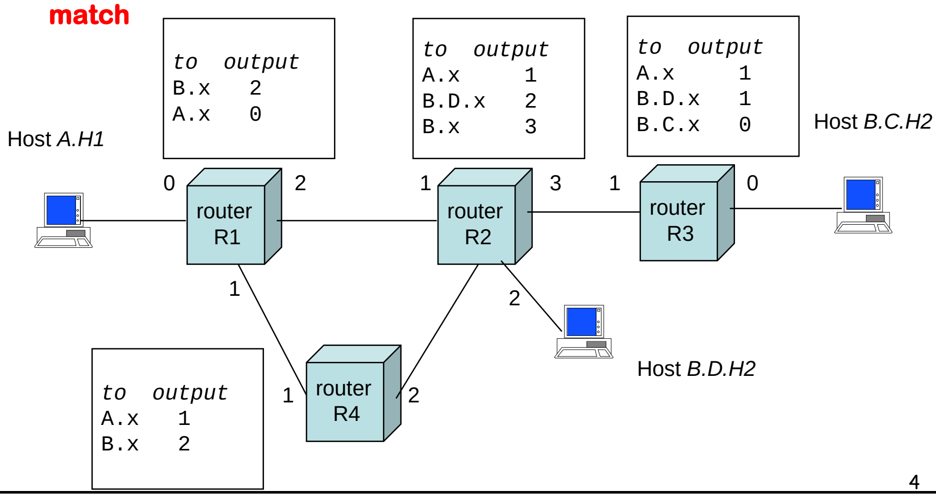
  **Q.** Why ? solution

- Solution: connectionless network layer (eg. Internet Protocol, IP):
  - every host receives a network layer address (IP address)
  - intermediate systems forward packets based on destination address



# Connectionless Network Layer

- **Connectionless** network layer = no connection
- every packet contains destination address
- intermediate systems ( = routers) forward based on **longest prefix match**



# IP Principles

**Homogeneous addressing**

- an IP address is unique across the whole network (= the world in general)
- IP address is the address of an interface
- communication between IP hosts requires knowledge of IP addresses

**Routers between subnetworks only**:

- a subnetwork = a collection of systems with a common prefix
- inside a subnetwork: hosts communicate directly without routers
- between subnetworks: one or several routers are used

- Host either sends a packet to the destination using its LAN, or it passes it to the router for forwarding

- Terminology:
  - host = end system; router = intermediate system
  - subnetwork = one collection of hosts that can communicate directly without routers

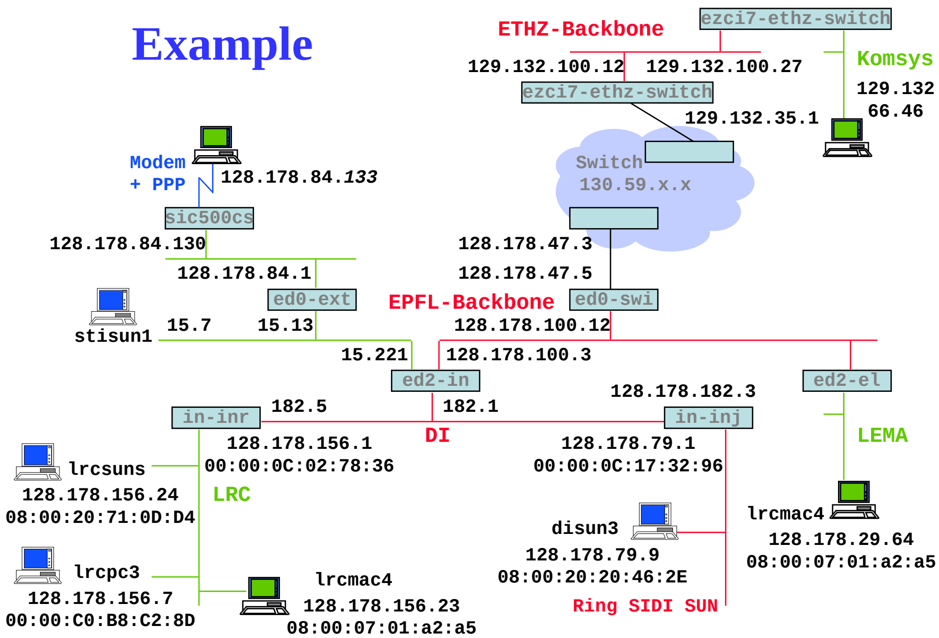


# 2. IP addresses

- IP address
  - Unique addresses in the world, decentralized allocation
  - An IP address is 32 bits, noted in dotted decimal notation: `192.78.32.2`

- Host and Prefix Part
  - An IP address has a prefix and a host part:
    - `prefix:host`
  - Prefix identifies a subnetwork
    - used for locating a subnetwork – routing
  - Prefix is usually identified in a host using a “subnet mask”

# Example






# Binary, Decimal and Hexadecimal

- Given an integer B “the basis”: any integer can be represented in “base B” by means of an alphabet of B symbols
- Usual cases are
  - decimal: 234
  - binary: b1110 1010
  - hexadecimal: xEA
- Mapping binary <-> hexa is simple: one hexa digit is 4 binary digits
  - xE = b1110 xA = b1010 xEA= b1110 1010
- Mapping binary <-> decimal is best done by a calculator
  - b1110 1010 = 128 + 64 + 32 + 8 + 2 = 234
- Special Cases to remember
  - xF = b1111 = 15
  - xFF = b1111 1111 = 255

# Representation of IP Addresses

- **dotted decimal**: group bits in bytes, write the decimal representation of the number
  - example 1: `128.191.151.1`
  - example 2: `129.192.152.2`
- **hexadecimal**: hexadecimal representation -- fixed size string
  - example 1: `x80 BF 97 01`
  - example 2: `x`
- **binary**: string of 32 bits (2 symbols: 0, 1)
  - example 1: `b0100 0000 1011 1111 1001 0111 0000 0001`
  - example 2: `b`

solution



# A Subnet Prefix is written using one of two Notations: masks / prefixes

- Using a mask: address + mask :
  - example : 128.178.156.13 mask 255.255.255.0
    - the mask is the dotted decimal representation of the string made of : 1 in the prefix, 0 elsewhere
    - bit wise address & mask gives the prefix
    - here: prefix is 128.178.156.0
  - example 2: 129.132.119.77 mask 255.255.255.192
    - Q1: what is the prefix ?
    - Q2: how many host ids can be allocated ?

solution

# Prefix Notation

- prefix – notation: 128.178.156.1/24
  - the 24 first bits of the binary representation of the string, interpreted as dotted decimal
  - here: the prefix is 128.178.156.0
  - bits in excess are ignored
    - 128.178.156.1/24 is the same as 128.178.156.22/24 and 128.178.156/24
- example 2:
  - Q1: write 129.132.119.77 mask 255.255.255.192 in prefix notation
  - Q2: are these prefixes different ?
    - 201.10.0.00/28, 201.10.0.16/28, 201.10.0.32/28, 201.10.0.48/28
    - how many IP addresses can be allocated to each of the distinct subnets ?

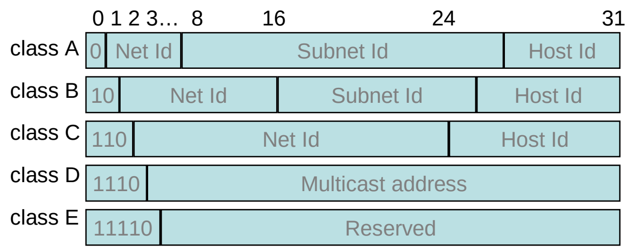
solution



# IP Address Hierarchies

- The prefix of an IP address can itself be structured into subprefix in order to support aggregation
  - For example: 128.178.x.y represents an EPFL host
    128.178.156 / 24 represents the LRC subnet at EPFL
    128.178 / 16 represents EPFL
  - Used between routers by routing algorithms
  - This way of doing is called classless and was first introduced in inter domain routing under the name of CIDR (classless interdomain routing)
- IP address classes
  - IP addresses are sorted into classes
  - This is an obsolete classification – no longer used
    - At the origin, the prefix of an IP address was defined in a very rigid way. For class A addresses, the prefix was 8 bits. For class B, 16 bits. For class C, 24 bits. The interest of that scheme was that by simply analyzing the address you could find out what the prefix was.
    - It was soon recognized that this form was too rigid. Then subnets were added. It was no longer possible to recognize from the address alone where the subnet prefix ends and where the host identifier starts. For example, the host part at EPFL is 8 bits; it is 6 bits at ETHZ. Therefore, an additional information, called the subnet mask, is necessary.
    - Class C addresses were meant to be allocated one per network. Today, they are allocated in contiguous blocks.

# IP address classes




Examples: 128.178.x.x = EPFL host; 129.132.x.x = ETHZ host
9.x.x.x = IBM host 18.x.x.x = MIT host

| Class | Range |
|---|---|
| A | 0.0.0.0 to 127.255.255.255 |
| B | 128.0.0.0 to 191.255.255.255 |
| C | 192.0.0.0 to 223.255.255.255 |
| D | 224.0.0.0 to 239.255.255.255 |
| E | 240.0.0.0 to 247.255.255.255 |

- Class B addresses are close to exhausted; new addresses are taken from class C, allocated as continuous blocks



# Address allocation

- World Coverage
  - Europe and the Middle East (RIPE NCC)
  - Africa (ARIN & RIPE NCC)
  - North America (ARIN)
  - Latin America including the Caribbean (ARIN)
  - Asia-Pacific (APNIC)
- Current allocations of Class C
  - `193-195/8, 212-213/8, 217/8` for RIPE
  - `199-201/8, 204-209/8, 216/8` for ARIN
  - `202-203/8, 210-211/8, 218/8` for APNIC
- Simplifies routing
  - short prefix aggregates many subnetworks
  - routing decision is taken based on the short prefix

## Address delegation

- Europe
  - 62/8, 80/8, 193-195/8, …
  - ISP-1
    - 62.125/16
    - customer 1: banana foods
      - 62.125.44.128/25
    - customer 2: sovkom
      - 62.125.44.50/24
  - ISP-2
    - 195.44/14
    - customer 1:
      - 195.46.216/21
    - customer 2:
      - 195.46.224/21

solution

**Q.** Assume sovkom moves from ISP-1 to ISP-2; comment on the impact.



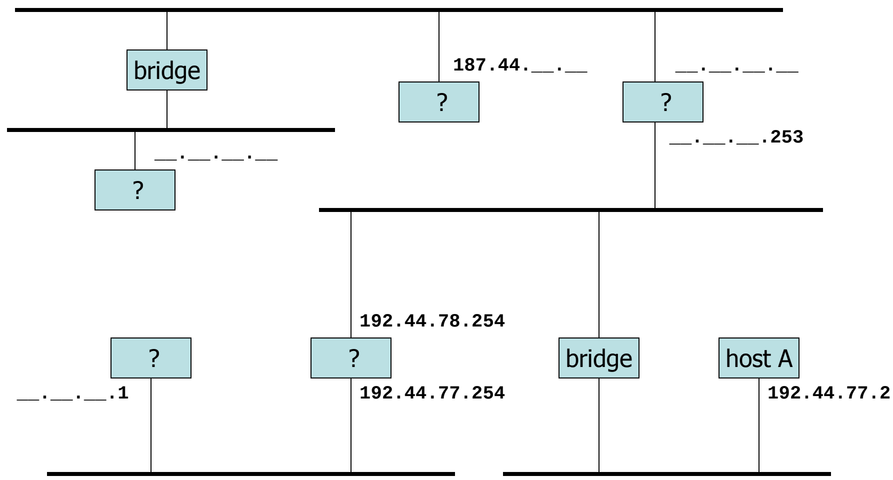
## Special case IP addresses

| | |
|---|---|
| 1. 0.0.0.0 | this host, on this network |
| 2. 0.hostId | specified host on this net (initialization phase) |
| 3. 255.255.255.255 | limited broadcast (not forwarded by routers) |
| 4. subnetId.all 1's | broadcast on this subnet |
| 5. subnetId.all 0's | BSD used it for broadcast on this subnet (obsolate) |
| 6. 127.x.x.x | loopback |
| 7. 10/8<br>172.16/12<br>192.168/16 | reserved networks for internal use (Intranets) |

- 1,2: source IP@ only; 3,4,5: destination IP@ only

# Test Your Understanding (1)

solution




Q: Can host A have this address? (masks are all 255.255.255)



# Test your Understanding (2)

- **Q1: An Ethernet segment became too crowded; we split it into 2 segments, interconnected by a router. Do we need to change some IP host addresses?**
- **Q2: same with a bridge.**
- **Q3: compare the two**

**solutions**

# 3. IP packet forwarding

The IP packet forwarding algorithm is the core of the TCP/IP architecture. It defines what a system should do with a packet it has to send or forward. The rule is simple :

- Rule for sending packets (hosts, routers)
  - if the destination IP address has the same prefix as one of my interfaces, send directly to that interface
  - otherwise send to a router as given by the IP routing table

It uses the IP routing table; the table can be checked with a command such as "netstat" with Unix or "Route" with Windows.

In reality, there are exceptions to the rule. The complete algorithm is in the next slide; the cases should be tested in that order (it is a nested **if then else** statement).



# IP packet forwarding algorithm

destAddr = destination address /* unicast! */

**if /*case 1*/**: a **host route** exists for destAddr
for every entry in routing table
if (destinationAddr = destAddr)
then send to nextHop IPaddr; leave

**else if /*case 2*/**: destAddr is on a **directly connected network** (= on-link):
for every physical interface IP address A and subnet mask SM
if(A & SM = destAddr & SM)
then send directly to destAddr; leave

**else if /*case 3 */** there is a **matching entry in routing table**
find the **longest prefix match** for destAddr
send to nextHop IP addr given by matching entry; leave
/* this includes as special case the default route, if it exists */

**else /* error*/**
send ICMP error message "destination unreachable" to source

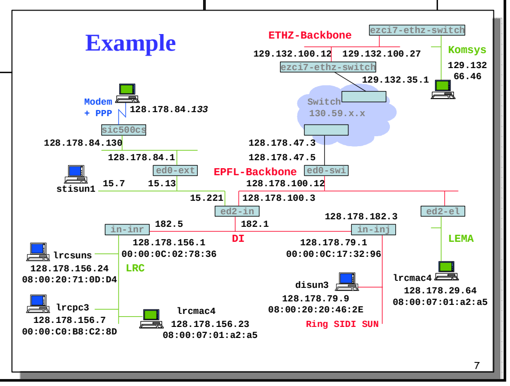
# Example

- Q1: Fill in the table if an IP packet has to be sent from **lrcsuns**

| final destination | next hop | case number |
|---|---|---|
| 128.178.79.9 | | |
| 128.178.156.7 | | |
| 127.0.0.1 | | |
| 128.178.84.133 | | |
| 129.132.1.45 | | |

- Q2: Fill in the table if an IP packet has to be sent from ed2-in

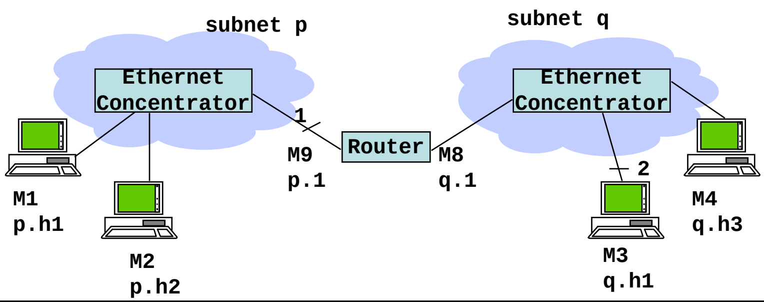
solutions




# Test Your Understanding (3)

- **Q1.** What are the MAC and IP addresses at points 1 and 2 for packets sent by M1 to M3 ? At 2 for packets sent by M4 to M3 ?(Mx = mac address)

solution

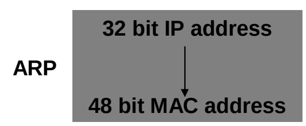


# Direct Packet Forwarding: ARP

- ❑ Sending to host on the same subnet = direct packet forwarding
  - does not use a router
- ❑ Requires the knowledge of the MAC address on a LAN
  (called “physical” address)

  There are four types of solutions for that; all exist in some form or another.
  1. write arp table manually: can always be implemented manually on Unix or Windows NT using the arp command
  2. Derive MAC address algorithmically from IP address. This requires that the MAC address fits in the IP address; it is used with IPv6 but not with the current version of IP.
  3. Write the mappings MAC <-> IP in a server (used in special cases like ATM or frame relay).
  4. Use a discovery protocol by broadcast. This is done on all LANs (Ethernet, WiFi).

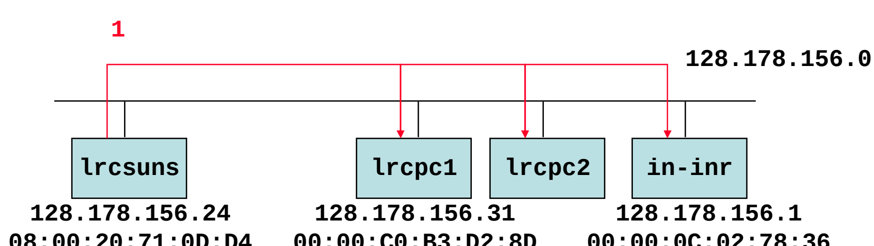
  - on LANs: uses the Address Resolution Protocol





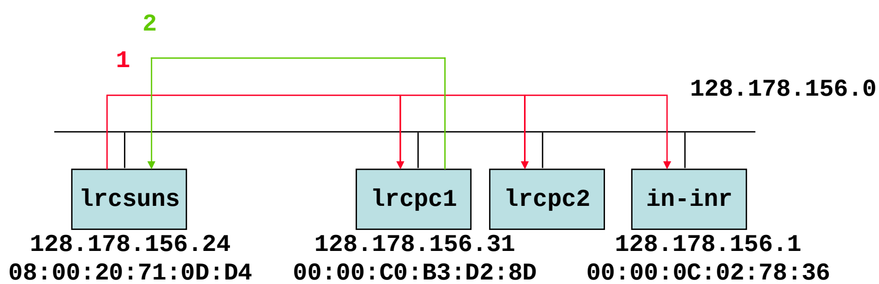

# ARP Protocol

- ❑ 1: `lrcsuns` has a packet to send to `128.178.156.31` (`lrcpc1`)




- this address is on the same subnet
- `lrcsuns` sends an ARP request to all systems on the subnet (broadcast)
- target IP address = `128.178.156.31`
- ARP request is received by all IP hosts on the local network
- is not forwarded by routers

# ARP Protocol

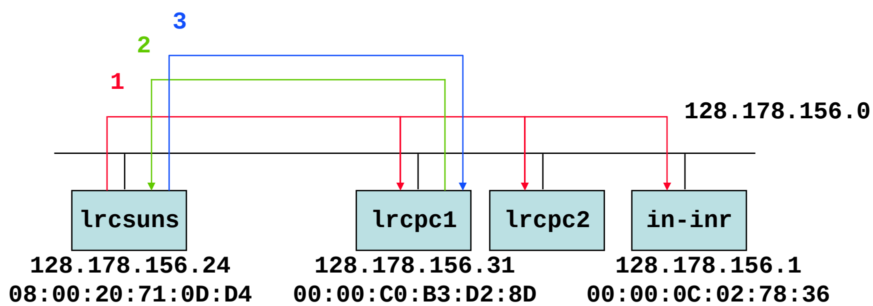



2: `lrcpc1` has recognized its IP address

- sends an ARP reply packet to the requesting host
- with its IP and MAC addresses



# ARP Protocol




3: `lrcsuns` reads ARP reply, stores in a cache and sends IP packet to `lrcpc1`

Systems learn from ARP-REQUESTs. At the end of flow 1, all systems have learnt the mapping IP <-> MAC addr for the source of the ARP REQUEST, namely, they have updated the following entry in their ARP table:

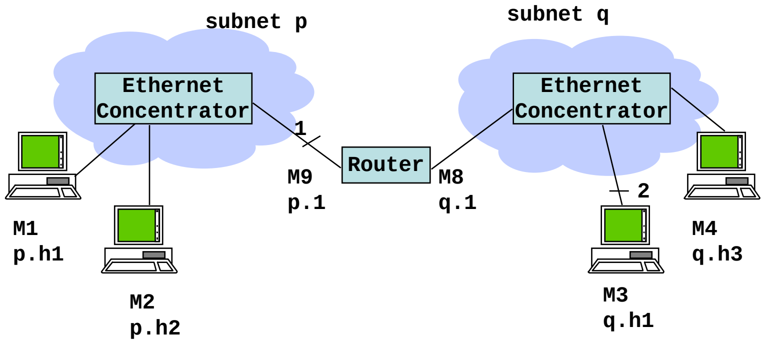
IP addr: 128.178.156.24
MAC addr: 08:00:20:71:0D:D4.

As a result, lrcpc1 will not send an ARP-REQUEST to communicate back with lrcsuns. Gratuitous ARP consists in sending an ARP-REQUEST to self's address. This is used at bootstrap to test the presence of a duplicate IP address. It is also used to force ARP cache entries to be changed after an address change (because systems learn from the ARP-REQUEST). As flow 2 shows, the ARP-REPLY is not broadcast, but sent directly to the system that issued the request. The "arp" command on Unix can be used to see or modify the ARP table.

# Test Your Understanding (3, cont'd)

- Q2: What must the router do when it receives a packet from M2 to M3 for the first time?

solution





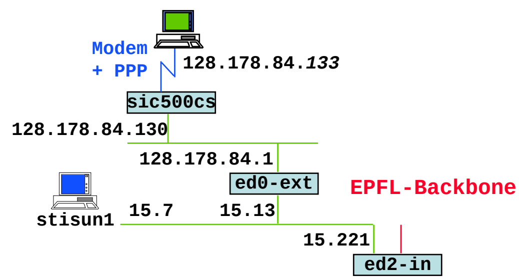

# Look inside an ARP packet

```
Ethernet II
    Destination: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source: 00:03:93:a3:83:3a (Apple_a3:83:3a)
    Type: ARP (0x0806)
    Trailer: 00000000000000000000000000000000...
Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: 00:03:93:a3:83:3a (Apple_a3:83:3a)
    Sender IP address: 129.88.38.135 (129.88.38.135)
    Target MAC address: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Target IP address: 129.88.38.254 (129.88.38.254)
```

# Proxy ARP

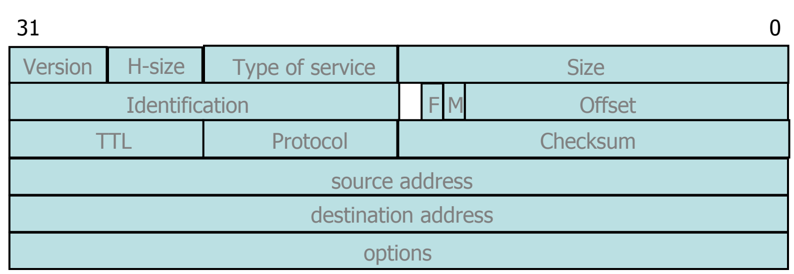
- ❑ Proxy ARP = a host answers ARP requests on behalf of others
  - example: `sic500cs` for PPP connected computers
  - Allows to *cheat*: connect to different physical networks that have same subnet prefix
  - Price to pay: ad-hoc configuration + single point of failure
- ❑ Q1: how must sics500cs routing table be configured ?
- ❑ Q2: explain what happens when ed2-in has a packet to send to 128.178.84.133

  solution






# 4. IP header




- ❑ Transmitted "big-endian" - bit 31 first
  - Version is always 4 (IPv6 uses a different packet format)
  - Header size
    - options - variable size
    - in 32 bit words

# IP header

- Type of service
  - Previously used to encode priority;
  - now used by DiffServ (Differentiated Services)
  - 1 byte codepoint determining QoS class
    - Expedited Forwarding (EF) - minimize delay and jitter
    - Assured Forwarding (AF) - four classes and three drop-precedences (12 codepoints)
  - Used only in corporate networks
- Packet size
  - in bytes including header
  - ≤ 64 Kbytes; limited in practice by link-level MTU (Maximum Transmission Unit)
  - every subnet should forward packets of 576 = 512 + 64 bytes
- Id
  - unique identifier for re-assembling
- Flags
  - M : more ; set in fragments
  - F : prohibits fragmentation
- Offset
  - position of a fragment in multiples of 8 bytes
- TTL (Time-to-live)
  - in seconds
  - now: number of hops
  - router : --, if 0, drop (send ICMP packet to source)
- Protocol
  - identifier of protocol (1 - ICMP, 6 - TCP, 17 - UDP)
- Checksum
  - only on the header



# IP Checksum

- The IP checksum is a simple example of error detecting code. It works as follows. Consider a sequence of bytes and group them by 16-bit words. If the sequence has an odd number of bytes, add an extra 0 byte at the end. Obtain the 16 bits words $W_0$ to $W_j$. Consider the number $x = 2^{16\,j}\,W_j + 2^{16\,(j-1)}\,W_{j-1} + \ldots + 2^{16}\,W_1 + W_0$

  The checksum is $y = (2^{16} - 1) - z$ with

  $z = x \bmod (2^{16} - 1)$

  The computation of y is algorithmically simple. Note that $2^{16} = 1 \bmod (2^{16} - 1)$ and thus

  $z = W_j + W_{j-1} + \ldots + W_1 + W_0 \bmod (2^{16} - 1)$

  The algorithm is:

  compute $z = W_j + W_{j-1} + \ldots + W_1 + W_0$

  group the result by blocks of 16 bits; obtain $x' = 2^{16\,j'}\,W'_{j'} + 2^{16\,(j'-1)}\,W'_{j'-1} + \ldots + 2^{16}\,W'_1 + W'_0$

  start again with x' instead of x

  until z is a 16 bit word

- Comments:
  - Addition modulo $(2^{16} - 1)$ is called « one's complement addition »
  - The method is the same as the « proof by 9 » used by scholars before calculators existed, with 9 replaced by $2^{16} - 1$;

    ex: 2345678 mod 9 = 2+3+4+5+6+7+8 mod 9 = 35 mod 9 = 3+5 mod 9 = 8
  - See RFC 1624 for how to do the computations in practice with 32 bit arithmetic.

# Examples of IP Checksums

**all numbers are written in hexa**

**data:** **0103 0012** **$W_1$=0103** **$W_0$= 0012**

**z =**

**checksum y =**

**data:** 0100 F203 F4F5 F6F7

z = 0100 + F203 + F4F5 + F6F7 =

checksum y =

solution

source: http://www.netfor2.com/checksum.html



# Verifying a Checksum

- **Destination receives $W_j$ … $W_0$ y**
  **If there is no error we should have: $W_j$ + … +$W_0$ + y = 0 mod ($2^{16}$–1)**
  **Destination computes the one's complement sum of the block including checksum and verifies if the result is 0 mod ($2^{16}$–1)**
- **Examples:**

**received block** **0103 0012 FEEA**
**verification:** **0103 + 0012 + FEEA = FFFF** √

**received block** 0100 F203 F4F5 F6F7 210E
**verification:** 0100 + F203 + F4F5 + F6F7 + 210E **= 2 FFFD**
2 + FFFD **= FFFF** √

# IP header Options

- Options
  - strict source routing
    - all routers
  - loose source routing
    - some routers
  - record route
  - timestamp route
  - router alert
    - used by IGMP or RSVP for processing a packet



# Look inside an IP packet

```
Ethernet II
    Destination: 00:03:93:a3:83:3a (Apple_a3:83:3a)
    Source: 00:10:83:35:34:04 (HEWLETT-_35:34:04)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 129.88.38.94 (129.88.38.94), Dst Addr: 129.88.38.241
   (129.88.38.241)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 1500
    Identification: 0x624d
    Flags: 0x04
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x82cf (correct)
    Source: 129.88.38.94 (129.88.38.94)
    Destination: 129.88.38.241 (129.88.38.241)
```

# Facts to Remember

- IP is a connectionless network layer
- IP addresses are 32 bit numbers
- One IP address per interface
- Routers scale well because they can aggregate routes
- Hosts on the Internet exchange packets with IP addresses



# Solutions

# 1. Why a network layer?

- We would like to interconnect all devices in the world. We have seen that we can solve the interconnection problem with bridges and the MAC layer. However this is not sufficient as it does *scale* to large networks.

  **Q.** Why ?
  **A.**

  1. Bridges use a tree. This is not efficient in a large network, as the tree concentrates all traffic.
  2. Bridges use forwarding tables that are not structured. A bridge must lookup the entire table for *every* packet. The table size and lookup time would be prohibitive.

- Solution: connectionless network layer (eg. Internet Protocol, IP):
  - every host receives a network layer address (IP address)
  - intermediate systems forward packets based on destination address

back



# Representation of IP Addresses

- **dotted decimal**: group bits in bytes, write the decimal representation of the number
  - example 1: `128.191.151.1`
  - example 2: `129.192.152.2`

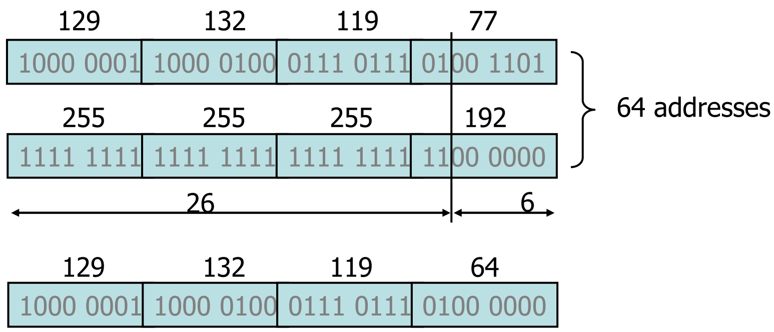
- **hexadecimal**: hexadecimal representation -- fixed size string
  - example 1: `x80 BF 97 01`
  - example 2: `x81 C0 98 02`

- **binary**: string of 32 bits (2 symbols: 0, 1)
  - example 1: `b0100 0000 1011 1111 1001 0111 0000 0001`
  - example 2: `b0100 0001 1100 0000 1001 1000 0000 0010`

back

# A Subnet Prefix is written using one of two Notations: masks / prefixes

- example 2: 129.132.119.77 mask 255.255.255.192
  - Q1: what is the prefix ? A: 129.132.119.64




  - Q2: how many host ids can be allocated ? A: 64 (minus the reserved addresses: 62)

back



# Prefix Notation

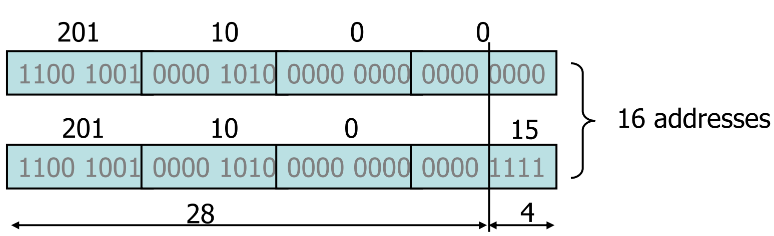
example 2:

- Q1: write 129.132.119.77 mask 255.255.255.192 in prefix notation
  A: 129.132.119.77/26 or 129.132.119.64/26
- Q2: are these prefixes different ?
  - 201.10.0.00/28, 201.10.0.16/28, 201.10.0.32/28, 201.10.0.48/28

  A: they differ in bits that are not the last 4 ones, thus they are all different prefixes

  - how many IP addresses can be allocated to each of the distinct subnets ?

  A: 14 (16 minus 2 reserved)




back

# Address delegation

- Europe
  - 62/8, 80/8, 193-195/8, …
  - ISP-1
    - 62.125/16
    - customer 1: banana foods
      - 62.125.44.128/25
    - customer 2: sovkom
      - 62.125.44.50/24
  - ISP-2
    - 195.44/14
    - customer 1:
      - 195.46.216/21
    - customer 2:
      - 195.46.224/21

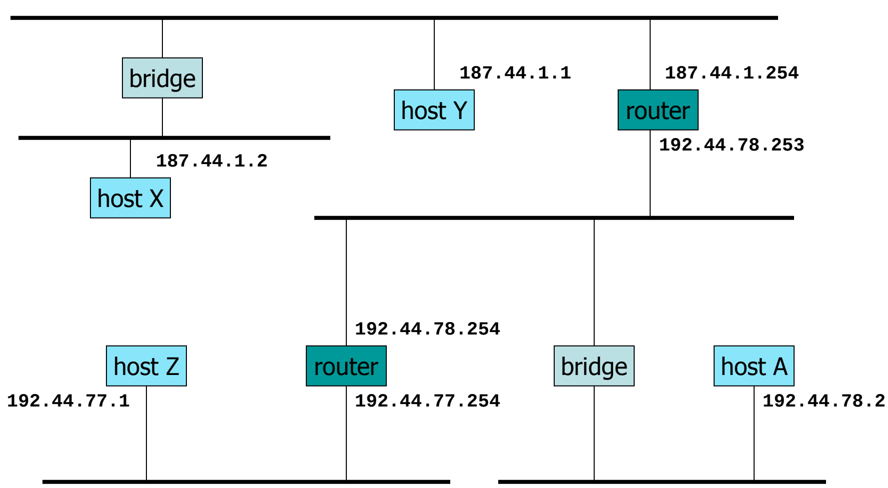
back

**Q.** Assume sovkom moves from ISP-1 to ISP-2; comment on the impact.
**A.** If sovkom keeps the same IP addresses, the set of addresses of ISP-2 is no longer continguous. It cannot be represented by one single entry in routing tables. Routing tables in the internet need to represent ISP-2 by two entries: 195.44/14 and 62.125.44.50/24



# Test Your Understanding (1)

back




- A: No, host A is on subnetwork **192.44.78**

# Test your Understanding (2)

- Q1: An Ethernet segment became too crowded; we split it into 2 segments, interconnected by a router. Do we need to change some IP host addresses ?
  A: yes in general. Two different subnets cannot have the same prefix
- Q2: same with a bridge
  A: no, bridging is transparent.
- Q3: compare the two
  A: bridging is plug and play but the network performance is more difficult to guarantee (broadcasts + spanning tree)

back



back

# Example

- Q: Fill in the table if an IP packet has to be sent from `lrcsuns`

| final destination | next hop | case number |
|---|---|---|
| 128.178.79.9 | 128.178.156.1 | 3 |
| 128.178.156.7 | 128.178.156.7 | 2 |
| 127.0.0.1 | loopback | 2 |
| 128.178.84.133 | 128.178.156.1 | 3 |
| 129.132.1.45 | 128.178.156.1 | 3 |

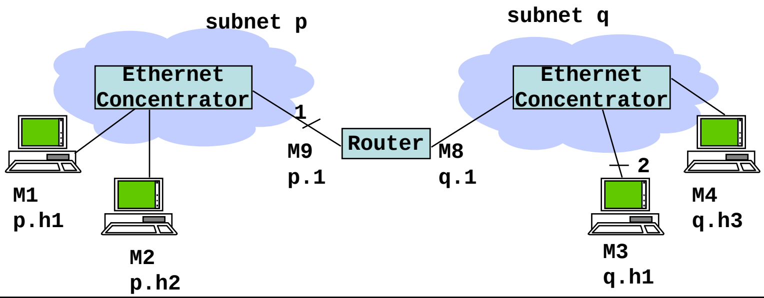
- Q: Fill in the table if an IP packet has to be sent from `ed2-in`

| final destination | next hop | case number |
|---|---|---|
| 128.178.79.9 | 128.178.182.3 | 3 |
| 128.178.156.7 | 128.178.182.5 | 3 |
| 127.0.0.1 | loopback | 2 |
| 128.178.84.133 | 128.178.15.13 | 3 |
| 129.132.1.45 | 128.178.100.12 | 3 |

## Test Your Understanding (3)

back

- Q1: What are the MAC and IP addresses at points 1 and 2 for packets sent by M1 to M3 ? At 2 for packets sent by M4 to M3 ?(Mx = mac address)
  A: at 1: srce IP@=p.h1, dest IP@=q.h1, MACsrce=M1, MACdest=M9
  at 2: srce IP@=p.h1, dest IP@=q.h1, MACsrce=M8, MACdest=M3
  at 2: srce IP@=q.h3, dest IP@=q.h1, MACsrce=M4, MACdest=M3

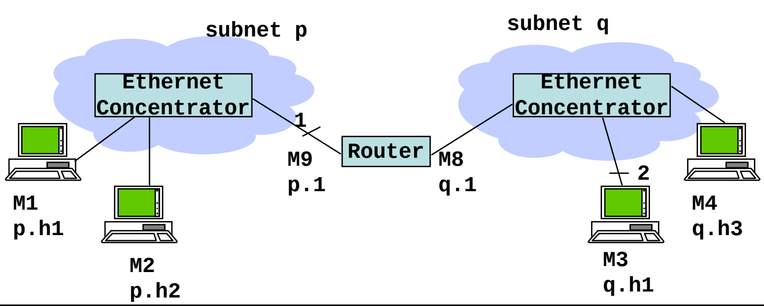





## Test Your Understanding (3)

- Q2: What must the router do when it receives a packet from M2 to M3 for the first time?
  A: send an ARP request broadcast on LAN q

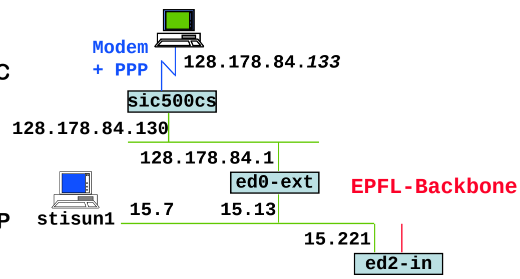
back



# Proxy ARP

- Q1: how must sics500cs routing table be configured ?
  - A: one host route per host such as 128.178.84.133
- Q2: explain what happens when ed2-in has a packet to send to 128.178.84.133
  - packet sent to ed0-ext
  - ARP sent by ed0-ext for target address = 128.178.84.133
  - sics500cs responds with MAC addr = sic500cs's MAC addr
  - packet sent ed0-ext to sic500cs
  - sic500cs reads host route and forwards to 128.178.84.133 (case 1 of IP forwarding algorithm)

back






# Examples of IP Checksums

**all numbers are written in hexa**

**data: 0103 0012 $W_1$=0103 $W_0$= 0012**

**z = 0103 + 0012 = 01 15**

**checksum y = FFFF – z = FEEA**

**data:** 0100 F203 F4F5 F6F7

z = 0100 + F203 + F4F5 + F6F7 = 0002 DEEF

z = 0002 + DEEF = DEF1

checksum y = FFFF - DEF1= 210E

back

source: http://www.netfor2.com/checksum.html