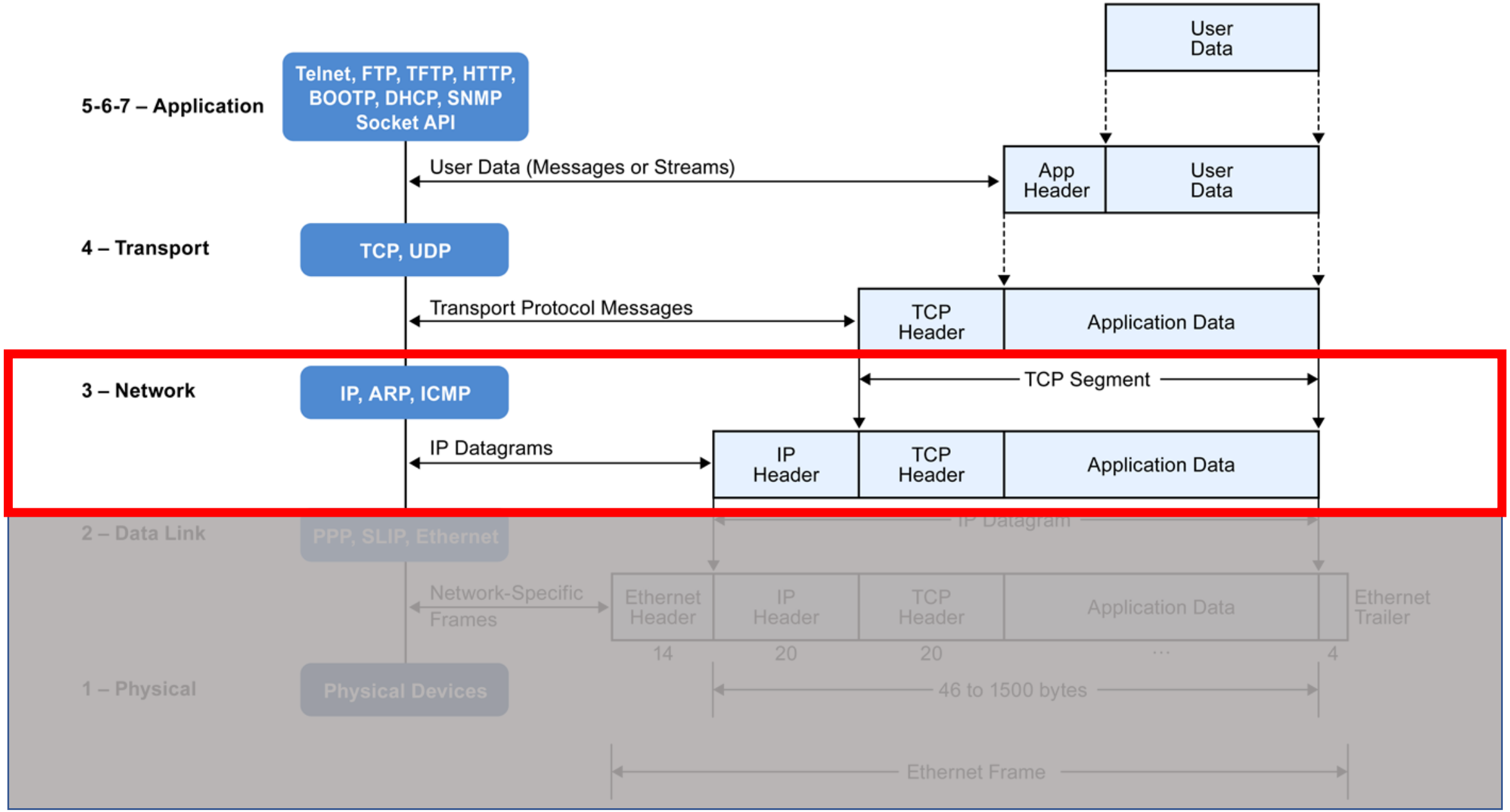
# Livello Internet (IP = Internet Protocol)


5-6-7 – Application
Telnet, FTP, TFTP, HTTP, BOOTP, DHCP, SNMP Socket API
User Data
User Data (Messages or Streams)
App Header
User Data
4 – Transport
TCP, UDP
Transport Protocol Messages
TCP Header
Application Data
3 – Network
IP, ARP, ICMP
TCP Segment
IP Datagrams
IP Header
TCP Header
Application Data
2 – Data Link
PPP, SLIP, Ethernet
IP Datagram
Network-Specific Frames
Ethernet Header
IP Header
TCP Header
Application Data
Ethernet Trailer
14
20
20
...
4
1 – Physical
Physical Devices
46 to 1500 bytes
Ethernet Frame

# Il Livello Internet

Come dice il nome stesso, questo livello è quello che ha permesso ad Internet di diventare quella che conosciamo oggi. Pensate a come sarebbe il mondo se all'improvviso sparisse Internet….

Vediamo prima i principi base che hanno ispirato la progettazione nel passato e reso Internet un successo ai nostri giorni.

Consiglio la lettura della RFC 1958 di Brian Carpenter in cui i principi base sono elencati

Vediamo in ordine di importanza decrescente questi principi.

Questi principi sono applicabili a molti altri contesti diversi da questo!

# 1: deve funzionare

- Siamo sicuri che funzioni: migliaia di prototipi devono aver funzionato l’uno con l’altro prima di essere sicuri che lo standard è stato progettato correttamente.
- Inutile scrivere 1000 pagine di standard per accorgersi di un errore e uscire poi con la versione 1.1 .

# 2: keep it simple

- KISS principle: Nel dubbio scegli la soluzione più semplice.
- In pratica: Non aggiungere un sacco di features. Se una feature non è essenziale lasciala fuori, soprattutto se puoi ottenere lo stesso effetto combinando altre features.

# 3: scelte chiare

- Se ci sono diversi modi per fare una cosa scegline uno e scarta gli altri.
- Avere due o più modi per fare la stessa cosa è andare in cerca di guai.

# 4: sfrutta la modularità

- Questo principio conduce all'idea di avere degli stacks di protocolli.
- Ogni layer indipendente o quasi dagli altri.
- Quando, a causa di nuovi requirements o nuove opportunità tecnologiche, devo cambiare un modulo o un layer, gli altri non vengono colpiti.

# 5: aspettati eterogeneità

- Diversi tipi di hardware, di strumenti trasmissivi, di applicazioni sono presenti in una rete molto grande.
- Per gestire tutti questi oggetti, essi devono avere una rete progettata in modo semplice, generale e flessibile.

# 6: evita parametri statici

- Se devo inserire un parametro, per esempio una dimensione massima di un pacchetto, è meglio che sender e receiver possano negoziare un valore piuttosto che ci siano delle scelte predefinite.

# 7: meglio buono o perfetto?

- Bisogna puntare ad un progetto buono, non deve essere perfetto.
- Spesso il progettista arriva ad un buon progetto ma non riesce a gestire degli strani casi particolari.
- Piuttosto che rovinare il progetto è meglio accontentarsi di quanto ottenuto e delegare la soluzione tecnica a chi arriva con gli strani casi particolari.

# 8: rigido o tollerante?

- Bisogna essere di stretta osservanza alle regole quando si trasmette e molto tolleranti quando si riceve.
- I pacchetti che si mandano devono rispettare strettamente gli standard.
- Bisogna aspettarsi che i pacchetti che si ricevono siano leggermente non standard e cercare di gestirli in qualche modo.

# 9: scalabilità

- Se il sistema deve gestire milioni di nodi e miliardi di utenti in modo efficiente, non ci deve essere un database centralizzato.
- Il carico deve essere distribuito nel modo più equilibrato possibile su tutte le risorse disponibili.

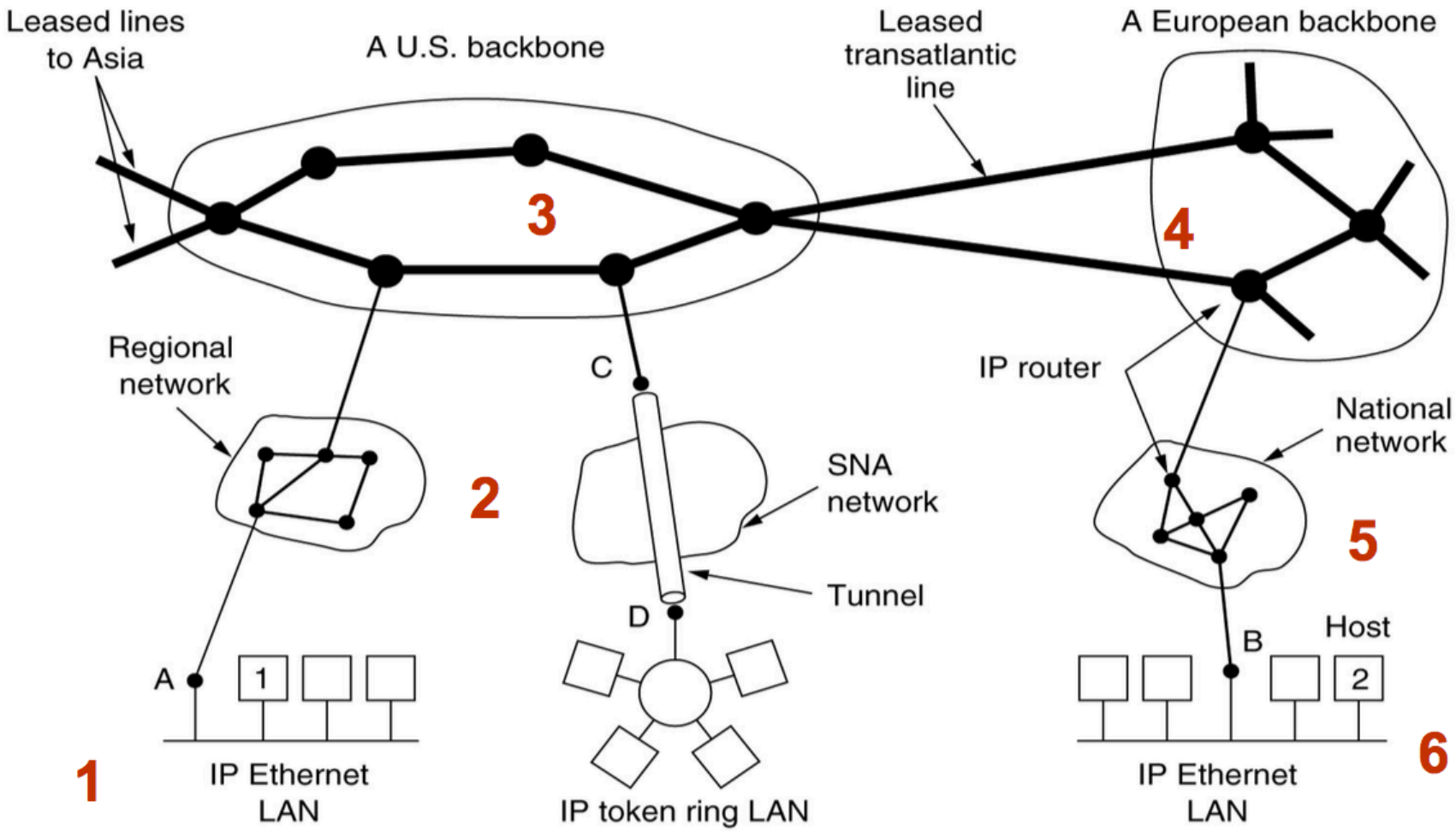
# 10: costi e prestazioni

- Se un sistema ha costi troppo elevati o prestazioni troppo deludenti nessuno lo usa.

# Internet

- A livello network Internet viene vista come un insieme di sottoreti, dette “Autonomous System”, interconnesse
- Non esiste una vera e propria struttura, ci sono invece dei backbone con linee e router ad alte prestazioni.
- Attaccati ai backbone ci sono reti regionali o di organizzazione
- Attaccate alle reti regionali ci son le diverse LAN di compagnie, università e ISP.

# Internet


Leased lines to Asia
A U.S. backbone
Leased transatlantic line
A European backbone
3
4
Regional network
C
IP router
National network
2
SNA network
5
Tunnel
D
B
Host
A
1
2
1
IP Ethernet LAN
IP token ring LAN
IP Ethernet LAN
6

# IP

- La colla che tiene tutto insieme è un comune protocollo di livello network chiamato **IP** (**Internet Protocol**)
- A differenza di altri protocolli di livello network, questo è stato progettato fin dall'inizio con l'idea dell'internetworking
- Fornisce un servizio (non garantito) che trasporta datagram da una sorgente ad un destinatario

# Comunicazione al livello Internet: datagrammi IP

- Il layer di trasporto prende i pacchetti del protocollo di trasporto e li spezza in datagrammi.
- Questi possono essere grandi fino a 64KBytes ma in pratica non sono più grandi di 1500 Bytes (quindi stanno in un frame Ethernet)
- Ogni datagramma viene spedito nella Internet, magari frammentato in pezzi più piccoli.
- Quando tutti i pezzi arrivano a destinazione vengono riassemblati nel datagramma originale.

# IPv4 e IPv6

Esistono due versioni di protocollo a livello internet:

- IP versione 4 (IPv4) [RFC 791]
- IP versione 6 (IPv6) [RFC 4291]

Inizieremo a descrivere IPv4.

# IPv4

E’ diventato il protocollo più diffuso di livello rete ed è la base di Internet.

Si tratta di un protocollo senza connessione:

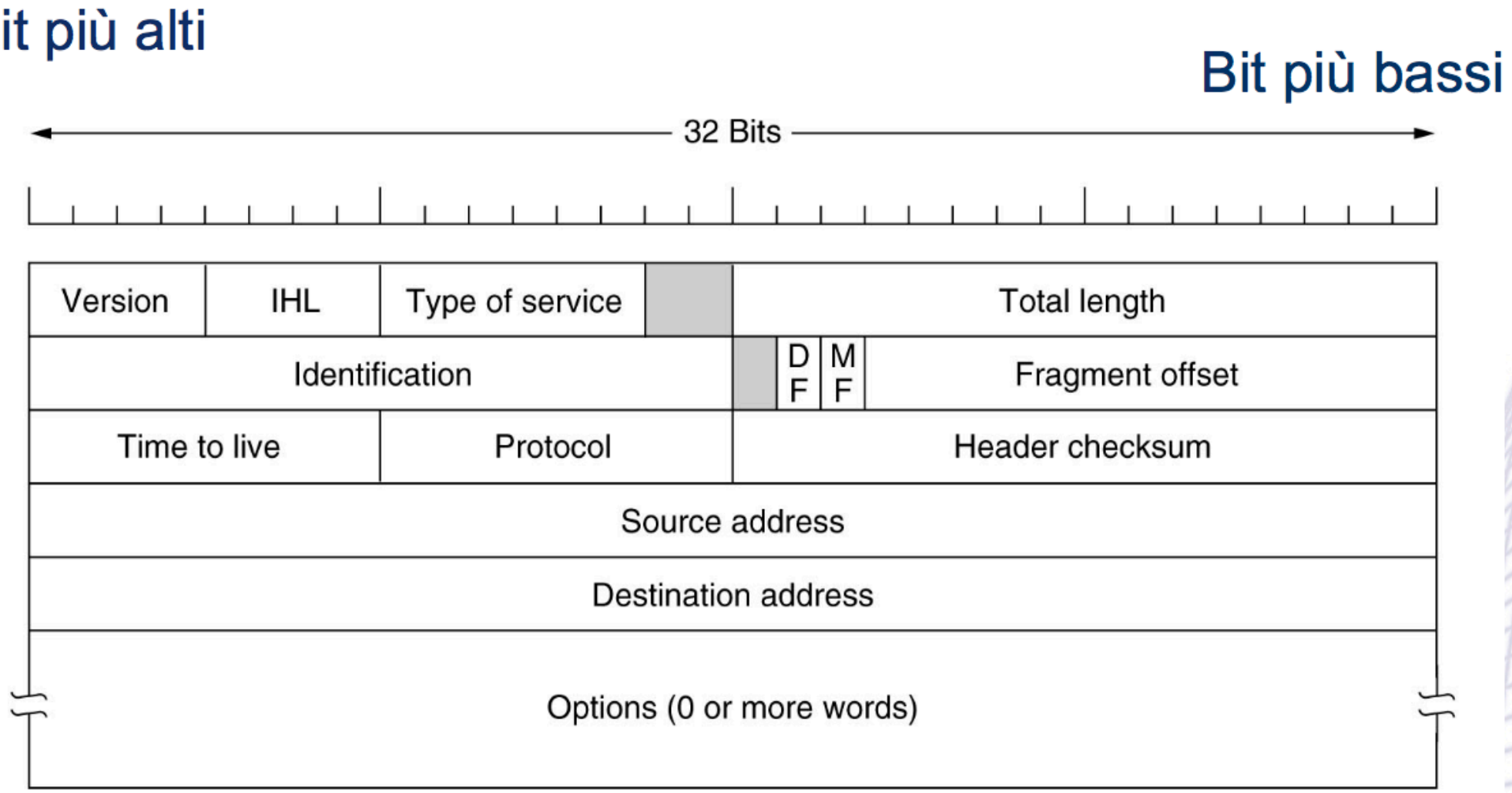
- I datagrammi IP contengono l’identità della destinazione
- Ogni datagramma deve essere poter gestito indipendentemente

Il protocollo IP offre un servizio non affidabile.

# Datagramma IPv4

- È costituito da una parte di header e una parte di dati
- La lunghezza massima di un datagramm IP è 65535 byte, ma di solito ha lunghezza 1024 byte.
- L'header ha una parte fissa di 20 bytes e una parte variabile opzionale (40 byte)
- Il campo "Version" tiene traccia di quale versione del protocollo stiamo usando. Questo permette di avere due tipi di protocolli contemporaneamente e di transire da uno all'altro come ad esempio con la transizione IPv4 – IPv6

# Datagramma IPv4



# Version

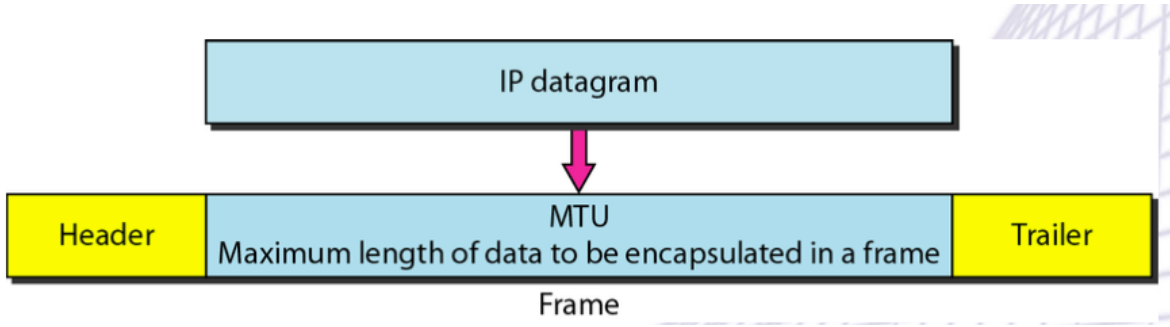
- E’ il numero di versione del protocollo (4 bit).
- Per IPv4, ha valore 4 (da qui il nome IPv4)

# Frammentazione dei datagrammi IP

Prima di descrivere gli altri campi parliamo della frammentazione dei datagrammi.

Ogni protocollo di livello interfacce di rete ha una dimensione massima di frame chiamata MTU (maximum transmission unit).

| *Protocol* | *MTU* |
|---|---|
| Hyperchannel | 65,535 |
| Token Ring (16 Mbps) | 17,914 |
| Token Ring (4 Mbps) | 4,464 |
| FDDI | 4,352 |
| Ethernet | 1,500 |
| X.25 | 576 |
| PPP | 296 |



# Frammentazione dei datagrammi IP

- Quando un datagram viene incapsulato in un frame non deve superare questa dimensione che è imposta dalla tecnologia hardware utililizzata e quindi dal protocollo utilizzato nel layer fisico
- Dal momento che quasi tutti i protocolli hanno un MTU più piccolo i datagram devono essere frammentati
- Questa operazione avviene già nel nodo mittente ma potrebbe darsi che il pacchetto poi attraversi link intermedi con MTU più piccola (router…)
- Il riassemblaggio poi avviene solo alla destinazione finale
- **NOTA**: è impossibile garantire che i frammenti viaggino per lo stesso percorso e arrivino in ordine o anche garantire che non si perdano datagrammi.

# Riassemblaggio dei datagrammi IP

Per poter riassemblare i datagrammi, nell'header ci sono i campi:

- Identification
- Flags: DF, MF
- Fragment offset (spiazzamento di frammentazione)

# Identification

- Permette all'host di destinazione di capire a quale datagramma appartiene il frammento che è arrivato
- Tutti i frammenti di un certo datagram contengono lo stesso valore del campo **Identification**
- È lungo 16 bit quindi dopo 2^16 incrementi si riazzera, comunque un numero molto grande, ci vuole parecchio tempo per cui in una rete lo possiamo considerare come un numero unico (nota che ogni datagramma contiene anche indirizzo sorgente e indirizzo destinazione)

# Flags e Fragment Offset

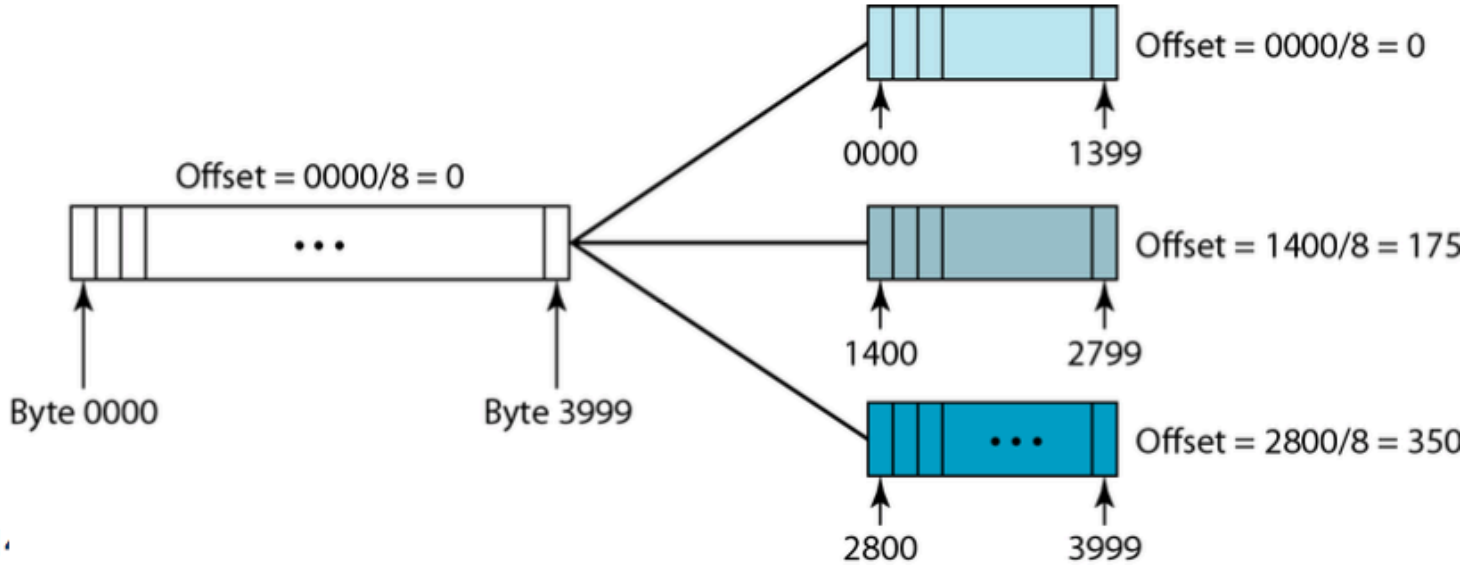
- **Flags**: dopo un bit non usato si trova un bit **DF** (**Don't Fragment**) che ordina di non frammentare il pacchetto perché il ricevente non lo sa riassemblare.
  - Vuol dire che per arrivare a destinazione potrebbe passare per una route non ottimale.
  - Se non ci riesce deve scartarlo e mandare un messaggio con ICMP
- Poi c'è un bit **MF** (**More Fragment**).
  - Tutti i frammenti escluso l'ultimo hanno questo bit settato. Serve per capire quando sono arrivati tutti i frammenti
- **Fragment Offset**:
  - Serve per capire all'interno del datagram dove va situato il corrente frammento.
  - Tutti i frammenti a parte l'ultimo devono essere multipli di 8 Byte che è l'unità fondamentale di frammentazione. Dal momento che il campo è lungo 13 bit segue che ho al massimo 8192 frammenti per datagram e un lunghezza totale di 65536 bytes .

**NOTA:** nel caso in cui un datagramma arrivi incompleto TCP lo rispedisce.

# Esempio di frammentazione

Datagram di 4000 byte frammentato in 3 frammenti lunghi 1400byte

- Devo numerare da 0 a 3999
- Il primo da 0 a 1399 per cui l'offeset è 0/8 = **0.** **MF = 1**
- Il secondo da 1400 a 2799: l'offset vale 1400/8 = **175.** **MF = 1**
- Il terzo 2800 a 3999: l'offset vale 2800/8 = **350.** **MF = 0**

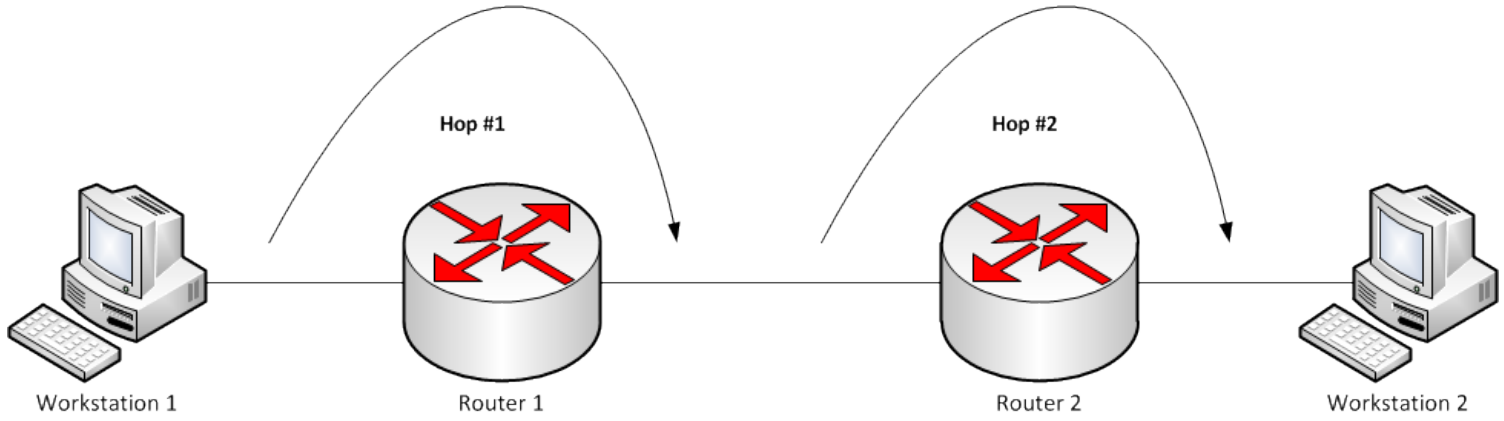


# Time to live

- Un contatore che limita la durata di vita del pacchetto.
- Dovrebbe contare in secondi con un massimo di 255 secondi  (8 bit)
  - Ma questo richiede che tutti i router abbiano i clock sincronizzati e che siano in grado di calcolare il tempo che il datagramma impiega per essere trasferito
  - Invece viene decrementato di uno ad ogni hop e decrementato più volte quando il pacchetto è accodato per lungo tempo in un router
  - In pratica conta gli hops  sommati agli accodamenti
  - TTL iniziale spesso 64 ma cambia da s.o. di host mittente
- Quando il contatore arriva a zero il pacchetto viene scartato e viene mandato un warning al mittente. Questo accorgimento impedisce ad un datagram di viaggiare all'infinito a causa per esempio di un loop nel routing

# Cosa è un Hop?

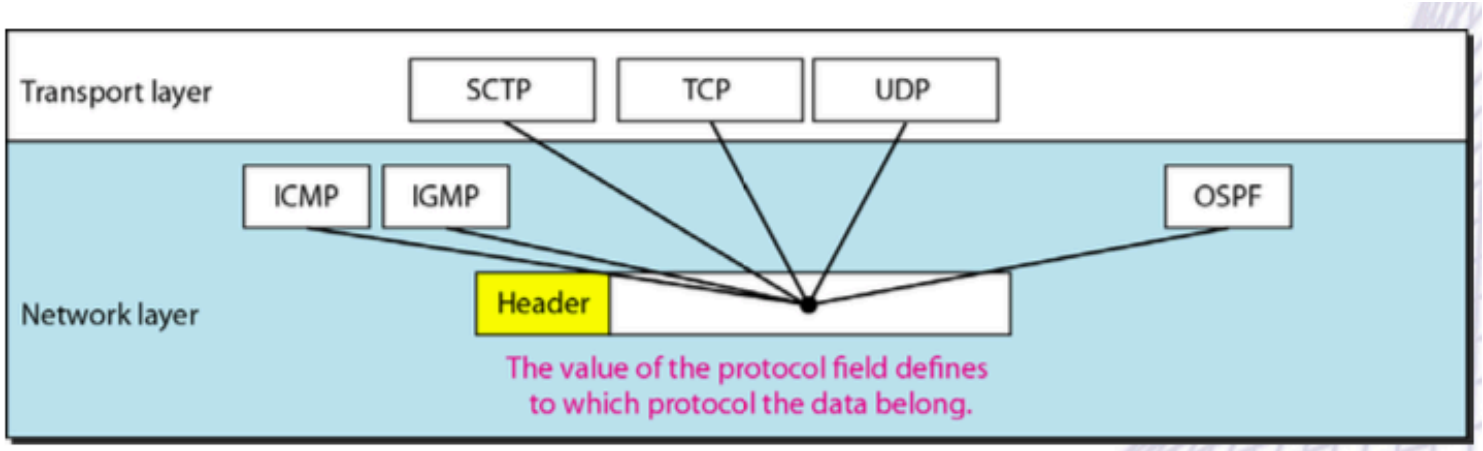
L’hop è una porzione del percorso tra sorgente e destinazione. Il numero di hop conta il numero di device intermedi (router) sul percorso.



# Protocol

- Cosa devo fare quando ho riassemblato il pacchetto?
- Devo consegnarlo al livello di trasporto! In questo campo (8 bit) posso specificare che vada a TCP oppure a UDP o ad un altro protocollo definito da IANA (Internet Assigned Numbers Authority https://www.iana.org/) e in origine definito nella RFC 1700.

| *Value* | *Protocol* |
|---|---|
| 1 | ICMP |
| 2 | IGMP |
| 6 | TCP |
| 17 | UDP |
| 89 | OSPF |



# Header Checksum

Verifica solo l’header

- I dati sono eventualmente verificati dai livelli superiori
- Utile per rivelare errori dovuti a disfunzioni della memoria all’interno dei router
- Se tutto è ok il valore è zero:
  - Si tratta del complemento a uno della somma dei complementi a uno dei campi dell’header.
- Questo valore deve essere ricalcolato ad ogni hops dal momento che ogni volta cambia almeno il TTL

# Source e Destination Address

- Indicano l’indirizzo di rete delle NIC (=scheda di rete) degli host del mittente e destinatario (32 bit ciascuno)
- Li vedremo in dettaglio in seguito

# Campo Options

- Sono state pensate per permettere a future versioni del protocollo di includere informazioni non presenti nell'idea originale
- Per sperimentare nuove idee senza riservare nell'header dei bit che sarebbero stati usati raramente
- Hanno lunghezza variabile e ognuna inizia con un codice identificativo di un 1byte
- Alcune hanno anche un byte di lunghezza e poi uno o più data byte
- All'inizio erano 5 e le vedremo in dettaglio. La lista aggiornata viene mantenuta da IANA

# Indirizzi IP

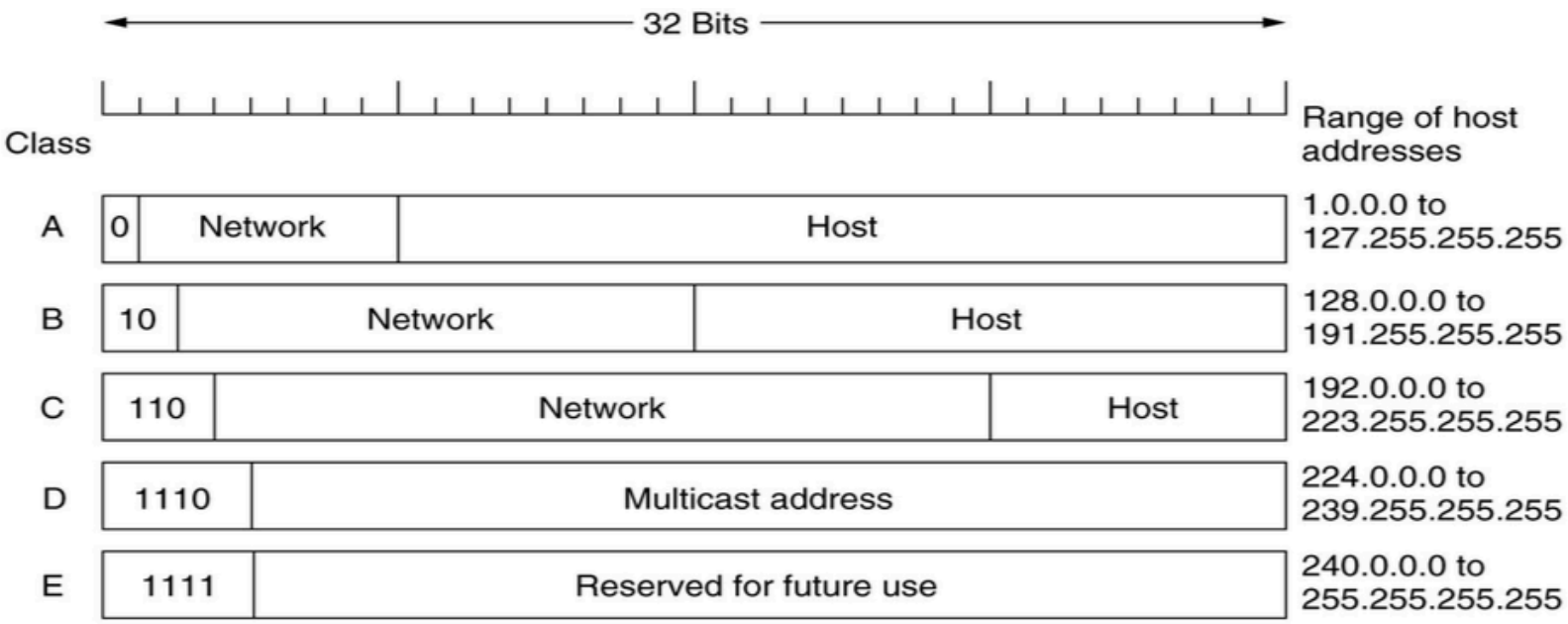
Ogni NIC (presente negli host o nei router) ha un indirizzo IP che codifica l'indirizzo della NIC

- Gli indirizzi IP sono in teoria unici e sono lunghi 32 bit
- 2^32 = 4.294.967.296 indirizzi diversi
- I 32 bit si indicano come sequenza di bit di solito raggruppati in byte
- Es: **01110101 10010101 00011101 00000010**
- Nella notazione decimale ogni byte è convertito in base 10.
- Es: **117.149.29.2**

Ricordate un router ha almeno un indirizzo IP per ogni porta di rete  e che una porta di un host o di un router ha un indirizzo per ogni rete su cui è presente.

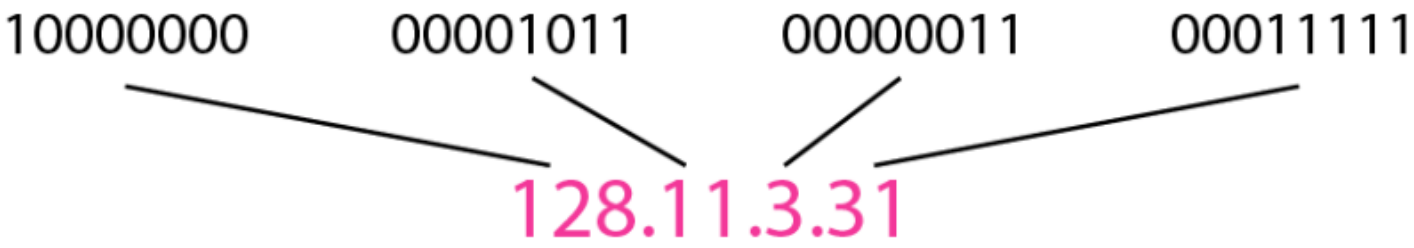
# Classi di indirizzi IP

- Classe A: 128 networks con 16M indirizzi ciascuna
- Classe B: 16384 networks con 64K indirizzi ciascuna
- Classe C: 2M networks con 256 indirizzi ciascuna
- Classe D: Indirizzi Multicast (enorme spreco 268 M indirizzi)
- Classe E: Riservata (di nuovo 268M indirizzi)



# Numerazione

- I numeri degli indirizzi IP sono assegnati da un'associazione no-profit chiamata ICANN (https://www.icann.org/) per evitare conflitti (due nodi con lo stesso numero)
- ICANN delega poi gerarchicamente l'assegnazione delle reti ad altri enti
- Vediamo di nuovo la notazione "dotted decimal" per cui C0290614 viene scritto 192.41.6.20
- Altro esempio in figura 800B031F diventa 128.11.3.31



# Indirizzi di rete e di host

Con l'indirizzamento a classi un indirizzo di classe A,B o C viene diviso in un **indirizzo di rete** e un **indirizzo di host**

- Classe A: 7 bit di indirizzi per la rete (il primo identifica la classe), 24 bit per gli host
- Classe B: 14 per la rete (i primi due per la classe) e 16 per gli host
- Classe C: 21 bit per la rete (primi 3 per la classe) e 8 bit per gli host
- Il concetto di host e rete non esiste per le classi D e E

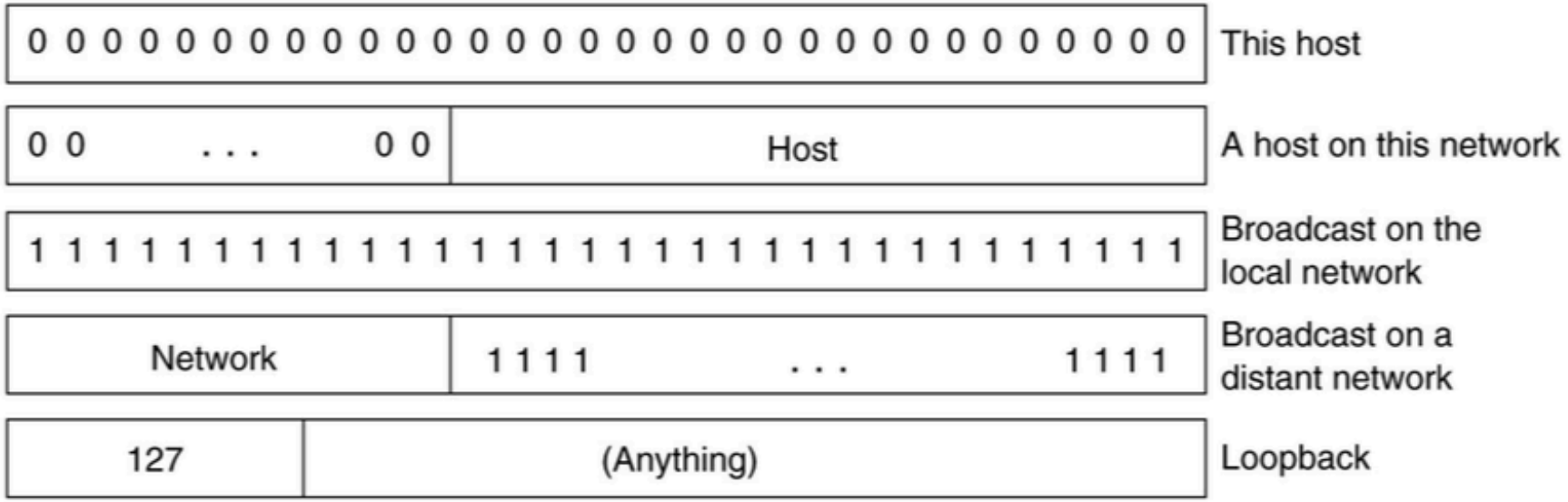
# Maschere

- Per estrarre rapidamente da un indirizzo IP l'indirizzo di rete o quello di host si usano delle maschere di bit
- Per avere l'indirizzo di rete faccio un AND (bit a bit) tra l'indirizzo e la sua maschera
- Per avere l'indirizzo di host faccio un AND tra l'indirizzo e il complemento della maschera

| *Class* | *Binary* | *Dotted-Decimal* | *CIDR* |
|---|---|---|---|
| A | **11111111** 00000000 00000000 00000000 | **255**.0.0.0 | /8 |
| B | **11111111 11111111** 00000000 00000000 | **255.255**.0.0 | /16 |
| C | **11111111 11111111 11111111** 00000000 | **255.255.255**.0 | /24 |

# Indirizzi speciali

- L'indirizzo 0.0.0.0 viene usato dagli host nella fase di boot
- Indirizzi con la parte network a zero indicano che si vuole indirizzare una macchina nella stessa rete
- Indirizzo con tutti i bit a 1→255.255.255.255 significa "tutti i nodi", un broadcast nella LAN a livello network



# Indirizzi speciali

- Indirizzi con una parte di rete normale e solo la parte di host a 1 significa un broadcast su quella rete remota (di solito l'amministratore del router blocca questi pacchetti)
- Indirizzi che cominciano con 127 sono per il test del loopback. I pacchetti inviati a questo indirizzo non vengono messi sul cavo ma sono processati localmente e trattati come pacchetti in ingresso.
- In particolare l'indirizzo di loopback di default per il localhost è 127.0.0.1.
- **NOTA:** data una network (subnet) il primo e l'ultimo indirizzo non sono assegnabili a nessun host, in quanto sono riservati rispettivamente come indirizzo della network e come indirizzo di broadcast.