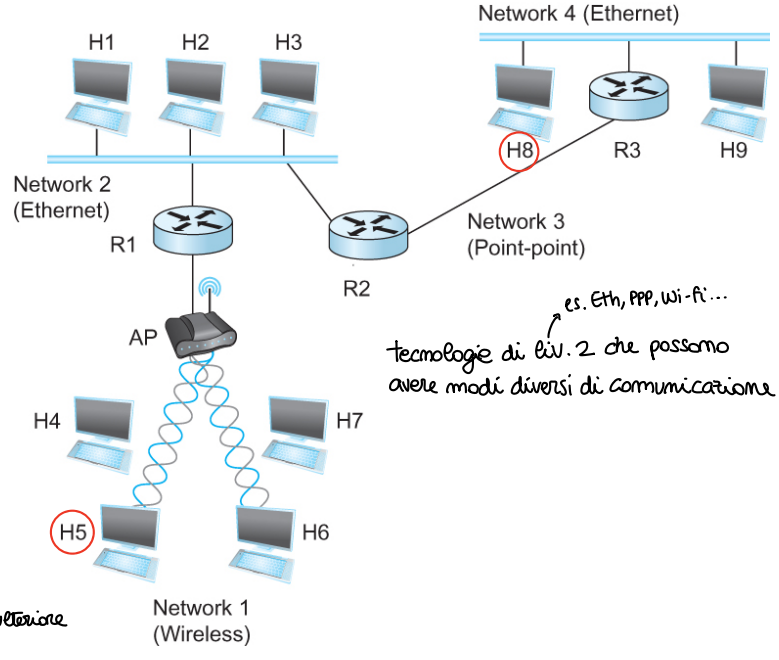
Internetworking

livello 3 interrete, il commutatore liv. 3 è il ROUTER: esso INTERCONNETTE reti ETEROGENEE (lo SWITCH, AP interconnettono reti OMOGENEE (i frame hanno i MAC))

- What is an **internetwork**?
  - An **arbitrary collection of networks interconnected to provide some sort of host-to-host packet delivery service**
  - Example aside: a simple internetwork where H represents hosts and R represents routers

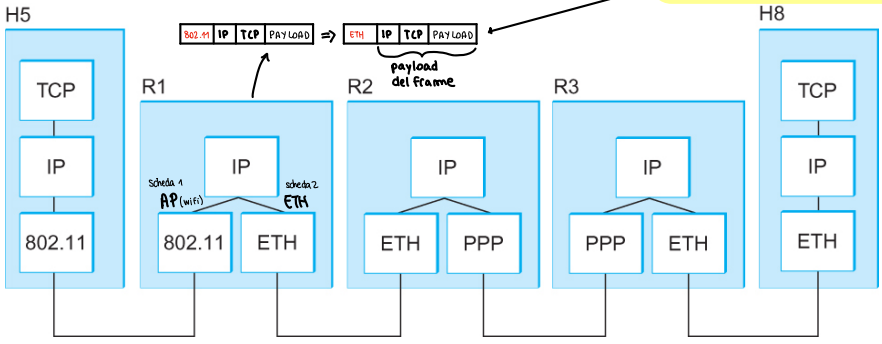


se H5 vuole trasmettere ad H8, come fa? → deve identificare gli host con un'ulteriore astrazione, ovvero l'indirizzo IP

Internetworking

- In order to implement this abstraction we need a level 3 protocol
- Most used nowadays: IP (which stands for *Internet Protocol)*
  - Key tool used today to build scalable, heterogeneous internetworks
  - It runs on all the nodes in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork

ROUTER: decapsulare un payload di livello 2 e re-incapsularlo per un altro livello 2 + decrementare TTL

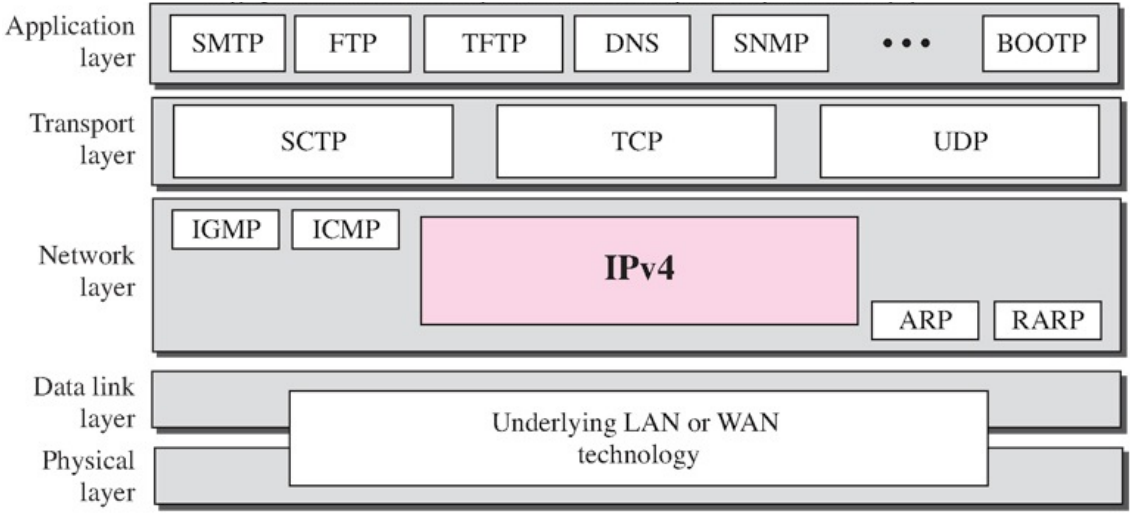


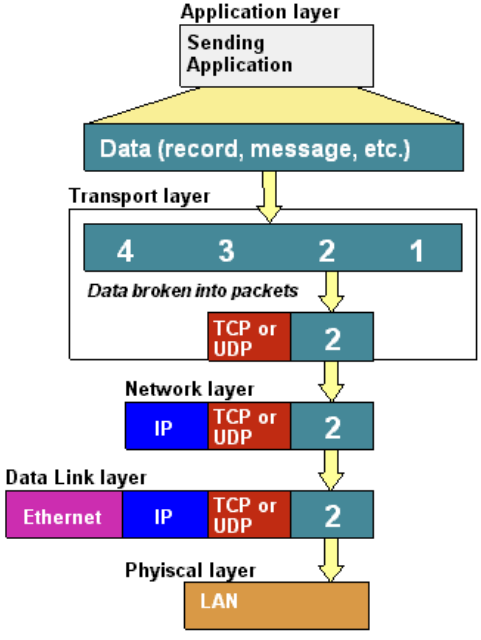
trasmissione da H5 a H8

IP: position in TCP/IP stack

liv. 2: dato un MAC destinatario, se il dispositivo (destinatario) è sulla mia rete, io gli faccio arrivare il frame

liv. 3: dato un IP io te lo faccio arrivare, indipendentemente se è connesso nella mia rete o no il destinatario
↳ es. numero di telefono ↳ dovunque sia nel mondo io glielo faccio arrivare





Internetworking

- Key aspect: the IP layer does also **packet forwarding**

↳ dove decide l'interfaccia (di liv. 2) con cui fare uscire (es. Eth, PPP, bluetooth, LTE, ...)

- A packet (sometimes called “datagram”) can arrive to IP layer in two ways:
  - when an upper layer (e.g. TCP or UDP) require to send some data (“payload”) to another host: then this data is encapsulated in a IP datagram
  - as a packet received from a lower layer (i.e. from an interface).
- In any case, when IP has to deal with a packet:
  - if it is addressed to the local host (“myself”): the IP header is stripped and the payload is delivered up to the right transport protocol
  - if it is addressed to another host: it is sent out using an interface, i.e. using an underlying datalink layer (which encapsulates the IP packet in a datalink frame)

IP Service Model

- Packet Delivery Model
  - Connectionless model for data delivery
  - Best-effort delivery (unreliable service)
    - packets are lost
    - packets are delivered out of order
    - duplicate copies of a packet are delivered
    - packets can be delayed for a long time
- Global Addressing Scheme
  - Provides a way to identify all hosts in the internetwork
  - Abstracts from the underlying layer 2 (MAC) addresses

commutaz. di pacchetto, ogni pacchetto è indipendente. Ogni DATAGRAMMA porta con se sufficienti informazioni perchè la rete possa inoltrare il pacchetto fino alla sua destinazione corretta. Il pacchetto viene inviato e la rete fa del proprio meglio (BEST EFFORT) per farlo arrivare a destinazione

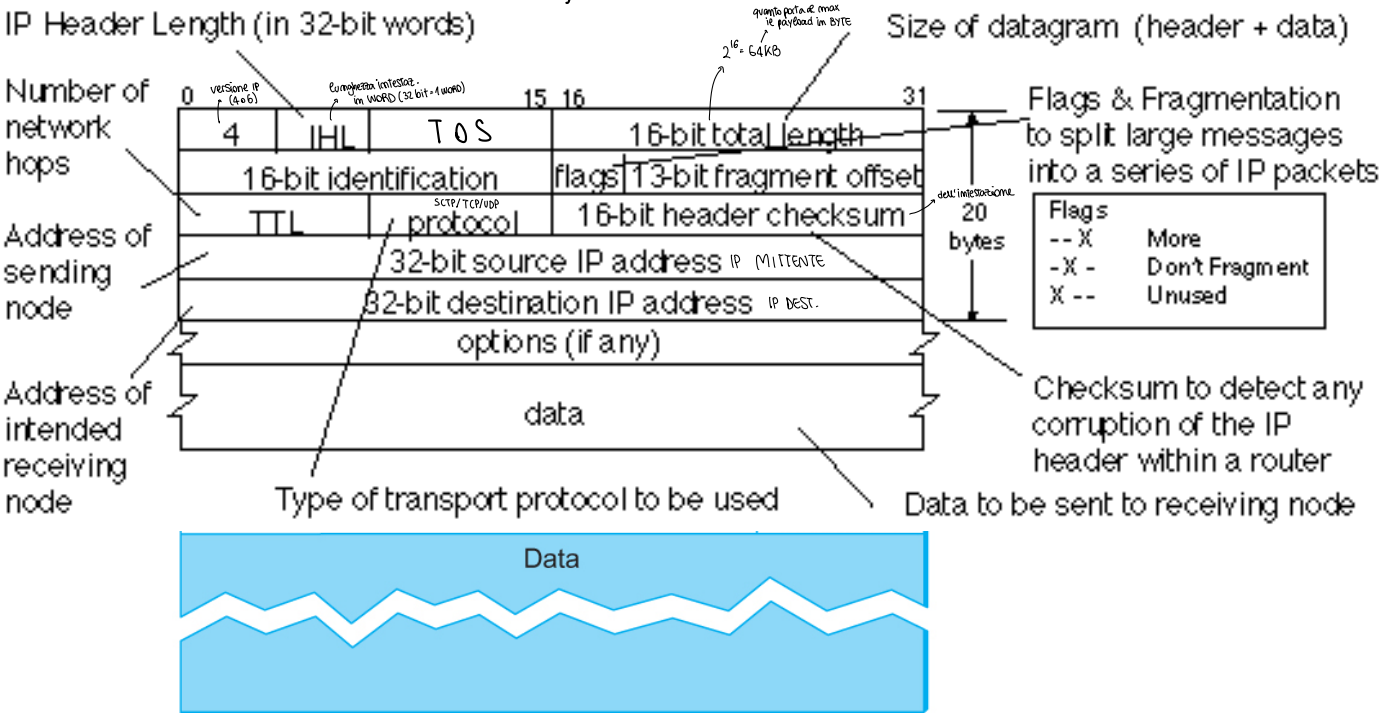
pacchetti possono andare persi

errori di liv 3 : - fa sbagliare strada ai pacchetti
- i router si intasa per poca memoria

La consegna best effort non significa solo che i pacchetti possono andare smarriti, ma anche che possano arrivare doppi. I protocolli di liv. superiore si occuperanno di raggruppare correttamente tutto

Packet Format

intestazione di un pacchetto IPv4
↳ MIN: 20 Byte



Datagram IPv4

- Version (4): currently 4 for IPv4 (or 6 for IPv6)
- Hlen (4): number of 32-bit words in header (hence real length is 4xHlen)
- TOS (8): type of service
- Length (16): total length of this packet, including header, in bytes
- Ident (16): used by fragmentation
- Flags (3) and Offset (13): used by fragmentation.
  - Flag bit 0: must be 0
  - Flag bit 1: Don't Fragment (DF)
  - Flag bit 2: More Fragments (MF)
- TTL (8): number of hops this datagram can travel
- Protocol (8): demux key (TCP=6, UDP=17)
- Checksum (16): of the header only
- DestAddr & SrcAddr (32 bits each)

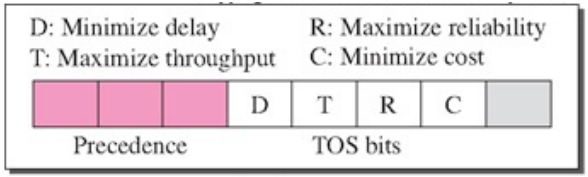
ogni volta che passa per un ROUTER, questo oltre che cambiare l'interfaccia di liv. 2, decrementa il TTL per evitare LOOP. (Nelle LAN si usa ALGO SPANNING TREE)

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

Datagram IPv4

implementiamo una specie di QoS, ovvero riadattare es. throughput, jittero ritardo in base al contesto

- TOS - Type of Service (8 bit)
  - should be used to specify how to treat the datagram
  - Not very followed nowadays… we cannot trust the user!

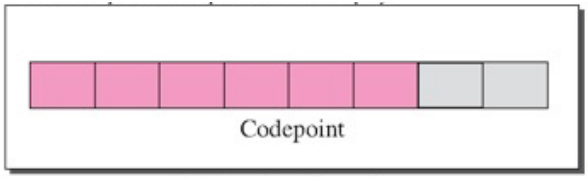


- Original interpretation: TOS
  - Priority in case of congestion
    - network handling datagrams should have the precedence
  - Kind of service requested: (lo scrive il mittente) → sono SUGGERIMENTI per le ROUTER, poi non è detto che lo faccia
    - 0000: normal (default) → decide le router
    - 1000 (D): minimize delay
    - 0100 (T): maximize throughput
    - 0010 (R): maximize reliability
    - 0001 (C): minimize cost → costo energetico o costo al Kb, dipende da router a router
  - every transport protocol has an associated default service
    - TELNET, FTP (com), SMTP (com): D; SNMP: R; FTP (dati), SMTP (dati): T; ecc.

↳ la PRIORITÀ è il TTL, ovvero delay basso (deve essere reattivo)

Datagram IPv4: DSCP

- Modern interpretation: **Differentiated services (DSCP)**
  - New codes
  - If 3 rightmost bits = 000:
    - 3 leftmost bits are interpreted as priority, as in TOS
  - If 3 rightmost bits dx ≠ 000:
    - 6 leftmost bit define 64 services
      - xxxxx0: standard IETF services
      - xxxx11: services defined by local authorities
      - xxxx01: temp. use, experimental



IP Fragmentation and Reassembly

la differenza tra AP e ROUTER: AP usa con WIFI payload da 1500 B (al posto che 2346) in modo da avere lo stesso payload della Ethernet (1500), quindi reti OMOGENEE. Infatti il commutatore di liv. 2 collega reti OMOGENEE
MTU
ROUTER invece se payload > 1500 allora segmenta

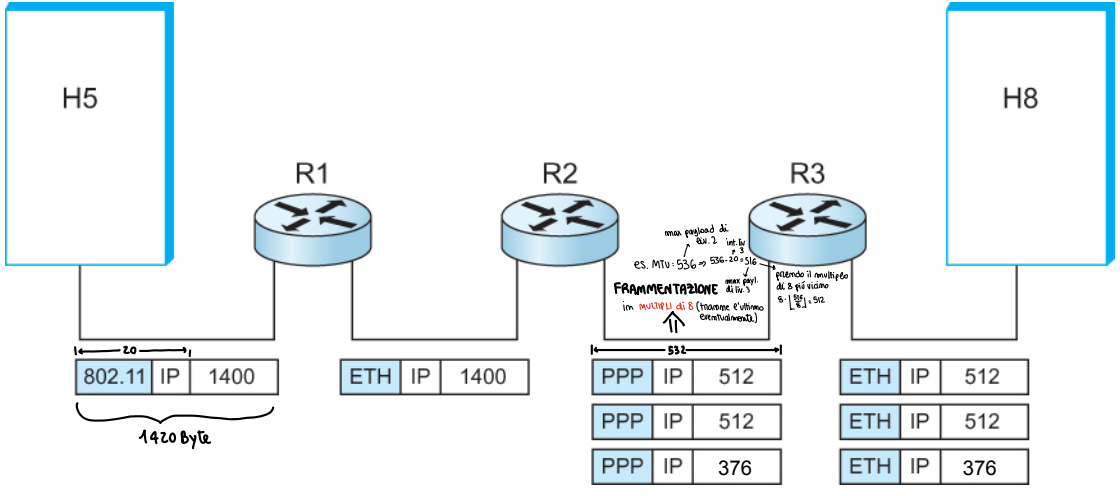
- Each physical network has some MTU (*Maximum Transmission Unit*), which depends on the technology
  - dim. massima del payload
- At transport level we do not know about MTU
  - es. se da WIFI voglio passare ad Eth, i payload non vanno bene
- What should we do if a datagram does not fit in a single frame?
  - If the Don't Fragment flag is 1, just drop the datagram and notify the sender (using ICMP)
  - If DF=0, the router can proceed with *fragmentation*

Network/protocol	MTU (bytes)
Hyperchannel	65535
Token Ring	17914
FDDI	4352
WiFi	2346
Ethernet	1500
X.25	576
PPP	Negotiated

IP Fragmentation and Reassembly

- Datagrams are fragmented at network level, transparently for higher levels. Higher levels see only whole datagrams.
- Strategy
  - Fragmentation occurs in a router when it receives a datagram that it wants to forward over a network which has (MTU < datagram)
  - Reassembly is done at the receiving host
  - All the fragments carry the same identifier in the Ident field
  - Fragments are self-contained datagrams
  - No recovery for missing fragments: if some are still missing after the timeout (~15-60 seconds), all data is discarded and an ICMP message is sent to sender

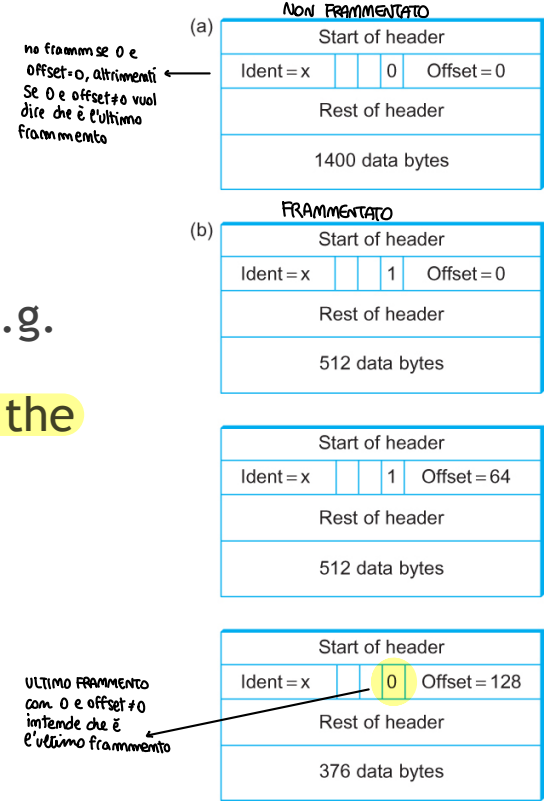
IP Fragmentation and Reassembly



- IP datagrams traversing the sequence of physical networks

IP Fragmentation and Reassembly

- Header fields used in IP fragmentation.
  - (a) Unfragmented packet; → NON SEGMENTATO
  - (b) corresponding fragmented packets. Notice: offset field = real offset / 8 (e.g. 64 = 512 / 8) hence in each fragment (but last one) the length of payload is a multiple of 8.



Global Addresses

- MAC (ind. liv. 2) sono assegnati dal produttore e così rimangono per sempre → assegnamento STATICO
- IP (ind. liv. 3) " " in base alla posizione del dispositivo → assegnamento DINAMICO

- Properties
  - *globally unique*: a host may have many addresses, but an address cannot be assigned to more than one host
  - *hierarchical:* the address is split in two parts
    - First part denotes the *network* — assigned uniquely to a single entity (*Autonomous System*) by central authorities → ogni
    - Second part denotes the *host* with the network
    - Each autonomous system can assign addresses within its networks as it prefers
- In IPv4: 32 bit, written using the dot notation
  - 10.3.2.4 → 00001010. → classe A
  - 128.96.33.81
  - 192.12.69.77
- How to tell the Network number and the host number?

il reindirizzamento non può avvenire con le tabelle di switch, perché non sono SCALABILI (per miliardi di host ci vorrebbe troppa memoria solo per memorizzare le tabelle e poi andrebbe scorsa => INEFFICIENTE)

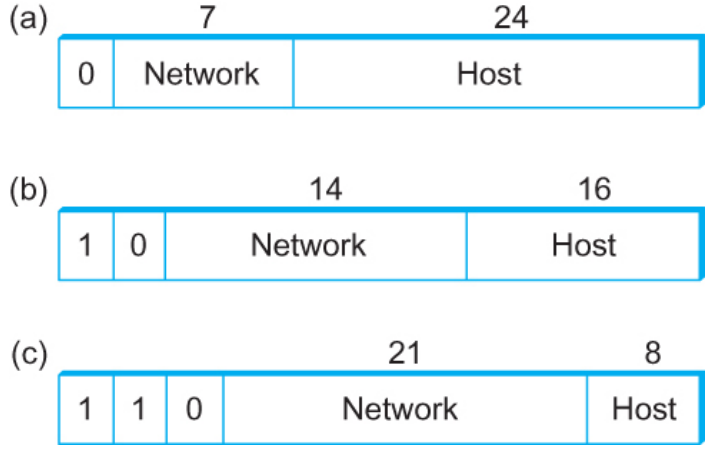
Network number	Host number

router per reindirizzare deve saper solo guardare l'IP destinazione, nella precisione solo la parte "NETWORK NUMBER" (così riduco le entry nella tabella dei router => occupa meno memoria => scansione + efficiente)
↳ la parte HOST NUMBER serve all'ultimo router prima dell'host

Global Addresses

- Originally, 3 classes of networks:
  - **Class A**: addresses beginning with 0; next 7 bits are interpreted as network number; remaining 24 bits are the host number within network
  - **Class B**: addresses beginning with 10; next 14 bits are interpreted as network number, the remaining 16 bits are the host number within network
  - **Class C**: addresses beginning with 110; next 21 bits are interpreted as network number, the remaining 8 bits are the host number within network
  - (Addresses beginning with 111 are not used for host addressing)

in binario



Global Addresses (IPv4) ⇒ così era stata pensata all'inizio degli anni 80

RANGE classe A { 10.0.0.0 → NETWORK NUMBER ⋮ 10.255.255.255 → BROADCAST
indirizzi disp (a rete): $256^3 - 2 = 2^{24} - 2$
reti possibili: 2^7
ind. totali ≃ 2^{31}

	RETI	IND. x RETE	TOTALE
A	$2^7 = 128$	$2^{24} - 2$	$\simeq 2^{31}$
B	$2^{14} = 16384$	$2^{16} - 2$	$\simeq 2^{30}$
C	2^{21}	$2^8 - 2 = 254$	$\simeq 2^{29}$
			$\simeq 3{,}75 \cdot 10^9$ MILIARDI

Dato che non tutti i dispositivi possono avere un IP (ce ne sono troppi) si utilizzano NAT e IPv6

- Within a network, two addresses cannot be used for hosts:
  - first address (all "0" in host part) identifies the network (e.g. 158.110.0.0)
  - last address (all "1" in host part) is used for broadcast (e.g. 158.110.255.255)
  - Remaining address can be assigned to hosts
- Number of hosts for each network
  - Class A: 2^7 = 128 possible networks, with 2^{24} - 2 = 16777214 hosts each
  - Class B: 2^{14} = 16384 possible networks, with 2^{16} - 2 = 65534 hosts each
  - Class C: 2^{21} = 2097152 possible networks, with 2^8 - 2 = 254 hosts each
- Overall: 2,1*10^6 networks (actually less, due to "special" networks), and theoretical limit of possible hosts: 3753869056 ≅ 3,75*10^9

IP Datagram Forwarding

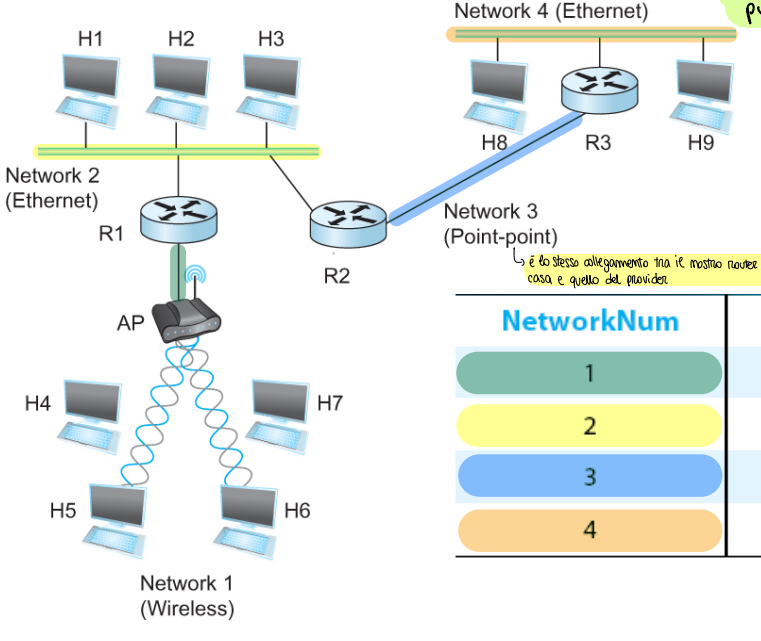
- Cannot be by destination address (too many), like in Level 2 switches
  - (Actually, level 3 switches do exist, but are limited inside single AS)
- Instead, forwarding is by *destination network*, that is, the network which the destination address belongs to
- forwarding table maps network number into next hop
- Strategy followed by each node:
  - every packet contains destination's address, from which we can get the destination network address
  - if the node is directly connected to destination network, then forward the packet to host
  - if not directly connected to destination network, then forward the packet to a router which should know how to handle it
- each host has a default router (*gateway*) and maintains a forwarding table
  - some algorithm will be needed for this

IP Datagram Forwarding

im LAN (wi-fi) i computer comunicano a liv. 2

Se io voglio inviare un pacchetto (fuori dalla mia rete) devo sapere l'IP di destinazione. Se è un server conosciuto, l'IP lo trovo con il DNS, altrimenti devo saperlo.
Se l'IP dest usa il NAT (e quindi so l'IP pubblico) non può arrivare il pacchetto dentro il NAT

- Example: routing table for router R2



è lo stesso collegamento tra il nostro router di casa e quello del provider

non so di preciso dov'è il destinatario, so che il next hop è quello a cui si avvicina

NetworkNum	NextHop	
1	R1	→ per arrivare alla rete 1, deve passare per R1
2	Interface 1	→ è connesso direttamente
3	Interface 0	→ è connesso direttamente
4	R3	→ per arrivare alla rete 4, deve passare per R3

IP Datagram Forwarding

→ ALGORITMO di FORWARDING ≠ ALG. DI ROUTING (che serve per comporre le tabelle di forwarding)

↳ deve essere veloce, miliomi di pacchetti al sec.

- **Algorithm**:

```
if (NetworkNum of destination == NetworkNum of one of my interfaces) then
        directly deliver packet to destination over that interface
else if (NetworkNum of destination is in my forwarding table) then
        deliver packet to NextHop router
else if there is a default router
        deliver packet to default router
else drop packet
```

↳ se l'ind. di rete è = ad una delle mie interfacce → siamo nella stessa rete locale

↳ se lo ho in tabella, non nella mia rete, consegno il pacchetto al next hop

↳ lo consegno al default gateway della mia rete

- For a host with only one interface and only one default router in its forwarding table, this simplifies to

↳ caso degenere del singolo pc che o invia nella sua rete o invia al gateway

```
if (NetworkNum of destination == my NetworkNum) then
        deliver packet to destination directly
else
        deliver packet to default router
```

↳ lo verifica facendo (SUB.MASK AND IP DEST.) se appartengono alla stessa network number

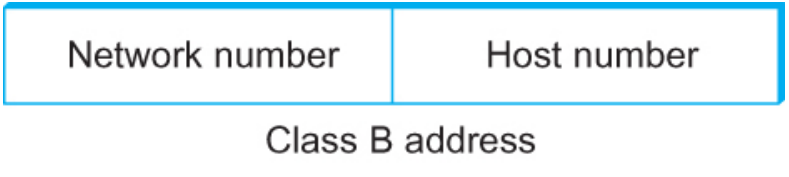
per i cicli c'è il TTL che eventualmente fa scartare i pacchetti (8 bit di TTL → max: 255 hop). ⇒ Devo avere le tabelle di FORWARDING piccole altrimenti la ricerca è inefficiente

Subnetting

→ sottoreti, lo uso per ottimizzare lo spazio
es. se ho 3 host in una rete non ha senso usare un pool di es. 1000 indirizzi

- A and B networks can be too large for a single physical network
- Add another level to address/ routing hierarchy: subnet
- Subnet masks define variable partition of host part of class A and B addresses, according to physical nets → la SUBNET MASK introduce un nuovo livello di GERARCHIA all'interno della rete
- Subnets visible only within network (i.e. autonomous system)

USO una sottoparte dei bit di host per identificare come sottorete



↳ la lunghezza della subnet mask di una sottorete può essere diversa rispetto a quella delle altre

es: indirizzo IP in UNI: 158.110.228.53 => classe B (precisamente la 158.110.0.0) ← comprato dall'UNI nel 1980
10011100. ...
↳ 65534 ind. disponibili → non sono tanti per l'università però sono tanti per una singola rete
↳ la SUDDIVIDO in TANTE SOTTORETI

quindi, SUBNET MASK: 255.255.255.0 =>
IP LAN: 158.110.0.0

{ 158.110.0.0 → rete 1
⋮
158.110.0.255 → broadcast 1

→ così ho reti da 254 host ciascuno

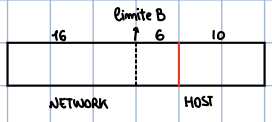
=> { 158.110.1.0 → rete 2
⋮
158.110.1.255 → broadcast 2

in pratica il "range" della rete va fino al numero di bit LIBERI della subnet mask
↓ ovvero mancanti per arrivare a 255
es. 252 → 3 numeri liberi

=> { 158.110.2.0 → rete 3
⋮
158.110.2.255 → broadcast 3

⋮

Se volessi fare un subnetting della rete per contenere 1000 host: 1000 host → $2^{10} - 2 = 1022$ → 10 bit deve avere l'indirizzo host, contro gli 8 precedenti



SUBNET MASK: 11111111.11111111.111111**00**.00000000 , quindi la rete con 1022 host è { 158.110.0.0 → rete
⋮
158.110.3.255 → broadcast
in DECIMALE: 255 . 255 . 252 . 0

In generale, dati un IP (es. 128.96.34.15) e la sua subnet mask (255.255.255.128) posso "scoprire" il numero di sottorete usando AND (in binario)

128.96.34.15:	**10000000.01100000.00100010.00001111**	**AND**
255.255.255.128:	**11111111.11111111.11111111.10000000**	**=**
128.96.34.0:	**10000000.01100000.00100010.00000000**	

bit identificanti l'id dell'host all'interno della sottorete
bit identificanti la subnet mask
bit identificanti l'id della sottorete
bit identificanti che la rete è di classe B
bit identificanti l'id della rete

conclusioni: 10000000.01100000.00100010.00001111

Quando l'host A vuole inviare un pacchetto ad un host B (di cui ha l'IP), fa un AND tra la sua subnet mask (di A) e l'indirizzo B. Se sono nella stessa sottorete gli invia il pacchetto, altrimenti lo invia al default gateway.

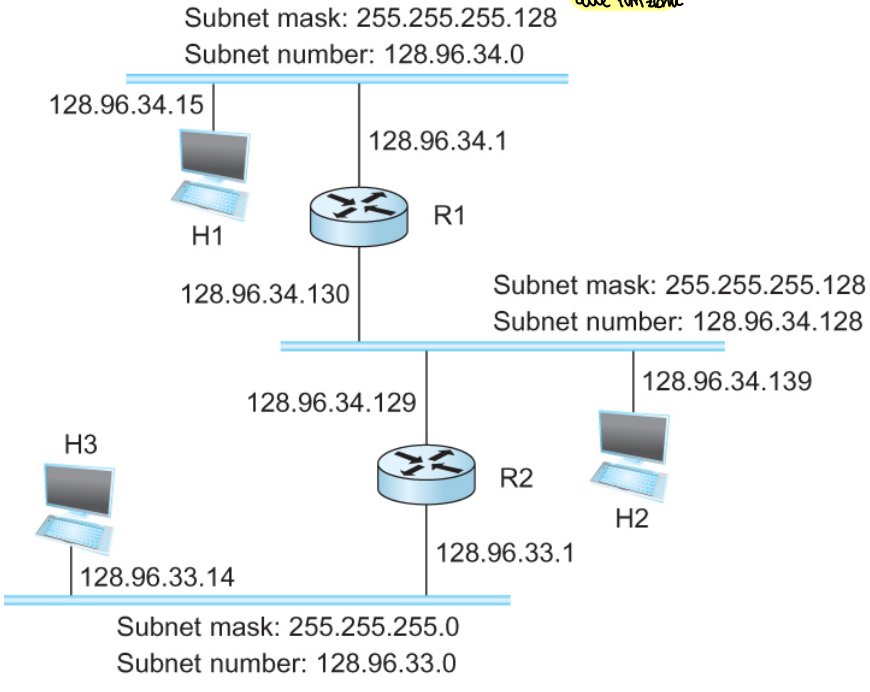
Con la subnet mask posso capire dove inizia e finisce la parte network →
- se devo avere più reti con meno host → AUMENTO LA PARTE NETWORK
- se devo avere più host (di quelli che ho con la subnet attuale) nella rete → AUMENTO PARTE HOST

Preso un IP PUBBLICO es. 179.15.28.0/24, posso farci solo una singola rete da 254 host, se volessi fare più reti separate (pubbliche, quindi senza usare il NAT) devo fare SUBNETTING

Subnet mask	Binary	Networks	Hosts
255.255.255.0	11111111.11111111.11111111.00000000	1	254
255.255.255.128	11111111.11111111.11111111.10000000	2	126
255.255.255.192	11111111.11111111.11111111.11000000	4	62
255.255.255.224	11111111.11111111.11111111.11100000	8	30
255.255.255.240	11111111.11111111.11111111.11110000	16	14
255.255.255.248	11111111.11111111.11111111.11111000	32	6
255.255.255.252	11111111.11111111.11111111.11111100	64	2
255.255.255.254	11111111.11111111.11111111.11111110	128	0

Subnetting

→ lo uso per creare sottoreti più piccole per partizionare meglio il traffico, o suddividere in base alle funzioni



SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

Forwarding table at router R1

es. stabilire la rete di 158.110.239.57 (11101111) => faccio AND LOGICO
255.255.240.0 → già dato
= 158.110.11110000.0
240

11101111
11110000 = 224
11100000

=> la rete EduRoam è 158.110.224.0 (and tra 31 e 0 = 0)
255.255.240.0

quanti utenti posso avere collegati? → $2^{12} - 2 = 4094$ (8+4)

Subnetting

la tabella di forwarding ha forma: | SUBNET NUM | SUBNET MASK | NEXT HOP |

- Forwarding Algorithm

```
D = destination IP address
for each entry <SubnetNum, SubnetMask, NextHop>
        D1 = SubnetMask & D        // bitwise "and"
        if D1 = SubnetNum
                if NextHop is an interface
                   deliver datagram directly to destination
                   and return
                else
                   deliver datagram to NextHop (a router)
if no entry matches
        drop datagram
```

servono entrambi per rilevare la sottorete

ad ogni entry nella tabella di forwarding l'algoritmo calcola la AND tra IP di destinaz. e subnet-mask di ogni entry della tabella e se il risultato coincide con subnet-number allora il pacchetto viene inoltrato al suo next hop.

Subnetting

- Would use a default router if nothing matches
- Not necessary for all 1's in subnet mask to be contiguous
  - weird subnets can be done!
- Can put multiple subnets on one physical network
- Subnets are decided internally by network administrator, and not visible from the rest of the Internet
  - usually assigned by logical separations (e.g. different offices, branches, etc) and physical separations (different physical networks)

Classless Inter-Domain Routing

L'uso delle sottoreti (SUBNET) consente la suddivisione di un indirizzo di classe in più sottoreti, mentre il CIDR consente di aggregare più indirizzi di classe in un' unica "supernete"

- A technique that addresses two scaling concerns in the Internet
  - The growth of backbone routing table as more and more network numbers need to be stored in them
  - Potential exhaustion of the 32-bit address space
- Address assignment efficiency
  - Arises because of the IP address structure with class A, B, and C addresses
  - Forces us to hand out network address space in fixed-size chunks of three very different sizes
    - A network with two hosts needs a class C address
      - Address assignment efficiency = 2/254 = 0.78% → una volta che assegno quella rete gli altri non la possono utilizzare → SPRECO
    - A network with 256 hosts needs a class B address
      - Address assignment efficiency = 256/65535 = 0.39%

es. 255.255.255.0
200.7.0.0
⋮
200.7.1.0
} 255.255.254.0
200.7.0.0

Classless Addressing

- Exhaustion of IP address space centers on exhaustion of the class B network numbers
- Possible solution
  - Do not accept any *Autonomous System* (AS) that requests a class B address unless they can show a need for something close to 64K addresses
  - Instead give them an appropriate number of class C addresses
  - For any AS with at least 256 hosts, we can guarantee an address space utilisation of at least 50%
- What is the problem with this solution?

Classless Addressing

- Problem with this solution: Excessive storage requirement at the routers
- If a single AS has, say, 16 class C network numbers assigned to it, every Internet backbone router needs 16 entries in its routing tables for that AS, even if the path to every one of these networks is the same
- If we had assigned a class B network to the AS
  - The same routing information can be stored in one entry
  - Efficiency = 16 × 255 / 65536 = 6.2%

Classless Addressing

CIDR cerca di bilanciare il desiderio di minimizzare il numero di entry che un router deve conoscere e la necessità di assegnare IP efficientemente

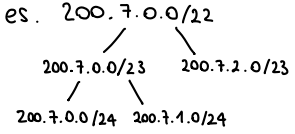
- **Classless Inter-Domain Routing (CIDR)** tries to balance the desire to minimize the number of routes that a router needs to know against the need to hand out addresses efficiently.
- CIDR uses *aggregate routes*
  - Uses a single entry in the forwarding table to tell the router how to reach a lot of different networks
  - Breaks the rigid boundaries between address classes

Classless Addressing

- Consider an AS with 16 class C network numbers.
- Instead of handing out 16 addresses at random, hand out a block of contiguous class C addresses
- Suppose we assign the class C network numbers from 192.4.16 through 192.4.31
- Observe that top 20 bits of all the addresses in this range are the same (11000000 00000100 0001)
  - We have created a 20-bit network number (which is in between class B network number and class C number)
- Requires to hand out blocks of class C addresses that share a common prefix

Classless Addressing

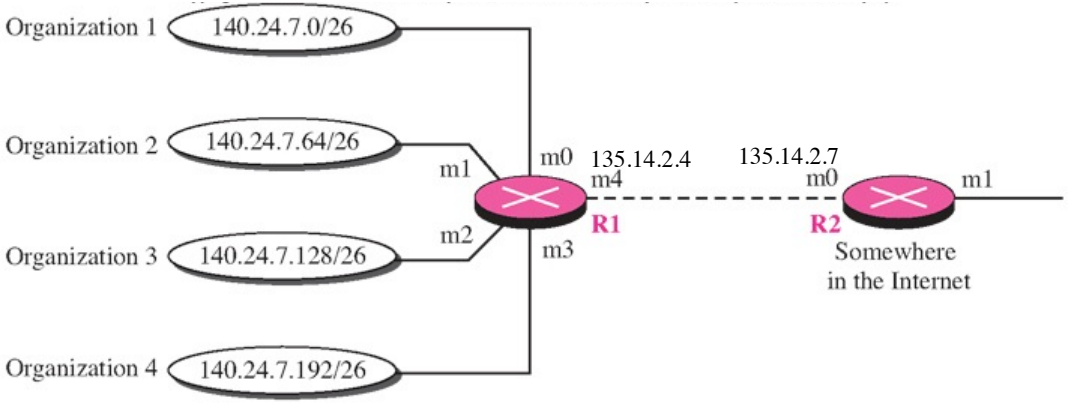
- Requires to hand out blocks of class C addresses that share a common prefix
- The convention is to place a /X after the prefix where X is the prefix length in bits → # bit di rete es. al posto che 255.255.255.0 scrivo 200.7.0.0/24
  - For example, the 20-bit prefix for all the networks 192.4.16 through 192.4.31 is represented as 192.4.16/20
  - By contrast, if we wanted to represent a single class C network number, which is 24 bits long, we would write it 192.4.16/24



Classless Addressing

- How do the routing protocols handle this classless addresses
  - It must understand that the network number may be of any length
- Represent network number with a single pair <length, value>
- All routers must understand CIDR addressing
  - IP forwarding mechanism assumes that it can find the network number in a packet and then look up that number in the forwarding table
  - We need to change this assumption in case of CIDR
  - CIDR means that prefixes may be of any length, from 2 to 32 bits

Classless Addressing: route aggregation



Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	m0
/26	140.24.7.64	----------	m1
/26	140.24.7.128	----------	m2
/26	140.24.7.192	----------	m3
/0	0.0.0.0	135.14.2.7	m4

255.255.255.192 ↓ applicando i primi due bit a sx

default (tutto il resto per uscire) → NON È OBBLIGATORIO

Mask	Network address	Next-hop address	Interface
/24	140.24.7.0	135.14.2.4	m0
/0	0.0.0.0	Default	m1

Routing table for R2

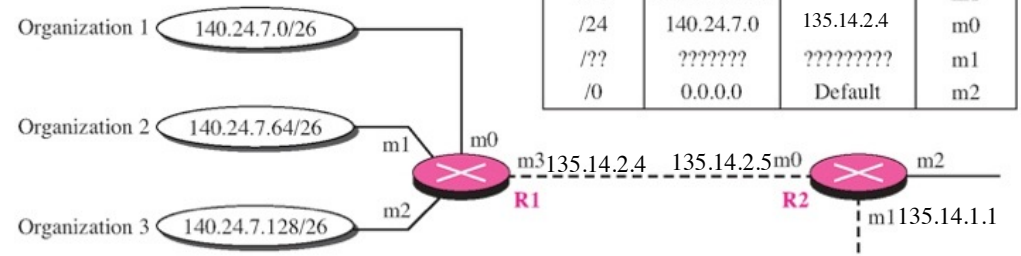
- Route aggregation with CIDR

IP Forwarding Revisited

un indirizzo può avere più prefissi, es. 171.69.10/16 e 171.69.10/24 → in questi casi il pacchetto viene messo in corrispondenza al prefisso più lungo

- It is also possible to have prefixes in the forwarding tables that overlap
  - Some addresses may match more than one prefix
- For example, in the forwarding table of a router we might find 171.69.0.0/16 and 171.69.10.0/24, in this order
  - A packet destined to 171.69.10.5 clearly matches both prefixes.
- Most routers choose following the principle of "longest match" (171.69.10,0/24 in this case) → scelgo quello con prefisso più lungo perchè so che sarà quello finale (tra i due) in quanto MAGGIORE è il prefisso MINORE è la dimensione della rete
- Some routers scan sequentially the forwarding table, and apply the first matching rule
  - In this case, a packet destined to 171.69.20.5 would match 171.69.0.0/16 and not 171.69.10.0/24

Longest match forwarding



Routing table for R2

Mask	Network address	Next-hop address	Interface
/26	140.24.7.192	135.14.1.2	m1
/24	140.24.7.0	135.14.2.4	m0
/??	???????	?????????	m1
/0	0.0.0.0	Default	m2

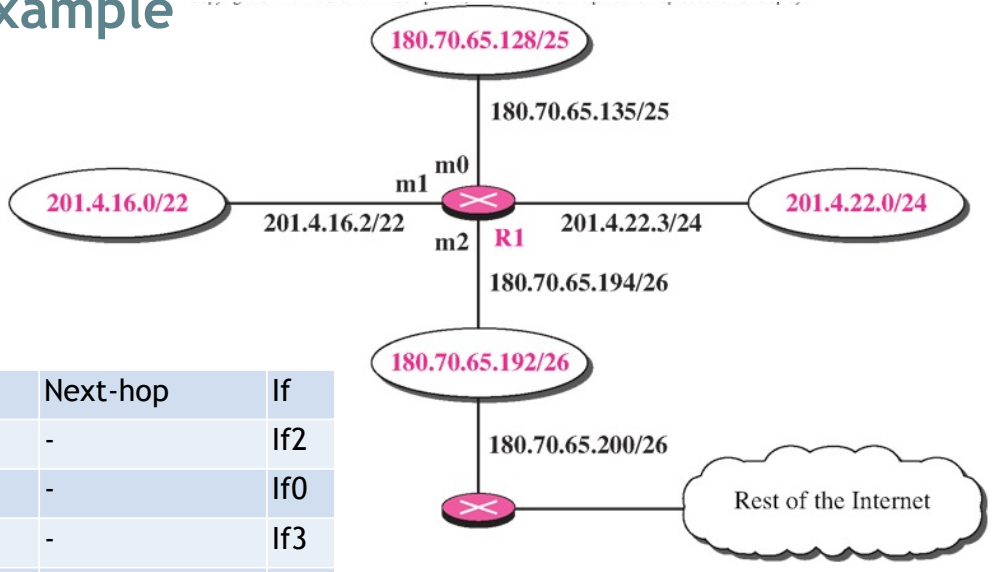
Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	m0
/26	140.24.7.64	----------	m1
/26	140.24.7.128	----------	m2
/0	0.0.0.0	135.14.2.5	m3

Routing table for R1

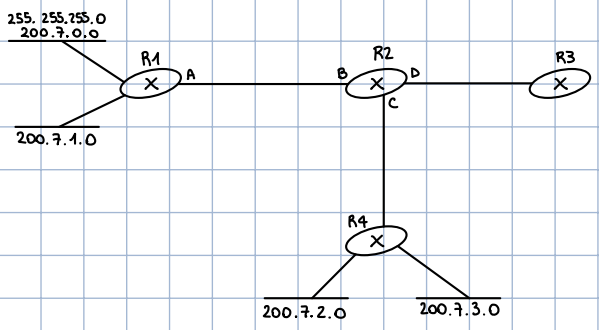
Mask	Network address	Next-hop address	Interface
/26	140.24.7.192	----------	m0
/??	???????	?????????	m1
/0	0.0.0.0	135.14.1.1	m2

Routing table for R3

Another example



Destination	Next-hop	If
180.70.65.192/26	-	If2
180.70.65.128/25	-	If0
201.4.22.0/24	-	If3
201.4.16.0/22	-	If1
*	180.70.65.200	if2



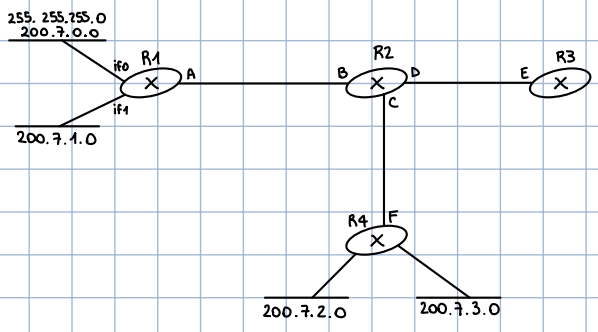
cambio solo l'ultimo bit, prefisso di 23 bit

R2	network	netw. mask	next hop
	200.7.0.0	255.255.254.0	A
	200.7.2.0	255.255.254.0	C

R3	network	netw. mask	next hop
	200.7.0.0	255.255.252.0	D

tolgo 3 perché 4 reti le rappresento con 3 bit (100)

Usando la notazione CIDR:



se sei collegato direttamente

R1	destination	next hop	interface
	200.7.0.0/24	—	if 0
	200.7.1.0/24	—	if 1
	0/0	B	—

R3	destination	next hop	interface
	200.7.0.0/22	D	—

primi 23 bit di prefisso uguali

R2	destination	next hop	interface
	200.7.0.0/23	A	—
	200.7.2.0/23	C	—
	0/0	E	—

R4	destination	next hop	interface
	200.7.2.0/24	—	if 0
	200.7.3.0/24	—	if 1
	200.7.0.0/23	C	—
	0/0	C	—

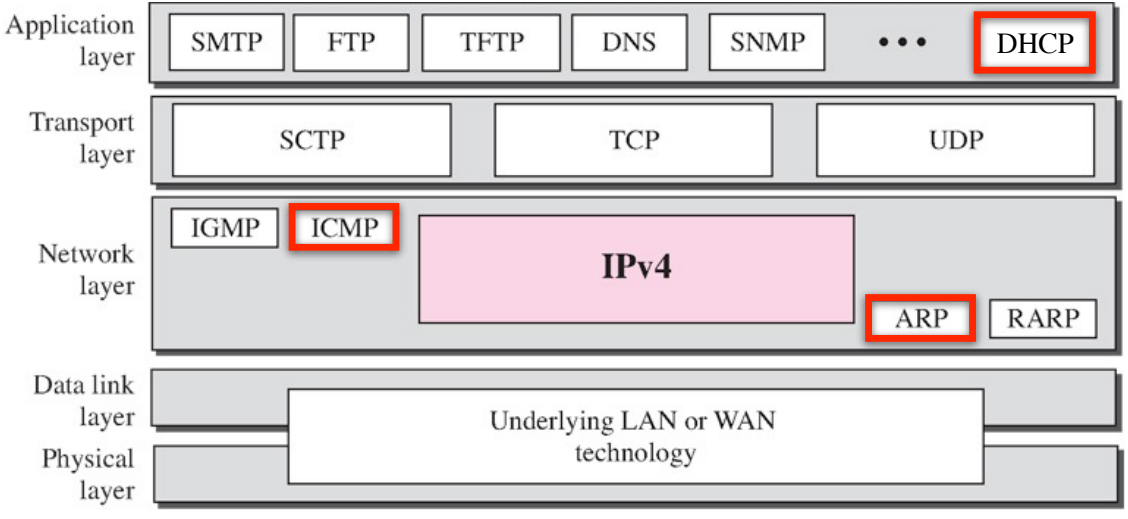
Special use IP addresses (RFC 5735)

↳ rete classe A "0" ⟶ inutilizzabile tutta questa rete

- **0.0.0.0/8**: non-routable, invalid address. Used to denote "any IP address"
- **10.0.0.0/8**: Used for private networks - not routable on the Internet
- **127.0.0.0/8**: Loopback (i.e. 'myself')
- **169.254.0.0/16**: autoconfiguration on a local link when DHCP not available
- **172.16.0.0/12, 192.168.0.0/16**: Used for private networks - not routable on the Internet
- **224.0.0.0/4**: Class D: Multicast addresses
- **240.0.0.0/4**: Class E: reserved for future uses (never actually used…)

IP Ancillary protocols

↳ protocolli che servono per far funzionare IP



Address Translation Protocol (ARP)

- Map IP addresses into physical addresses
  - destination host
  - next hop router

⇒ quando devo inviare un pacchetto all'host, dato che devo costruire un frame (scendendo nello stack) devo inserire oltre all'IP destinatario il MAC destinatario

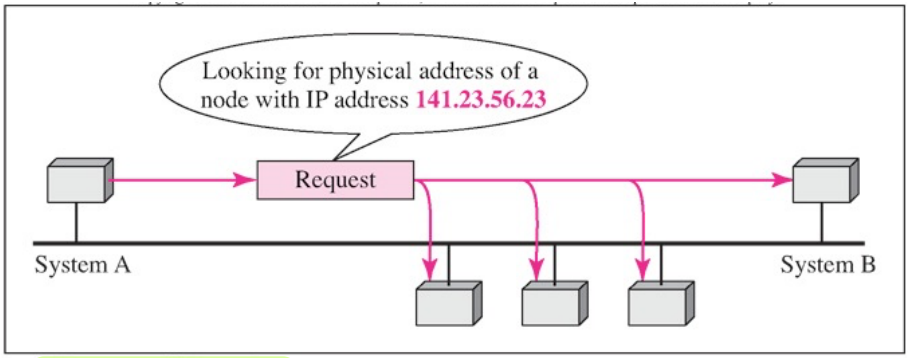
- Techniques
  - encode physical address in host part of IP address
  - table-based
    - Can be managed directly only for small networks

Logic Addr	Physic Address
192.44.80.1	FF:6E:A0:13:6C:A4
192.44.80.2	67:4A:6D:1A:B4:33
192.44.80.3	1A:27:44:8F:C2:14
192.44.80.4	1F:56:AB:C3:12:34
192.44.80.5	F2:65:29:04:72:60
192.44.80.6	2E:70:0E:A2:53:66
…	…

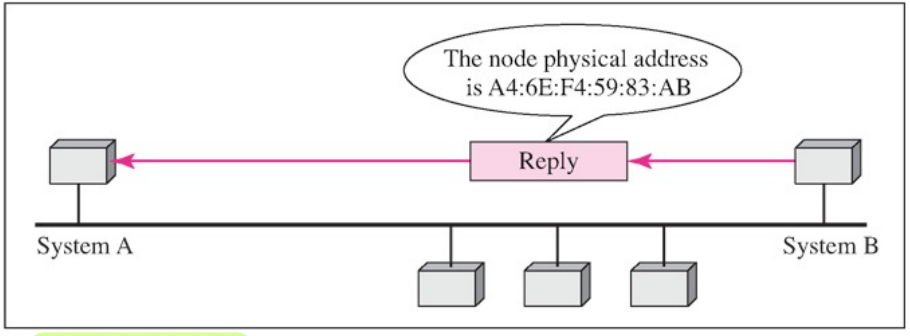
Address Translation Protocol (ARP)

- **ARP** (Address Resolution Protocol) → tabella contenuta in ogni host
  - handles automatically the table of IP to physical address bindings
  - broadcast request if IP address not in table (broadcast nella RETE LOCALE se nella mia tabella non ho l'IP (broadcast di liv.2 - FFFF...))
  - target machine responds with its physical address
  - table entries are discarded if not refreshed → time out di qualche minuto
  - invoked by IP module, when a datagram has to be delivered to a host of the same network, but the address is not present in ARP table
    - perché quando un frame arriva alla rete di destinaz. l'istantanea del frame è: MAC: quello del router, IP: quello dell host finale, perciò con ARP si trova il MAC dell'host destinatario e si reincapsula in un nuovo frame avente il MAC destinatario.

Mentre quando il mio host deve inviare un pacchetto, che passerà per il default gateway (quindi esce dalla sua rete), dato che per farlo passare sulla LAN deve "imbustarlo" in un frame, userà ARP per trovare il MAC del default gateway e come IP assegna quello finale dell'host

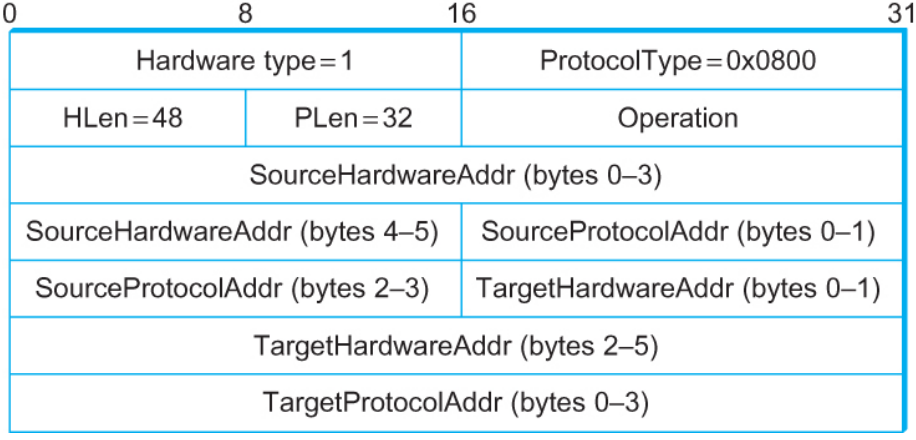


a. ARP request is broadcast



b. ARP reply is unicast

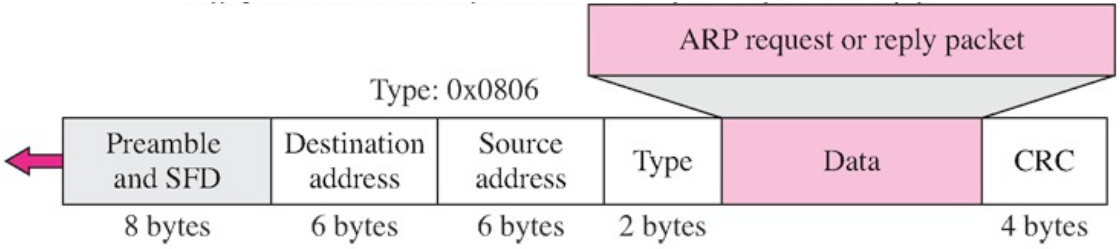
ARP Packet Format

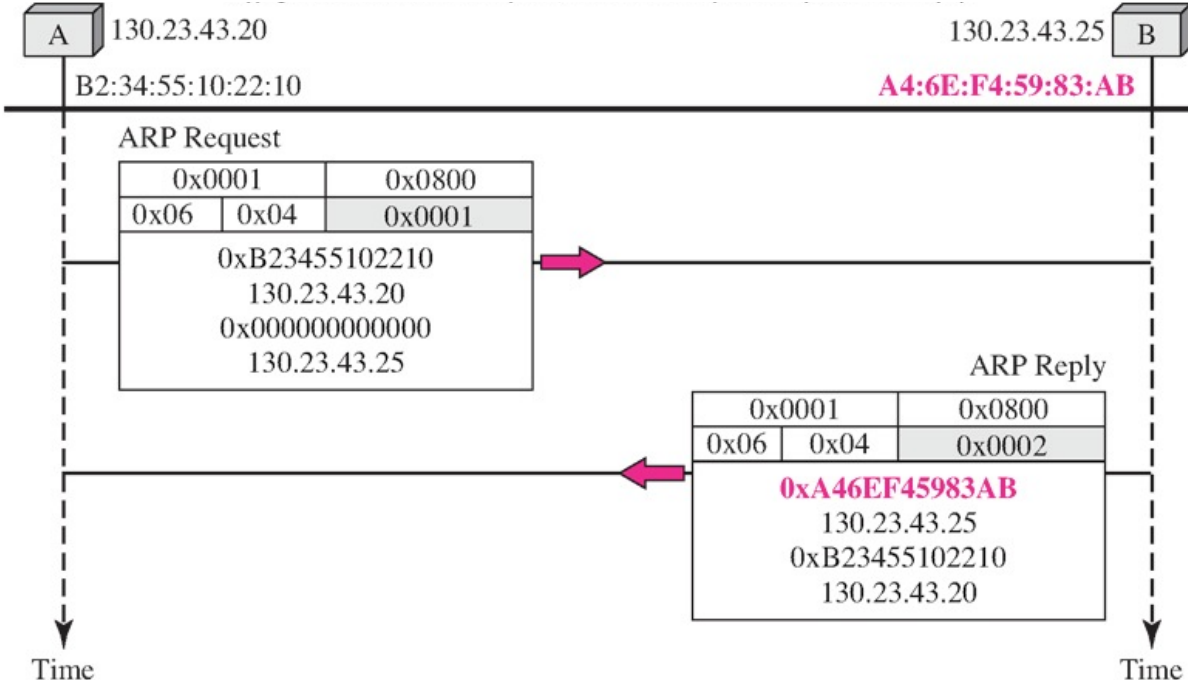


- HardwareType: type of physical network (e.g., Ethernet)
- ProtocolType: type of higher layer protocol (e.g., IP)
- HLEN & PLEN: length of physical and protocol addresses
- Operation: request or response
- Source/Target Physical/Protocol addresses
- TargetHardwareAddr is empty in request

Address Translation Protocol (ARP)

- ARP packets do not use IP (obviously)
- are directly encapsulated in datalink (e.g. Ethernet) frames

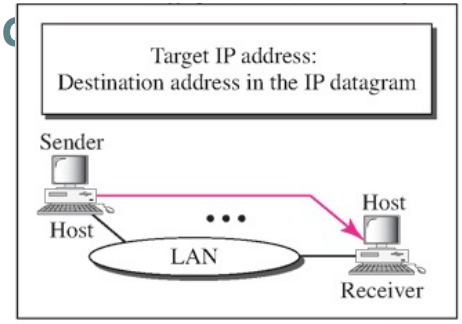



A
130.23.43.20
B2:34:55:10:22:10
130.23.43.25
B
A4:6E:F4:59:83:AB
ARP Request
0x0001
0x0800
0x06
0x04
0x0001
0xB23455102210
130.23.43.20
0x000000000000
130.23.43.25
ARP Reply
0x0001
0x0800
0x06
0x04
0x0002
0xA46EF45983AB
130.23.43.25
0xB23455102210
130.23.43.20
Time
Time

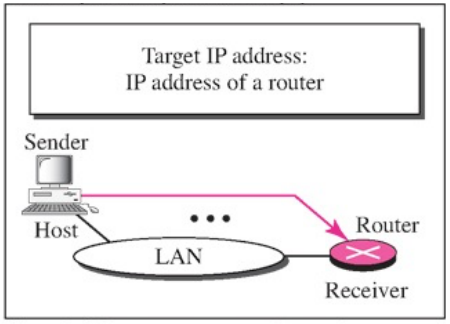
Address Translation Protocol (ARP)

- Which IP address should we resolve?
  - for direct delivery: the destination's
  - for indirect delivery: next hop's IP address → Se devo inviare un pacchetto ad un host che non è nella mia rete, invio un ARP request per sapere il MAC del next hop
  - thus **IP datagram is encapsulated in a datalink frame whose MAC address may be not that of destination**
  - the MAC address changes at each link the packet traverses, while the IP address remains unchanged

Which add



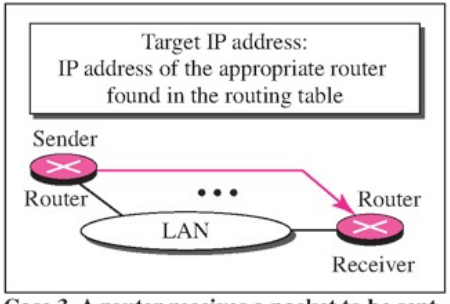
Case 1. A host has a packet to send to another host on the same network.



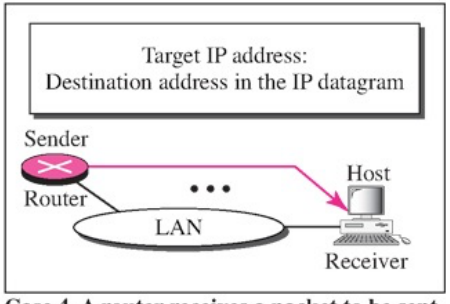
Case 2. A host wants to send a packet to another host on another network. It must first be delivered to a router.

Se deve passare per qualche router prima di arrivare a destinazione, allora il MAC destinatario è quello del router in mezzo, e l'IP di DESTINAZ. è quello dell'host finale
↳ questo rimane fisso, ad ogni hop invece si cambia l'indirizzo MAC

Quindi le informaz. che bastano sono IP, submet, gateway



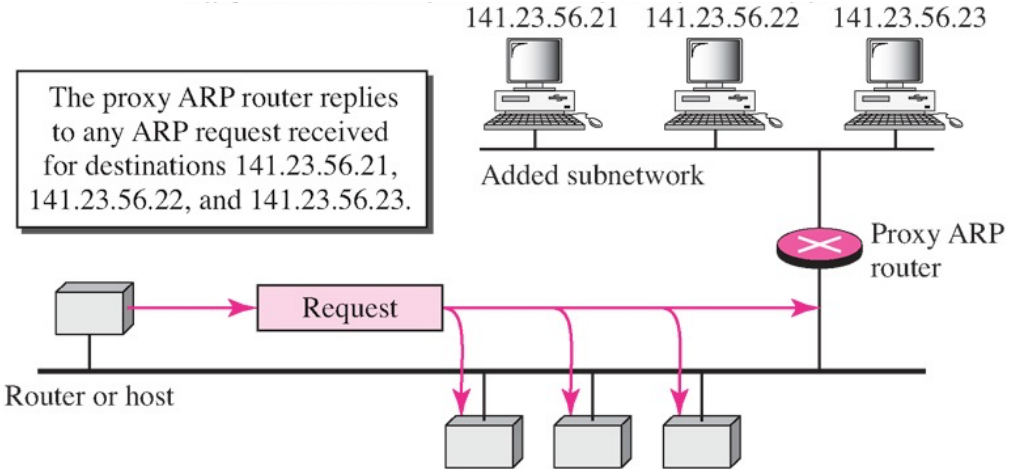
Case 3. A router receives a packet to be sent to a host on another network. It must first be delivered to the appropriate router.



Case 4. A router receives a packet to be sent to a host on the same network.

Proxy ARP

- A proxy ARP is a host representing other hosts (often entire networks)
- it answers with its own MAC to ARP requests for the "hidden" hosts
- it forwards the frames it receives to the proper destinations, like a switch (but changing MAC address)

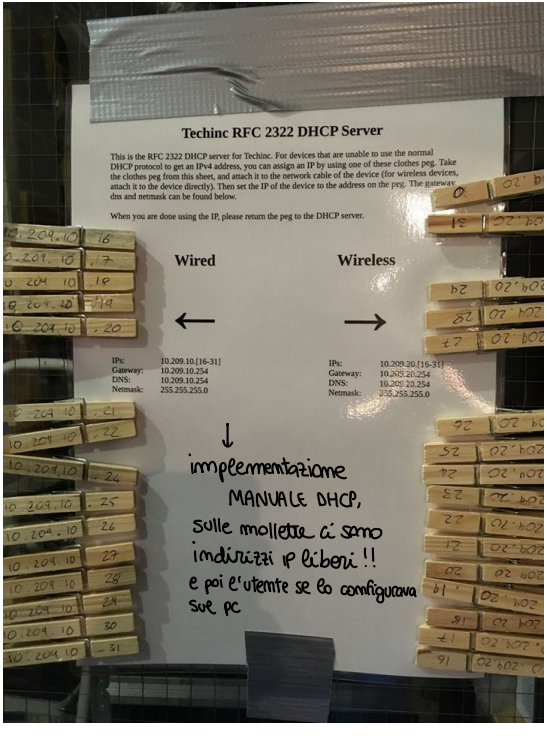


Host Configurations

- Ethernet addresses are configured into network by manufacturer and they are unique
- IP addresses must be unique on a given internetwork but also must reflect the structure of the internetwork
- Most host Operating Systems provide a way to manually configure the IP information for the host
- Drawbacks of manual configuration
  - A lot of work to configure all the hosts in a large network
  - Configuration process is error-prone
- Automated Configuration Process is required

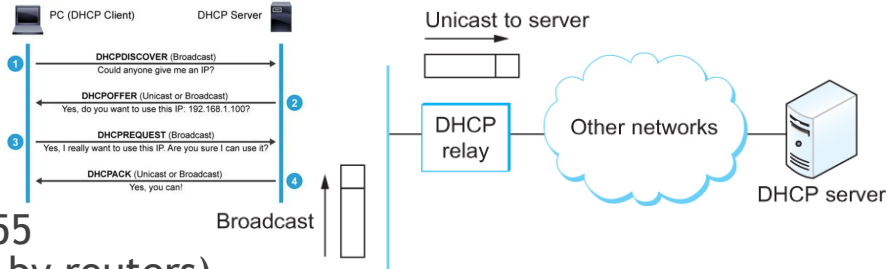
Dynamic Host Configuration Protocol (DHCP)

- DHCP server is responsible for providing configuration information to hosts
  - Other solutions (obsoleted): RARP, BOOTP
- For automatic configuration, at least one DHCP server for an administrative domain
- DHCP server maintains a pool of available addresses, which are assigned to hosts
  - statically (e.g. based on MAC addresses)
  - dynamically (time-limited “leases”)



DHCP

è a TIME LEASE



- Newly booted or attached host sends DHCPDISCOVER message
  - destination IP: 255.255.255.255 ("limited broadcast", ignored by routers)
  - source IP: irrelevant. Message contains also MAC address of source host
- (if present) DHCP relay agent unicasts the message to DHCP server and waits for the response

non è detto che per ogni rete esista un DHCP server, esiste però il DHCP Relay che viene configurato con un'unica informazione: IP del DHCP SERVER. Quando il relay riceve un DHCPDISCOVERY lo inoltra in unicast al server DHCP, attende e inoltra la risposta all'host

- DHCP server's answer contains the assigned IP address (con anche gateway e subnet mask)
- this message is delivered to the source host (using its MAC, not its IP)
- the IP layer of the host delivers it to the DHCP client, which sets the local IP address (and other parameters, like gateway, DNS, etc.) on the host

=> se DHCP non funziona allora si autoconfigura con 169.254.0.0 (APIPA)

Network signalling and management: Internet Control Message Protocol (ICMP)

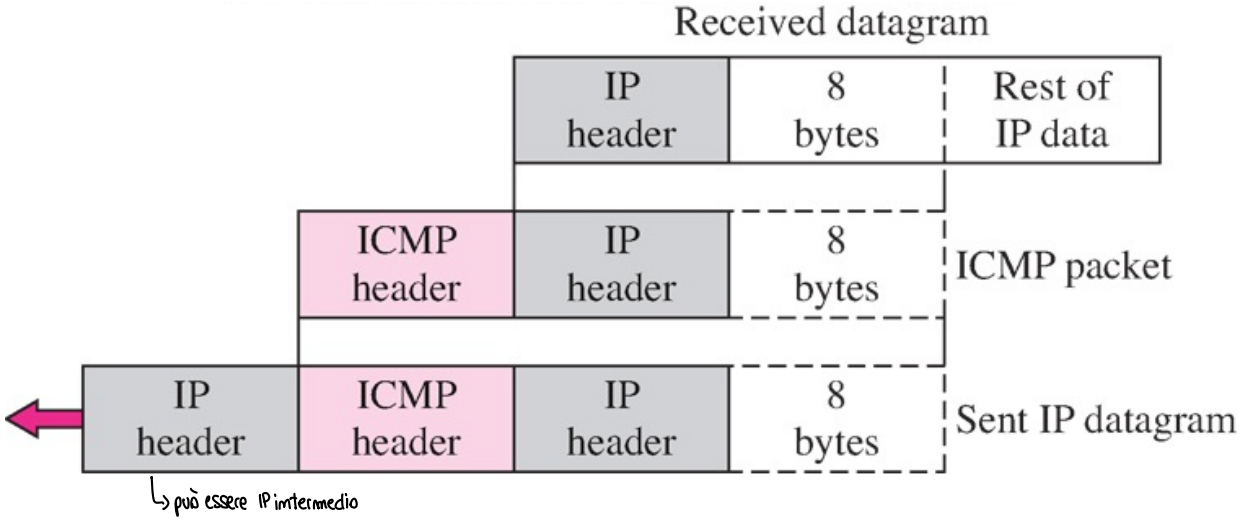
- Defines a collection of error and notification messages that are sent back to the source host
  - whenever a router or host is unable to process an IP datagram successfully
    - Destination host unreachable due to link /node failure
    - Reassembly process failed
    - TTL had reached 0 (and datagram has been discarded)
    - IP header checksum failed
  - when there is better route for that destination
    - From router to the source host (ICMP-Redirect)
- In all cases, only the source host is notified

→ processare un pacchetto

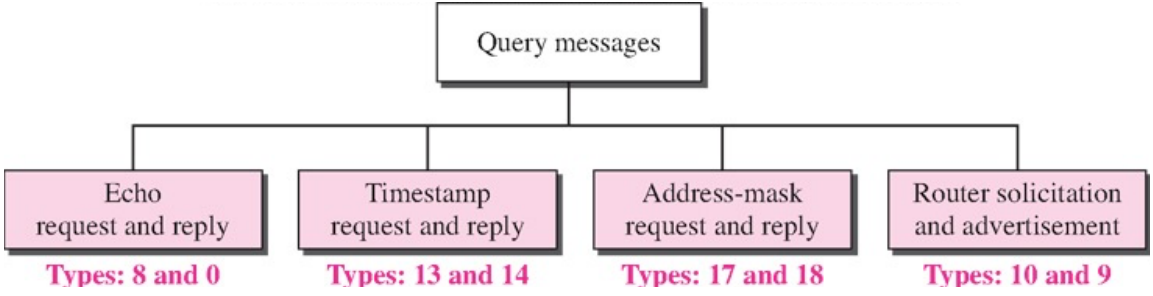
↳ non è detto poi che il pacchetto di notifica (ICMP) arrivi, ad esempio se l'ip del mittente si dammeggia l'ICMP non arriva

Internet Control Message Protocol (ICMP)

- The initial part of the rejected datagram is included in the ICMP message, for facilitating analysis and countermeasures



ICMP request messages



- Echo ("ping")
  - check whether a host is reachable
- Timestamp
  - for measuring round trip time (RTT)
- Mask discovery, router discovery
  - implemented by DHCP, nowadays

Ping

- Sends several ICMP Echo requests
  - computes round trip time

```
└──╼ ping www.google.com
PING www.google.com (216.58.198.36): 56 data bytes
64 bytes from 216.58.198.36: icmp_seq=0 ttl=54 time=25.934 ms
64 bytes from 216.58.198.36: icmp_seq=1 ttl=54 time=46.239 ms
64 bytes from 216.58.198.36: icmp_seq=2 ttl=54 time=25.844 ms
64 bytes from 216.58.198.36: icmp_seq=3 ttl=54 time=25.701 ms
64 bytes from 216.58.198.36: icmp_seq=4 ttl=54 time=27.354 ms
64 bytes from 216.58.198.36: icmp_seq=5 ttl=54 time=46.450 ms
64 bytes from 216.58.198.36: icmp_seq=6 ttl=54 time=29.715 ms
64 bytes from 216.58.198.36: icmp_seq=7 ttl=54 time=26.358 ms
64 bytes from 216.58.198.36: icmp_seq=8 ttl=54 time=26.206 ms
64 bytes from 216.58.198.36: icmp_seq=9 ttl=54 time=26.120 ms
64 bytes from 216.58.198.36: icmp_seq=10 ttl=54 time=27.245 ms
^C
--- www.google.com ping statistics ---
11 packets transmitted, 11 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 25.701/30.288/46.450/7.646 ms
```

Traceroute

```
prompt> traceroute fhda.edu
traceroute to fhda.edu (153.18.8.1), 30 hops max, 38 byte packets
1Dcore.fhda.edu          (153.18.31.254)   0.995 ms 0.899 ms 0.878 ms
2Dbackup.fhda.edu        (153.18.251.4)    1.039 ms 1.064 ms 1.083 ms
3tiptoe.fhda.edu         (153.18.8.1)      1.797 ms 1.642 ms 1.757 ms
```

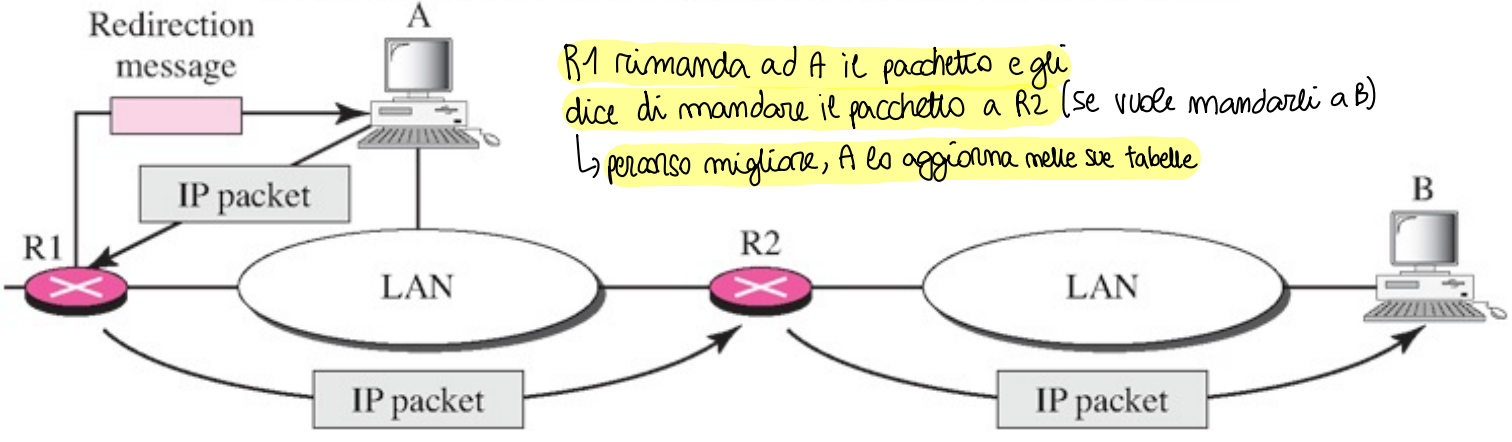
- Useful tool for discovering the route to an host
- Uses ICMP error messages and TTL → mando più ICMP ciascuno con un TTL incrementale per trovare il percorso che fa il pacchetto, ma dato che è commutaz. di pacchetto (e non circuito) NON È PER FORZA IL PERCORSO che viene seguito dagli ultimi. es. se sono arrivato al pacchetto TTL=5, non è detto che questo pacchetto ICMP passi per lo stesso percorso di TTL=4. → NON È UN CIRCUITO FISSO, bensì un possibile percorso
  - Sends a message with TTL=1
    - First router drops it, and send the sender an ICMP Error
    - Thus the sender discovers the address of first router
  - Sends a message with TTL=2
    - First router decrements TTL, and forwards it to next router
    - Second router drops it, and send the sender an ICMP Error
    - Thus the sender discovers the address of second router
- … etc.

Traceroute

- This technique does not work for destination host
  - even if TTL=0, the packet has arrived and no ICMP error is sent back
- For the last step, an UDP packet on port 1 is sent
  - port 1 is not valid, hence it generates an ICMP error of “port not reachable”

ICMP-Redirect

- Router R1 receives a datagram for a destination for which there is a better route
- datagram is delivered, but source is notified about the right route
- the host updates its route table accordingly



R1 rimanda ad A il pacchetto e gli dice di mandare il pacchetto a R2 (se vuole mandarli a B)
↳ percorso migliore, A lo aggiorna nelle sue tabelle

Private IP Networks and NAT

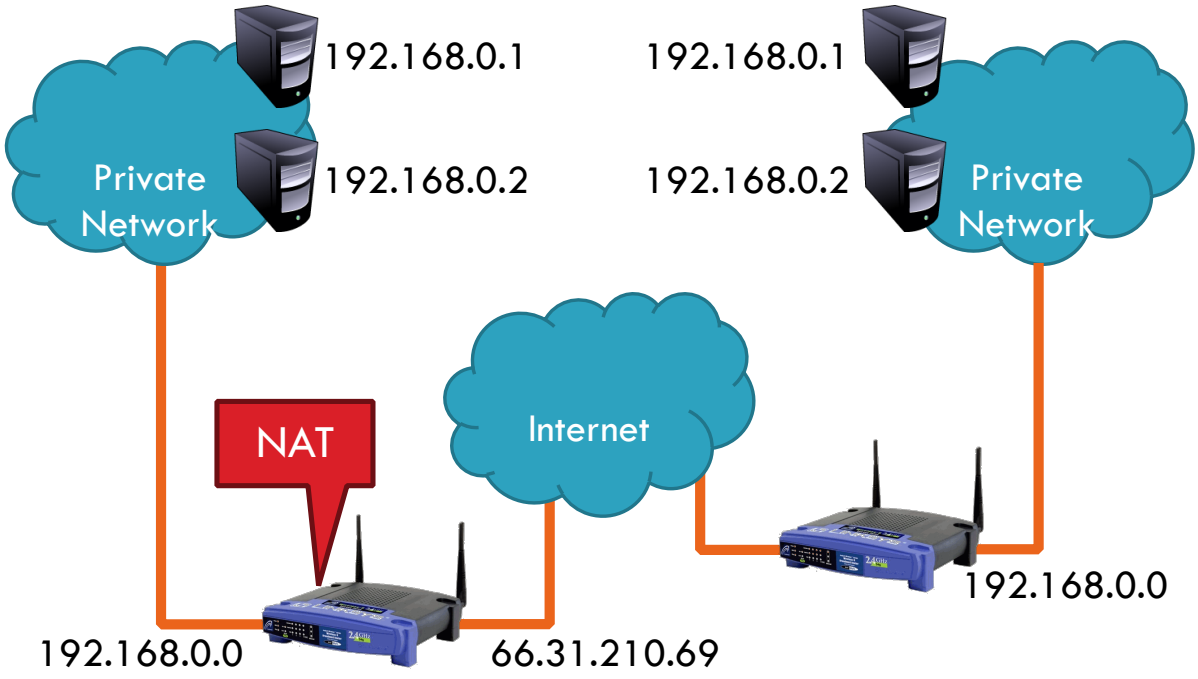
- Idea: create a range of private IPs that are separate from the rest of the network → visto che gli IPv4 stanno terminando, ISP ti assegnano un singolo IPv4 pubblico (es. 171.69.10.246) e poi con il NAT ti crei la rete privata di casa
  - Use the private IPs for internal routing
  - Use a special router to bridge the LAN and the WAN
- Properties of private IPs
  - **Not globally unique**
  - Usually taken from non-routable IP ranges (why?)
- Typical private IP ranges
  - 10.0.0.0/8, that is, 10.0.0.1 – 10.255.255.255
  - 172.16.0.0/12, that is, 172.16.0.1 – 172.31.255.255
  - 192.168.0.0/16, that is, 192.168.0.0 – 192.168.255.255

così facendo i router NAT/PAT non devono solo occuparsi di fare routing, ma anche di camuffare tutti i pacchetti uscenti ed entranti, ovvero deve intercambiare IP pubblico e IP privato

↳ OPERAZIONI CHE RALLENTANO IL THROUGHPUT
- deve scandire tutta la tabella
- deve ricalcolare il CHECKSUM quando cambia l'indirizzo

Private Networks

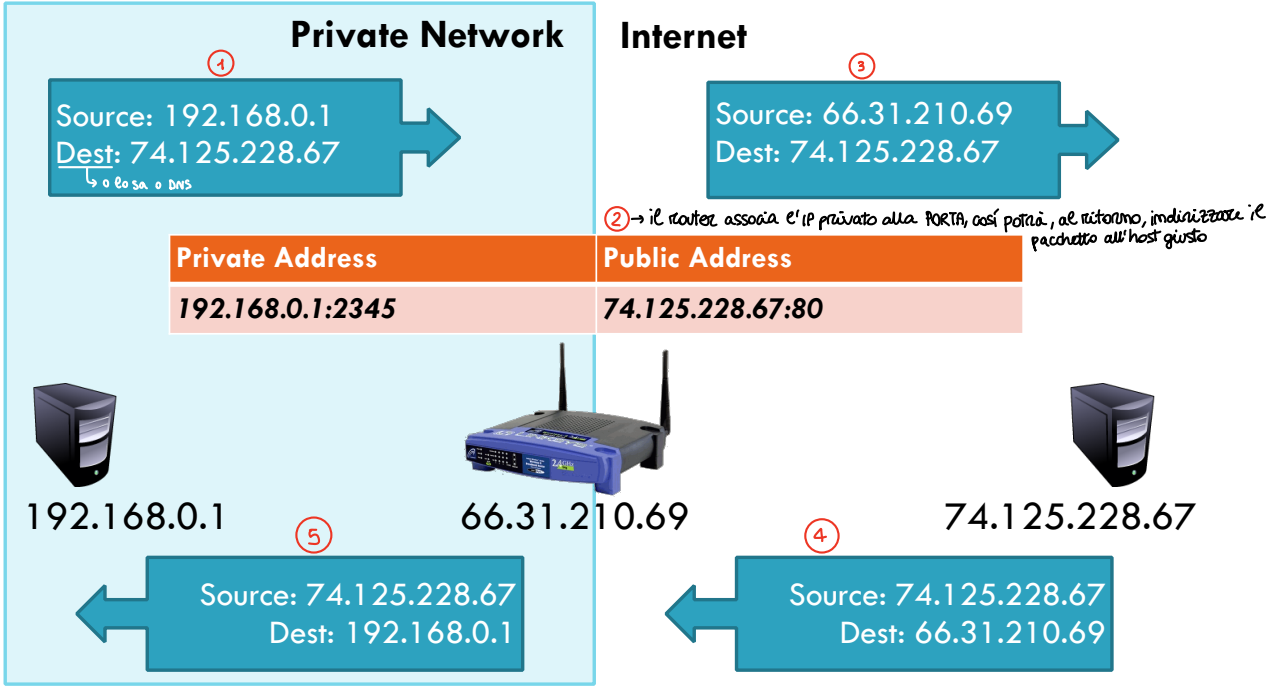
i router scartano gli IP privati e i router NAT cambiano l'IP privato in pubblico, oltre al TTL (come i router normali)



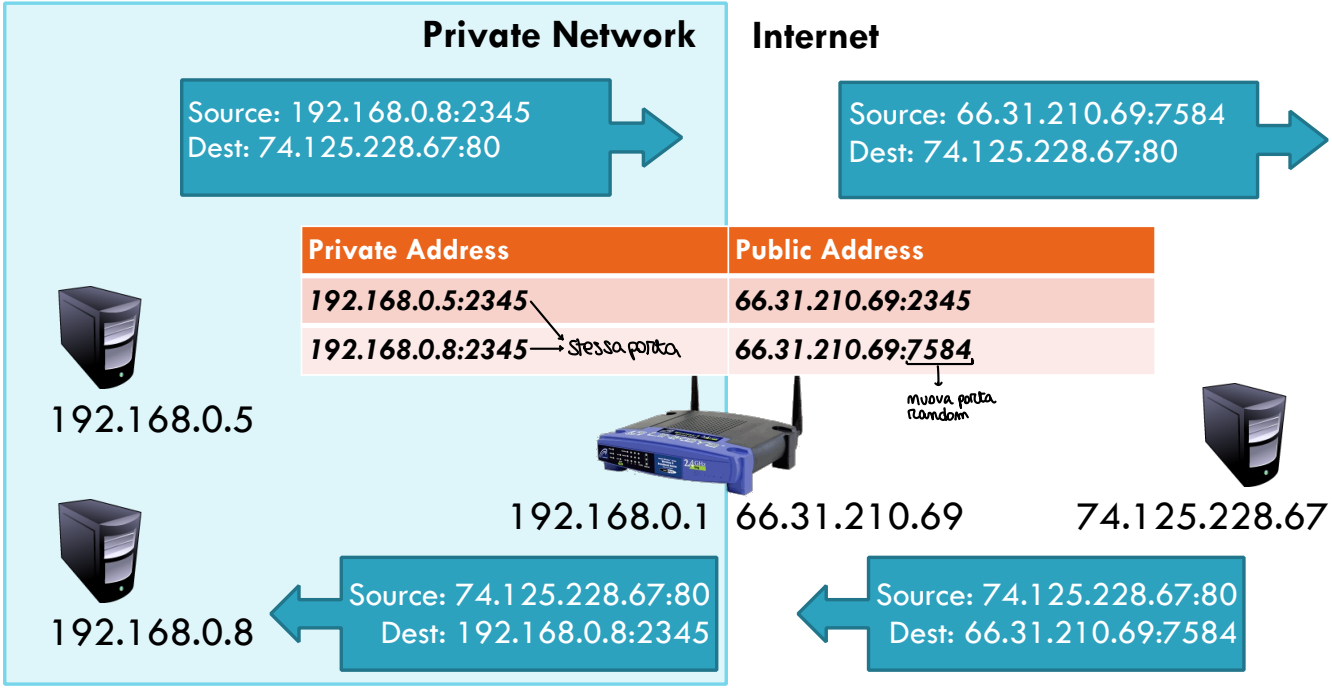
Network Address Translation (NAT) with Port Mapping

- NAT allows hosts on a private network to communicate with the Internet
  - Warning: connectivity is not seamless
- Special router at the boundary of a private network
  - Replaces internal IPs with external IP
    - This is "**Network Address Translation**"
  - May also replace TCP/UDP port numbers
    - This is "**Port Mapping**"
- The router has to maintain a table of active flows
  - Outgoing packets initialize a table entry
  - Incoming packets are rewritten based on the table

Basic NAT Operation



Basic NAT Operation (with PAT)

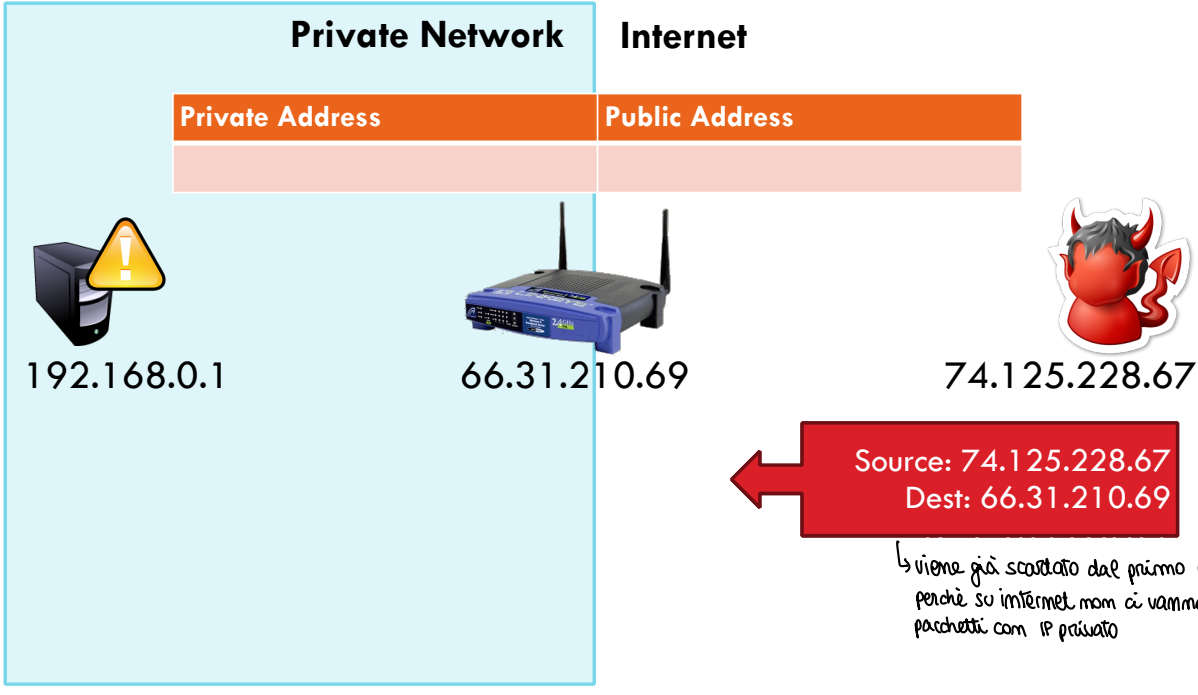


Applications of NATs

- Allow multiple hosts to share a single public IP
- Allow migration between ISPs
  - Even if the public IP address changes, you don't need to reconfigure the machines on the LAN
- Load balancing
  - Forward traffic from a single public IP to multiple private hosts

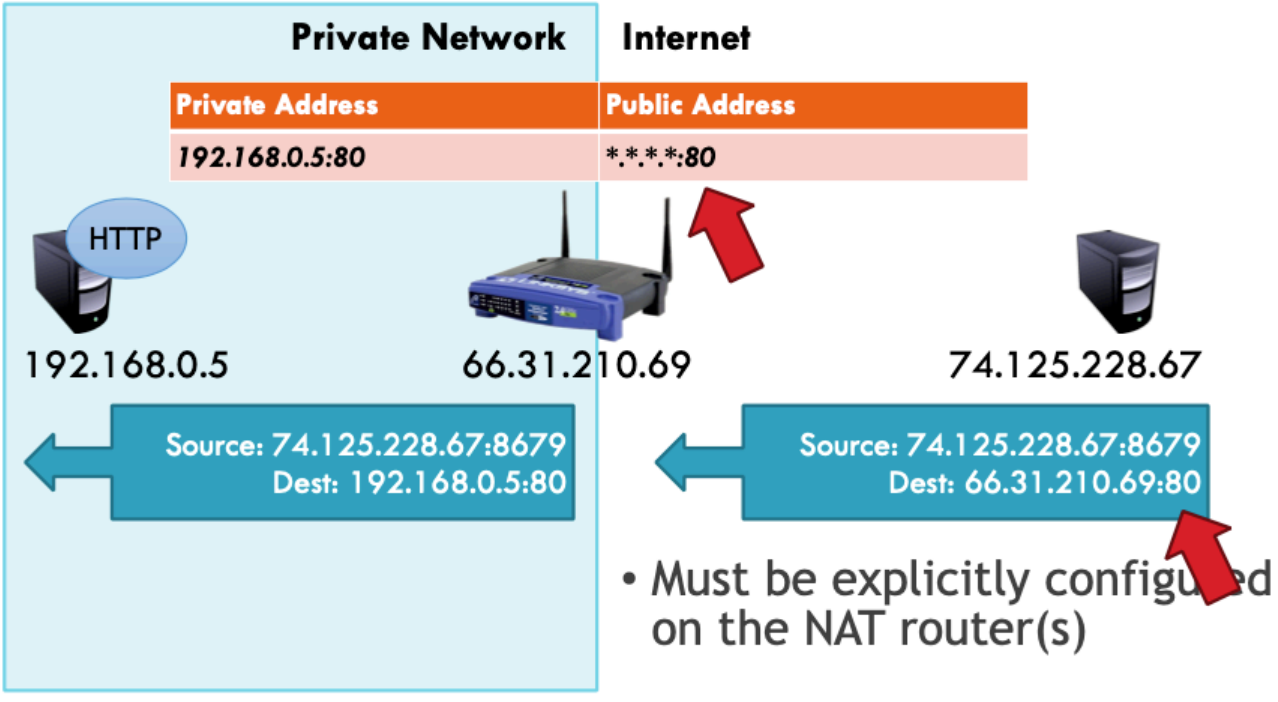
Natural "Firewall"

il NAT serve anche per limitare la visibilità degli host, anche se è semplice da sorpassare un NAT



↳ viene già scartato dal primo router se DEST = 192.168.0.5 perché su internet non ci vanno pacchetti con IP privato

Port Forwarding → crea "buchi" di sicurezza



Hole Punching

- Problem: How to enable connectivity through NATs?

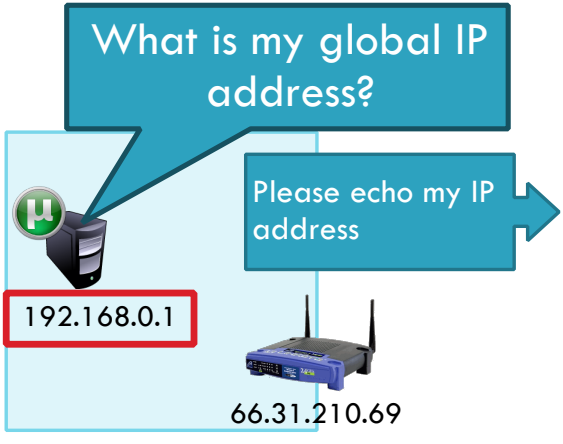
↳ - o apro le porte (non è detto che si possano aprire sempre le porte, tipo nelle grosse aziende non si fa)
- o STUN e TURN

- Two application-level protocols for hole punching
  - STUN
  - TURN

STUN

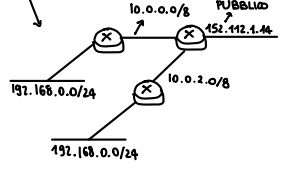
- **S**ession **T**raversal **U**tilities for **N**AT
  - Use a third-party to echo your global IP address
  - Also used to probe for symmetric NATs/firewalls
    - i.e. are external ports open or closed?

ti dice il tuo indirizzo pubblico



possono essere: - NAT LOCALI
- NAT PROVIDER: ovvero applicano un NAT all'indirizzo pubblico costruendo un altro strato privato

il PORT FORWARDING è inutile se ci sono più strati



Your IP is 66.31.210.69

STUN Server

↳ ti ritorna l'ultimo indirizzo IP del router, ovvero il tuo indirizzo pubblico

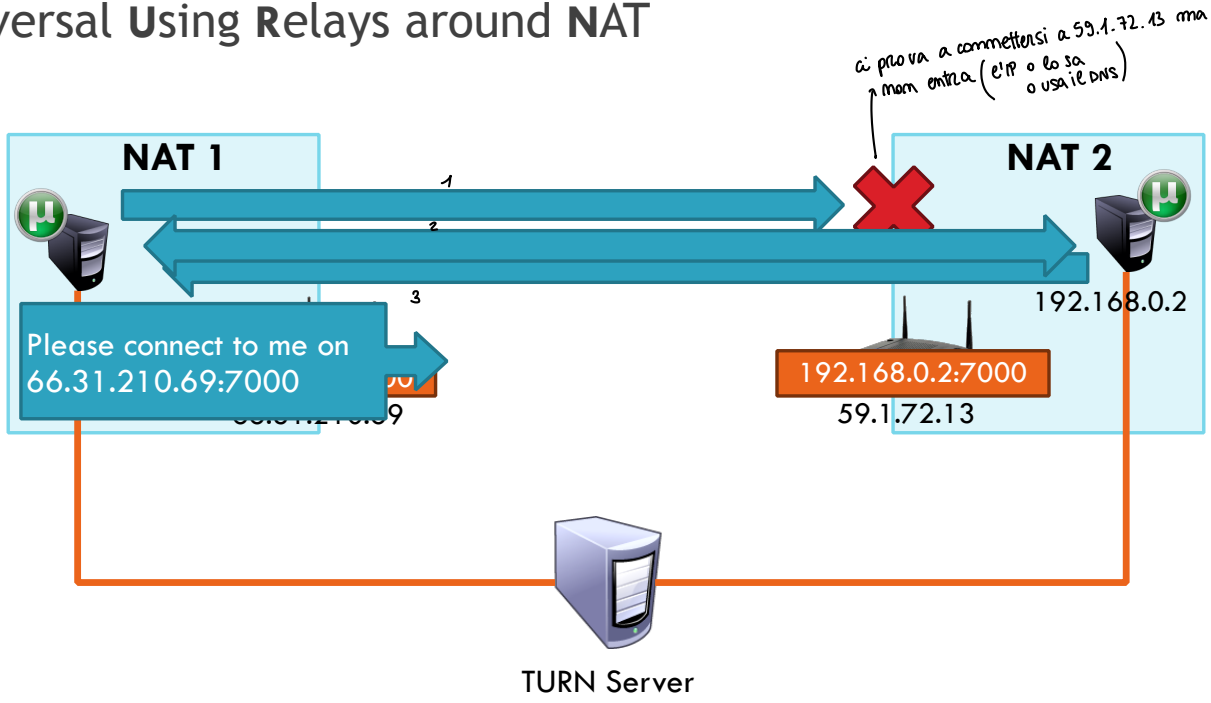
Problems With STUN

- Only useful in certain situations
  - One peer is behind a symmetric NAT → ovvero l'altro usa PORT FORW. altrimenti non va
  - Both peers are behind partial NATs
- Not useful when both peers (clients) are fully behind full NATs
  - ↳ il pacchetto può anche arrivare al router finale (es. 59.1.72.13) ma se non sono configurate le tabelle NON ENTRA, ovvero come fa a sapere qual è la porta dell'host nel NAT 2? Di solito funziona se faccio comunicazione NAT → non NAT

TURN

usa STUN

- **T**raversal **U**sing **R**elays around **N**AT



Concerns About NAT

- Performance/scalability issues
  - Per flow state!
  - Modifying IP and Port numbers means NAT must recompute IP and TCP checksums
- Breaks the layered network abstraction → perché il NAT lavora sia a liv.3 che liv.4
- Breaks end-to-end Internet connectivity
  - 192.168.*.* addresses are private
  - Cannot be routed to on the Internet
  - Problem is worse when both hosts are behind NATs
- Overall, NAT is a "necessary evil"
  - It is not needed if we have enough public IP address
  - Network separation can be obtained using firewalls

Next Generation IP (IPv6)

IPv4 finiti nel 2019

- 128-bit addresses
  - Address space in the order or $3*10^{38}$.
  - As a comparison: Earth surface = $5,1*10^{14}$ m^2 = $5,1*10^{20}$ mm^2, so $0,58*10^{18}$ addresses for each mm^2
- Multicast
- Real-time service
- Authentication and security → non è per forza integrato IPv4 (eredita IPsec da IPv6)
  una macchina IPv6 deve per forza avere servizi di sicurezza
- Auto-configuration
- End-to-end fragmentation
- Enhanced routing functionality, including support for mobile hosts
  ↳ ovvero posso mantenere il mio IP se mi sposto da una rete all'altra
- Extensible → si possono aggiungere funzionalità al bisogno

IPv6 Addresses

- Classless addressing/routing (similar to CIDR)
- Notation: x:x:x:x:x:x:x:x (x = 16-bit hex number)
  - contiguous 0s are compressed: 47CD::A456:0124
  - IPv6 compatible IPv4 address: ::128.42.1.87
- Address assignment
  - provider-based
  - geographic

diverse da quelle MAC: 80:A4:37:AF:10:32
IPv6: FEA0:07AC:0000:ABCD:FEAC:07FA:0101:0000
↳ 128 = 16 · 8

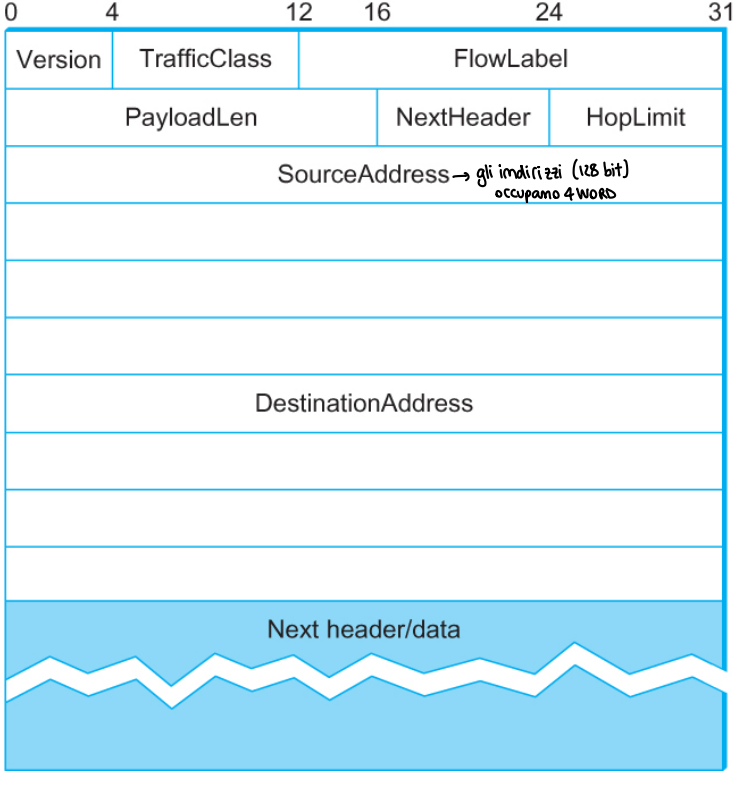
es :: sono tutti 0 in mezzo
::1 → 00...:...:0001
↳ SELF-ADDRESS

↳ es. indirizzi dell'Europa, al cui interno ha anche indirizzi dell'Italia ecc ecc

IPv6 Header

- 40-byte "base" header (dim. minima dell'intestazione (header); pensata per essere modulare, hanno spostato molte cose non strettamente necessarie nelle opzionali)
- 64Kbyte payload (with extensions)
- Extension headers (fixed order, mostly fixed length)
  - fragmentation
  - source routing (info extra per specificare che strada deve fare il mio pacchetto (ovvero per che router passerà))
  - authentication and security
  - and many other

di default non c'è la frammentazione, è un opzione



Priority

- *Traffic Class* defines the priority of a datagram with respect to others of the same source → suggerimento che diamo ai router per aiutarlo ad instradare
- Useful in case some datagrams have to be dropped due to congestions
- Possible values:
  - 0=Generic traffic → "ok router, scegli pure tu come instradare"
  - 1=Background traffic (e.g. NNTP) → "traffico per cui mom c'è nessuno in attesa, ASINCRONO", bassa priorità (es. update che faccio alla sera)
  - 2=Traffic without waiting (e.g. SMTP) → "traffico senza attesa", se c'è qualche ritardo mom importa
    ↳ protocollo MAIL
  - 3 e 5=reserved
  - 4=Traffic with waiting (e.g. FTP, HTTP) → "traffico in cui c'è qualcuno che aspetta", il tempo conta
  - 6=Interactive traffic (e.g. TELNET, SSH) → "traffico interattivo, c'è una persona fisica che aspetta", deve avere latenza bassissima
  - 7=Control traffic (OSPF, RIP, SNMP,...)
  - 8-15=Traffic not depended on congestion (it should not be dropped), ordered by redundancy (e.g. audio/video)
    → più è alto più è prioritario (es. streaming)

Flow Label → etichetta di flusso, per considerare come appartenenti alla stessa conversazione un insieme di pacchetti

- 20 bit casual number, assigned by source
- Datagrams belonging to the same flow are labelled with the same value
- This label can be used by routers for implementing a coherent service to all datagrams of the same flow. E.g.:
  - same routing
  - Quality of Service: usage of reserved resources (previously reserved by means of other protocols, e.g. RSVP)
    - useful for soft real-time

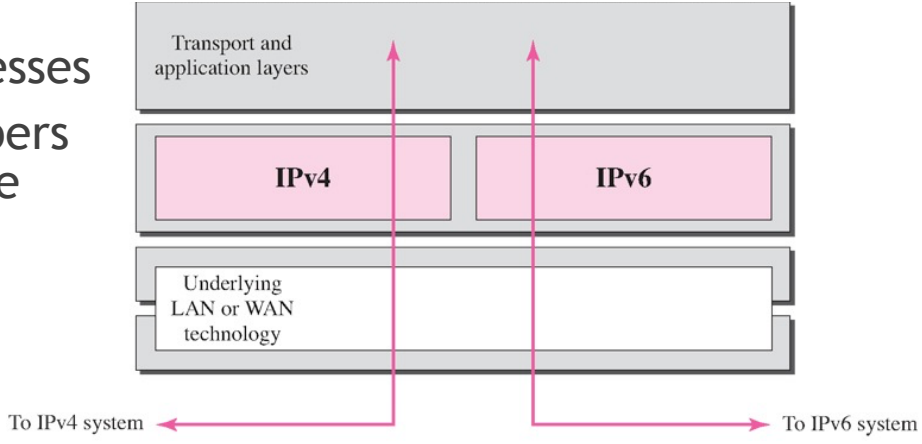
IPv4 to IPv6 transition

↳ basta solo vedere le intestazioni diverse dei pacchetti

- IPv4 and IPv6 are not compatible
- Eventually IPv4 should be replaced by IPv6
- Very hard. Too many hosts to update
  - Cannot be done at once
  - Cannot be forced “by law”
- 3 techniques to support transition
  - Dual protocol stack → macchine che implementano IPv4 e IPv6, quando inizio la comunicazione decido se usare IPv4 e IPv6
  - Tunneling
  - Header translation

Dual protocol stack

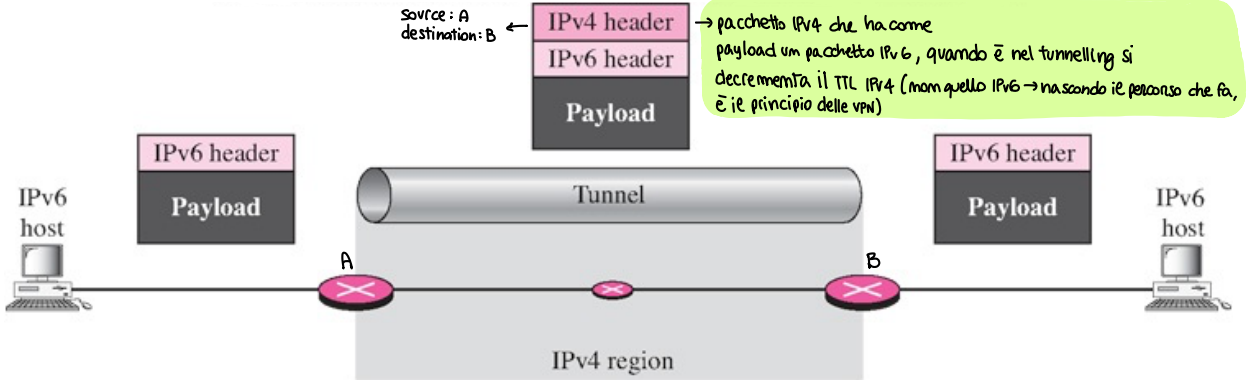
- A host (or router) can implement both protocols
  - Used in most modern operating systems
- Two protocols at level 3: the host belongs to two separated internetworks
- Applications have to deal explicitly with both addresses
  - more work for developers and deployers (e.g. see Apache httpd.conf)



Tunneling

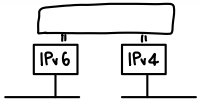
- Create "tunnels" or "bridges" from one IPv6 region to another, incapsulating IPv6 datagrams in IPv4 datagrams
- Managed transparently by routers
- Hosts can communicate in IPv6
- Benefits of IPv6 are lost on the IPv4 region

⇒ causa più OVERHEAD: spreca più spazio "inutilmente" a causa di intestazione più grande

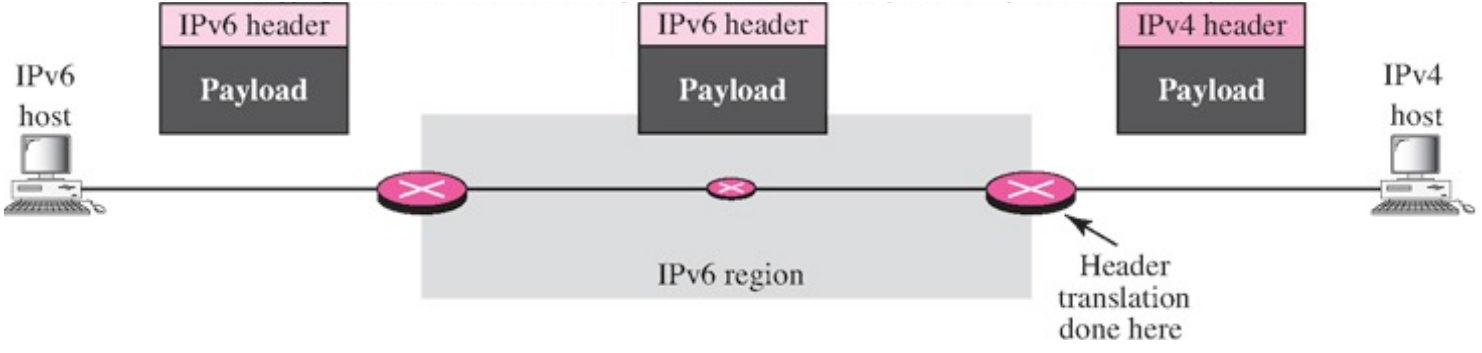


Header translation

=> BRIDGE di liv. 3



- Allows to transmit an IPv6 datagram to an IPv4-only host
- Destination router replaces IPv6 header with IPv4 “equivalent”, and vice versa
- Similar to NAT - but translation is between IPv4-IPv6 addresses and headers

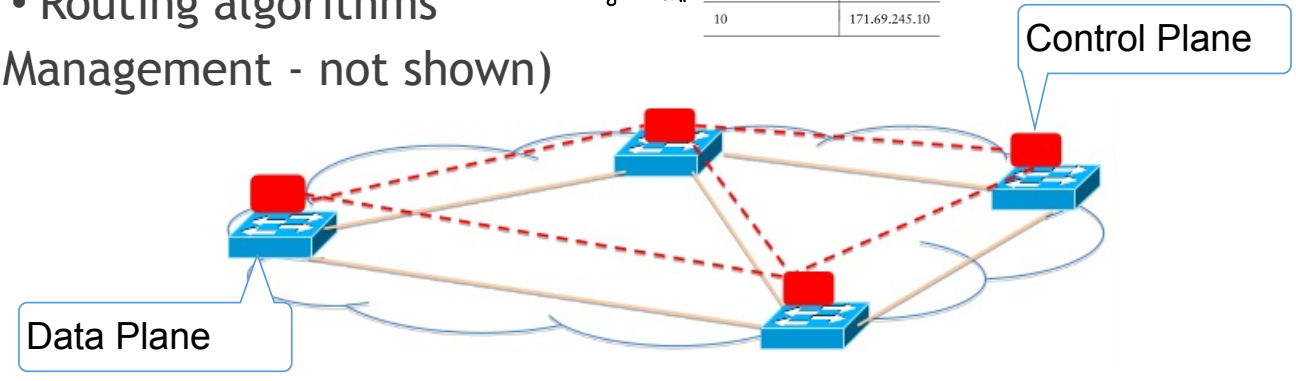


ROUTING

- Data flow is controlled by switches and routers and contains the following basic elements:
  - Data Forwarding Plane → gestisce le tabelle di forwarding (hanno gli indirizzi MAC)
    - algorithms for packet forwarding
  - Control Plane → gestisce le tabelle di ROUTING (hanno gli ind. IP)
    - Routing algorithms
  - (Management - not shown)

Network Number	Interface	MAC Address
10	if0	8:0:2b:e4:b:1:2

Network Number	NextHop
10	171.69.245.10



Forwarding versus Routing Algorithms

parallelismo con guidare la MACCHINA:
- decidere la destinaz. e il percorso → ROUTING
- cambiare, fermarsi, accelerare, ecc → FORWARDING

- **Forwarding**: → inoltrare velocemente
  - to select an output port for each packet, based on destination address and forwarding table
  - must be simple and fast
  - possibly implemented in hardware
- **Routing**: (costruire le tabelle di forwarding a seguito delle informazioni che raccoglie)
  - process by which routing table is built
  - routing table is a precursor of forwarding table
  - run as a background process (in user space)
  - can be complex

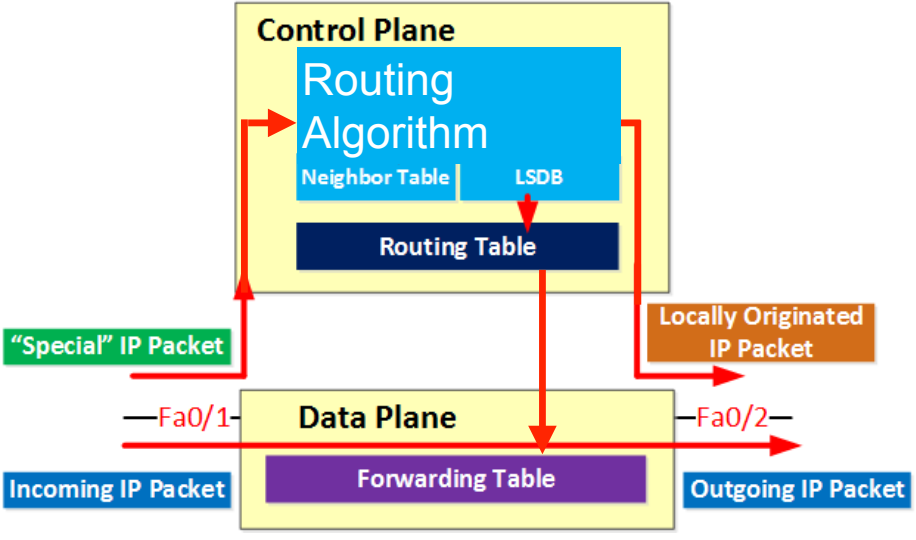
TABELLE INSTRADAMENTO (ROUTING) ≠ TABELLE DI INOLTRO (FORWARDING)

Forwarding table VS Routing table

- Forwarding table
  - Used when a packet is being forwarded and so must contain enough information to accomplish the forwarding function
  - A row in the forwarding table contains the mapping from a network number to an outgoing interface and some MAC information, such as Ethernet Address of the next hop
- Routing table
  - Built by the routing algorithm as a precursor to build the forwarding table
  - Generally contains mapping from network numbers to next hops

Forwarding table VS Routing table

- Forwarding table is obtained from routing table
- Routing table is built by routing algorithms using informations received from other nodes



Forwarding table VS Routing table

(a) routing table: used by control plane, contains level 3 data, with infos about costs, distances, etc.

(b) forwarding table: used by data plane, so it contains data for forwarding packets: which interface, address of next hop, etc.

(a)

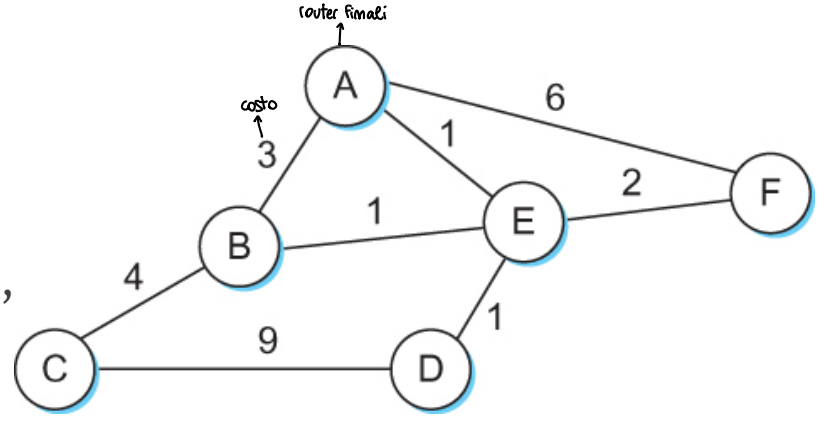
Prefix/CIDR (NETWORK NUMBER)	Cost	Next Hop
18/8	3	171.69.245.10
151.85/16	4	195.84.38.5

(b)

Prefix/CIDR (NETWORK NUMBER)	Interface	Next Hop
18/8	if0	84:3f:96:43:ad:6c
151.85/16	if2	6a:5c:04:22:5d:90

Routing

- Network as a Graph
  - nodes = routers connected to (basic) networks
  - links = connections
  - costs = speed, latency, costs, loss rate… many options



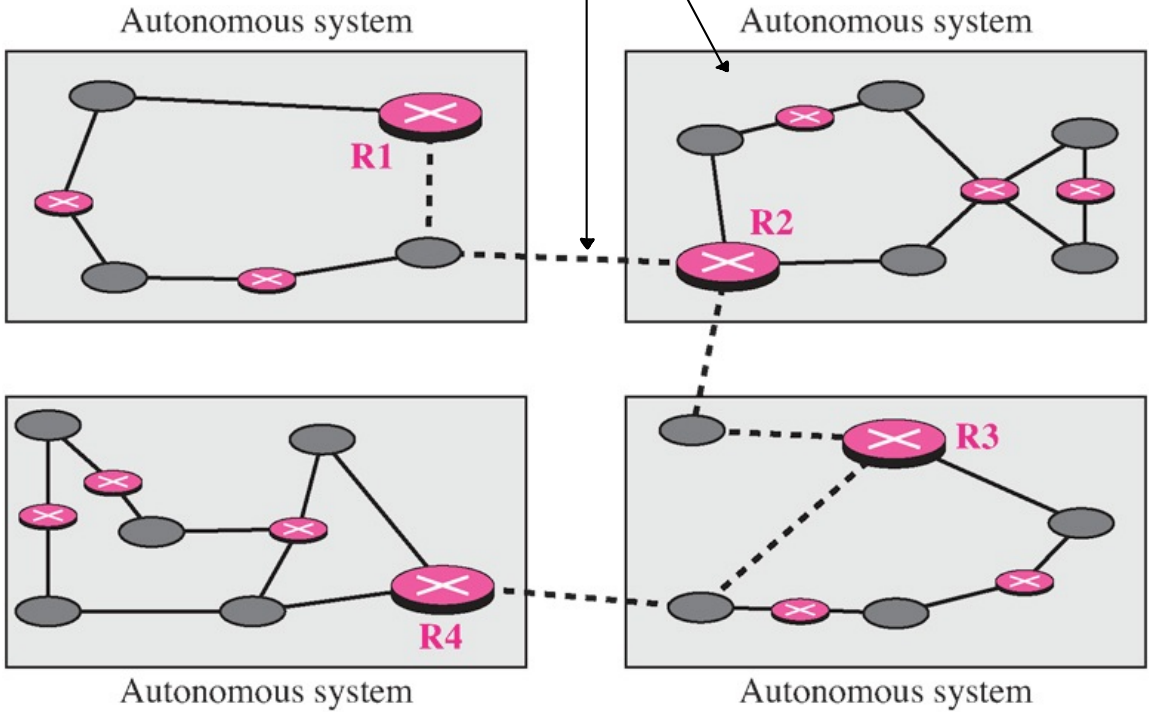
- The basic problem of routing is to **find the lowest-cost path between any two nodes** → tabelle di FORWARDING dovrebbero essere costruiti coi cammini MINIMI
  - Where the cost of a path equals the sum of the costs of all the edges that make up the path

Routing algorithms

- Difficult to have a unique algorithm for the whole Internet
  - Scalability issues
  - Different administrative domains may choose different routing policies
- Better separate in two levels
  - inside *Autonomous Systems* (es. provider) → blocchi con una stessa politica
    - each admin is responsible of routing algorithms
    - not too large networks => simpler algorithms (called *intra-domain*, or *interior* protocols)
  - among Autonomous Systems (tra diversi AS (Autonomous System))
    - **reachability** issues rather than efficiency
    - more complex algorithms (called *inter-domain*, or *exterior* protocols)

Routing algorithms

dentro un Autonomous system
tra Autonomous system diversi
cambiamo le regole



Routing (all'interno di un AUTONOMOUS SYSTEM) APPROCIO STATICO

- For a simple network, we can calculate all shortest paths and load them into some nonvolatile storage on each node.
- Such a static approach has several shortcomings
  - It does not deal with node or link failures
  - It does not consider the addition of new nodes or links
  - It implies that edge costs cannot change

Routing

- Solution: a **distributed** and **dynamic** protocol for constructing routing tables at each
- Three main classes of protocols
  - **Distance Vector** → scambio informaz. di TUTTI solo con i VICINI
    - Used in interior protocols like RIP (obsoleto)
  - **Link State** → scambio informaz. dei VICINI con TUTTI
    - Used in interior protocols like OSPF (più usato)

  → dentro l'AUTONOMOUS SYSTEM "cerco il percorso migliore"

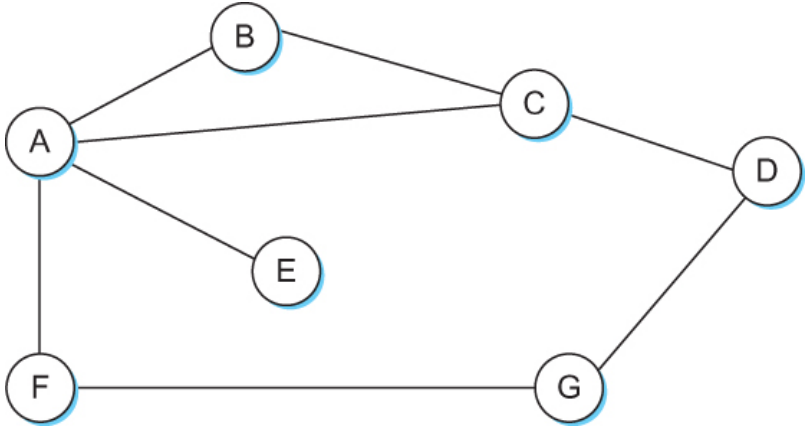
  - **Path vector**
    - Used in exterior protocols like BGP – next chapter → tra gli AUTONOMOUS SYSTEM

"Trovo un percorso e basta che funzioni, non serve che sia ottimale"

Distance Vector → costo di pochi byte

- Starting assumption is that each node knows the cost of the link to each of its directly connected neighbors
- Each node constructs a one dimensional array (a vector) containing the "distances" (costs) to all other nodes and distributes that vector to its immediate neighbors

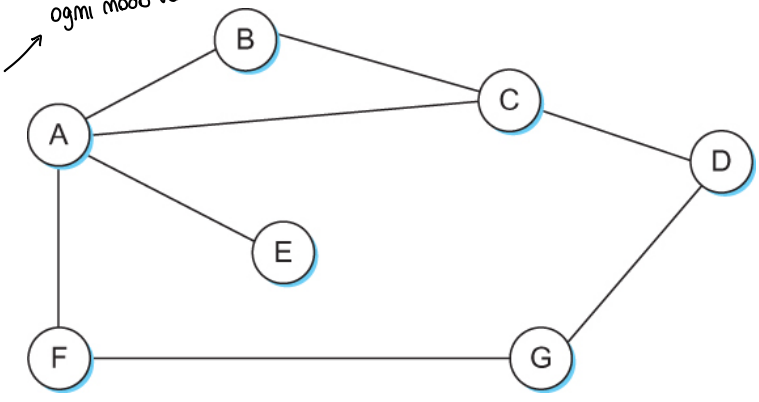
↳ solo dei vicini, non conosce tutto il grafo → ALGORITMO DISTRIBUITO (es. SPANNING TREE), ciascun nodo non conosce tutta la rete ma la collabora. di tutti fornisce la mappa di rete



Distance Vector

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

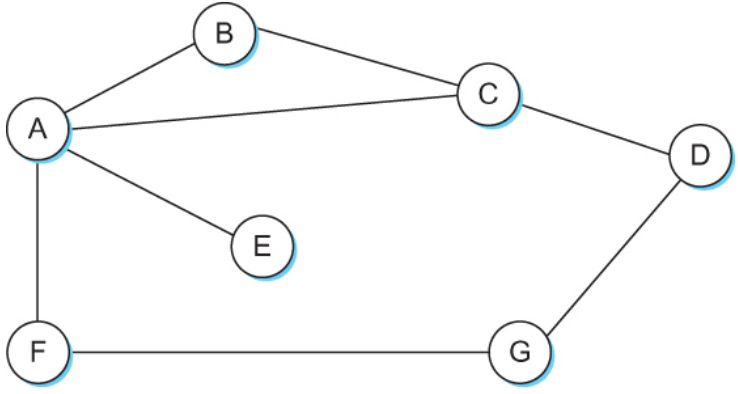
ogni nodo vede solo la sua riga, questa tabella con tutti i nodi non esiste



- Initial distances stored at each node (global view)

Distance Vector

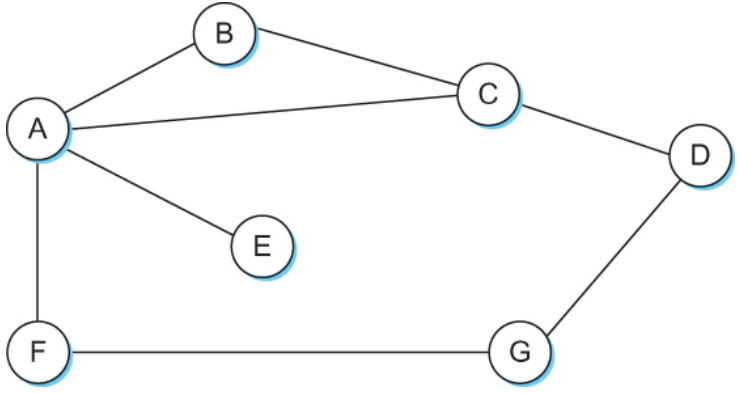
Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—



Initial routing table at node A

Distance Vector

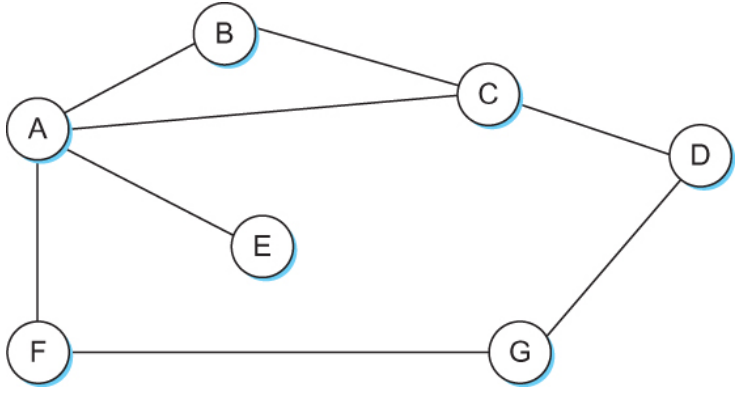
Destination	Cost	NextHop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F



- Final routing table at node A

Distance Vector

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



- Final distances stored at each node (global view)

Distance Vector

- The distance vector routing algorithm is sometimes called as **Bellman-Ford** algorithm:
  - Every T seconds each router sends its table to its neighbors
  - when a router receives a vector from a neighbor router, updates its table based on the new information: for each entry
    1. if we have a better route than the one already known, the entry is replaced
    2. if the best route is still through the neighbor router, the cost is updated

Distance Vector

es. per B, la sua tabella

B	DEST	dist	next h.
	A	1	A
	B	0	—
	C	1	C
	D	1	D
	E	1	E

invia il VETTORE ai vicini:

A 1
B 0
C 1
D 1
E 1

- V = vector received by neighbor R
- T = my local routing table
- cost(R) = cost of the link towards R
- Update Algorithm when we receive a vector V from R:

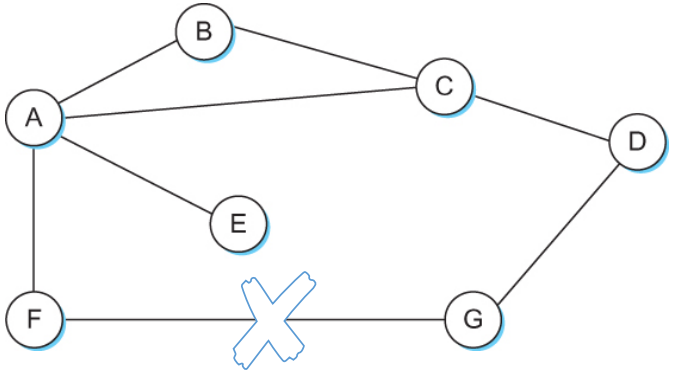
```
for each entry (D,c,N)∈T:
   c' = V(D)+cost(R)
   if R = N then
      replace (D,c,N) in T with (D,c',N)
   else if c' < c then
      replace (D,c,N) in T with (D,c',R)
```

caso 2, non sono stati trovati percorsi migliori, si aggiorna solo il costo (e non l'ultimo hop)

caso 1, trovato un percorso più breve, aggiorno l'ultimo hop

Distance Vector

- When a node detects a link failure
  - F detects that link to G has failed
  - F sets distance to G to infinity and sends update to A
  - A sets distance to G to infinity since it uses F to reach G
  - A receives periodic update from C with 2-hop path to G
  - A sets distance to G to 3 and sends update to F
  - F decides it can reach G in 4 hops via A



Distance Vector

tempo di convergenza dell'algo: MOLTO LENTO

- Slightly different circumstances can prevent the network from stabilizing
  - Suppose the link from A to E goes down
  - In the next round of updates, A advertises a distance of infinity to E, but B and C advertise a distance of 2 to E
  - Depending on the exact timing of events, the following might happen
    - Node B, upon hearing that E can be reached in 2 hops from C, concludes that it can reach E in 3 hops and advertises this to A
    - Node A concludes that it can reach E in 4 hops and advertises this to C
    - Node C concludes that it can reach E in 5 hops → perchè dato che per passare per E, C passa per A, allora se in A si aggiorna il costo per E, quando invia la sua tabella a C anche C aggiorna il suo costo per E (dato che passa per A)
    - and so on… **count to infinity!**

↳ perchè ogni volta che riceve dal vicino un valore maggiore, lui lo aggiorna perchè l'algo funziona così (finchè dai vicini non arriva un valore minore)

Count-to-infinity Problem

soluzione 1:

- Use some relatively small number as an approximation of infinity
- For example, the maximum number of hops to get across a certain network is never going to be more than 16
  - Used in actual implementations (RIP)
- When the number reaches the limit, stop incrementing and treat it as ∞
- Does not work if the network outgrows the limit
  - hence good only for limited networks, like AS

Count-to-infinity Problem

Soluzione 2:

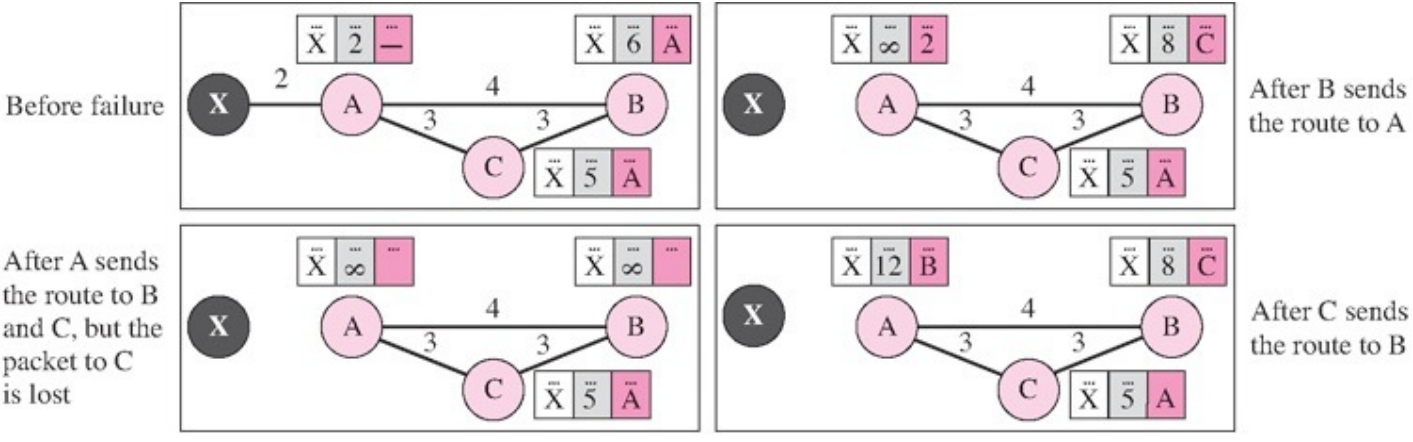
- One technique to improve the time to stabilize routing is called **split horizon** → non sempre funziona bene con 3 o più nodi
  - When a node sends a routing update to its neighbors, it does not send those routes it learned from each neighbor back to that neighbor → non invio la parte di DIST.VEC. a B che ha come next hop B
  - For example, if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, and so whenever B sends a routing update to A, it does not include the route (E, 2) in that update

Count-to-infinity Problem

- Split horizon does not allow a node to know whether the neighbors receive its messages
- In a stronger version of split horizon, called *split horizon with poison reverse*
  - B actually sends that back route to A, but it puts negative information in the route to ensure that A will not eventually use B to get to E
  - For example, B sends the route (E, ∞) to A

Count-to-infinity Problem

- Split horizon techniques does not solve the problem with loops of 3 or more nodes



- For this reason, usually "finite infinite" is used in actual implementations

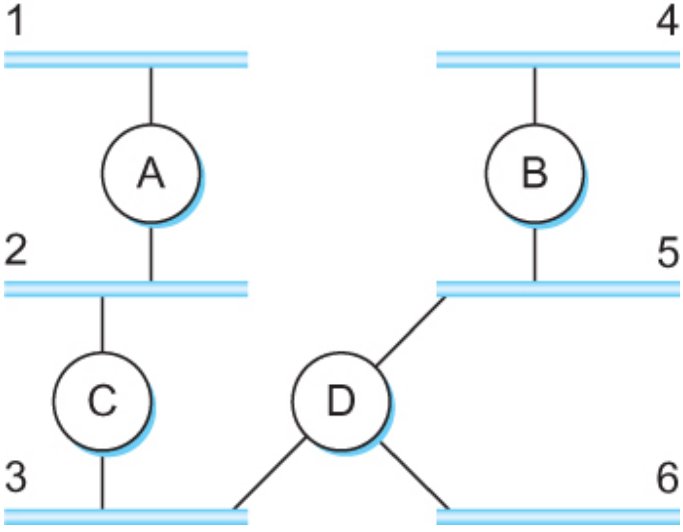
↳ il RIP è una sua implementazione
PONGO UN LIMITE AL NUMERO DI HOP

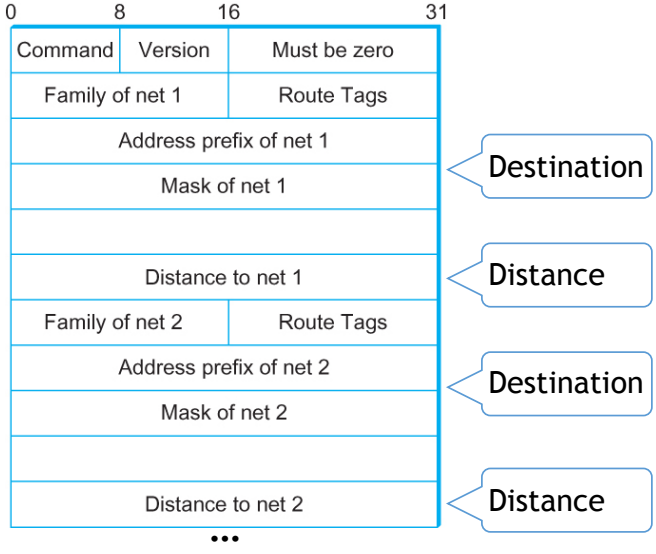
Routing Information Protocol (RIP)

- RIP = Routing Information Protocol (v2)
  - An application level protocol, run over UDP
- Common solution for intradomain routing
  - Implemented on *nix systems by routed(8)
- Adopts distance vector technique with finite infinite
  - final destinations are network address
  - nodes are routers directly connected to networks
  - Next hop is the address of a router, or null if the destination is in the same network
  - Each hop counts 1
  - Limit value for infinity is 16 (hence it works on networks with diameter ≤15)

le tabelle sono salvate in mem. utente

Routing Information Protocol (RIP)





- Example Network running RIP

RIPv2 Packet Format

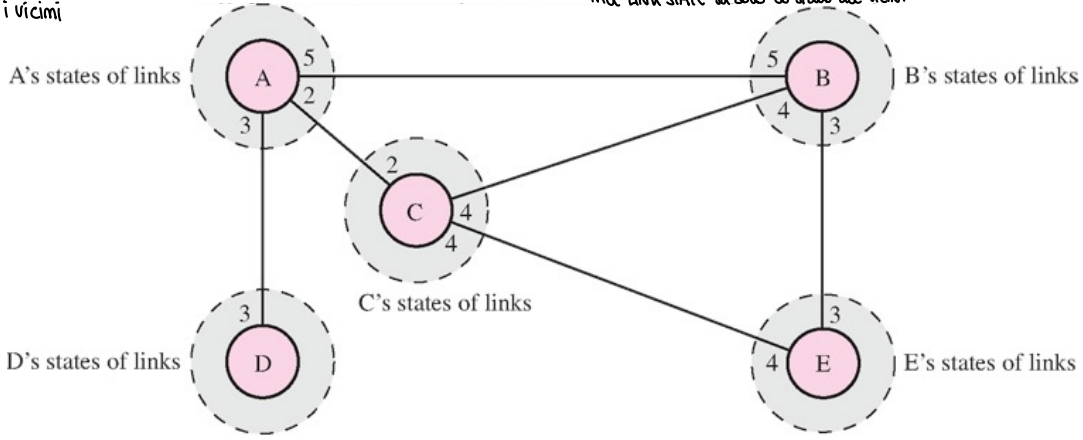
Link State Routing → pesano molto di più di RIP

- Completely different approach: each node constructs a **complete view** of the network (nodes and links)
  - Different nodes can have different views
- Each node checks the status of its direct connections, and informs the others of any change

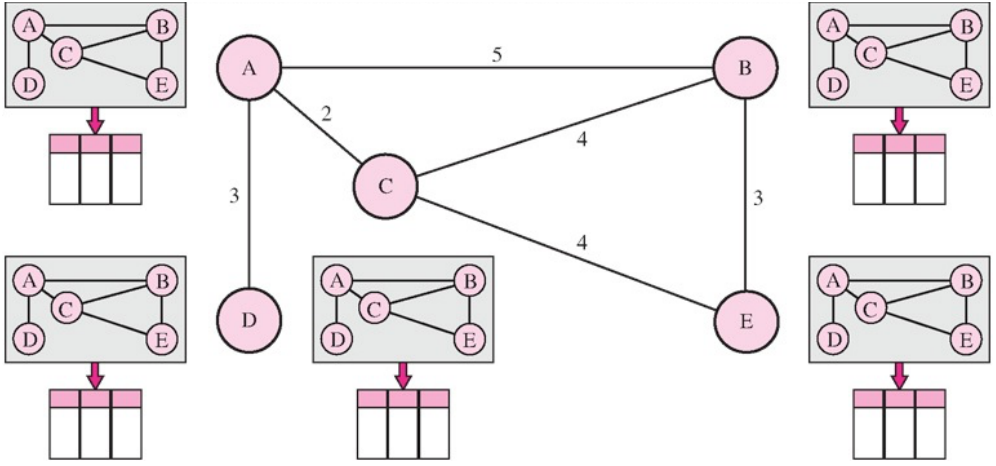
→ possono essere FUNZIONANTI / NON FUNZIONANTI → lo verifica con HELLO PACKETS

↳ mentre nel RIP sapeva lo stato di tutti, nel LINK STATE sa solo lo stato dei VICINI

↳ tutti, non solo i vicini



- When a node has a complete view of the network it can compute the minimal paths for each node
  - E.g. using Dijkstra algorithm



Link State Routing

- Strategy: Inform periodically all nodes (not just neighbors) information about directly connected links (not entire routing table).
  → inviato in FLOODING
- **Link State Packet** (LSP): contains
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO) → è un TIMESTAMP, si incrementa ogni volta che si genera un pacchetto LSP
  - time-to-live (TTL) for this packet
- Packets are propagated by *Reliable Flooding* → quando preparo pacchetto LSP, lo inoltro ai miei vicini che a loro volta lo inoltrano ai loro vicini e così via...
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - generate new LSP periodically; increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP; discard when TTL=0

Link State

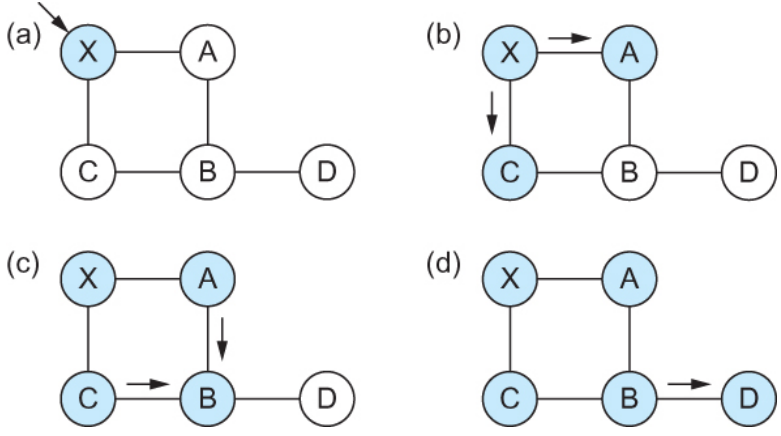
- **Reliable Flooding**
  - Flooding of link-state packets

(a) LSP arrives at node X;

(b) X floods LSP to A and C;

(c) A and C flood LSP to B (but not X);

(d) flooding is complete



Si utilizzano gli ACK per notificare l'arrivo dei pacchetti di flooding

=> una volta che tutti hanno ricevuto i pacchetti LSP, ciascun host è in grado di costruirsi la mappa di rete con Dijkstra

Shortest Path Routing

- Dijkstra's Algorithm - Assume non-negative link weights
  - N: set of nodes in the graph
  - l(i, j): the non-negative cost associated with the edge between nodes i, j $\in$N and l(i, j) = $\propto$ if no edge connects i and j
  - Let s$\in$N be the starting node which executes the algorithm to find shortest paths to all other nodes in N
  - Two variables used by the algorithm
    - M: set of nodes incorporated so far by the algorithm
    - C(n) : the cost of the path from s to each node n

Shortest Path Routing

- Dijkstra's Algorithm - The algorithm

```
M = {s}
for each n in N – {s}
  C(n) = l(s, n)
while (N ≠ M)
  M = M U {w} such that C(w) is the minimum for
  all w in (N-M)
  For each n in (N-M)
      C(n) = MIN (C(n), C(w) + l(w,n))
```

- Complexity: O(|l| + |N|log(|N|))

Shortest Path Routing

- In practice, each switch computes its routing table directly from the LSP's it has collected using a realization of Dijkstra's algorithm called the **forward search algorithm**
- Specifically each switch maintains two lists, known as Tentative and Confirmed
- Each of these lists contains a set of entries of the form (Destination, Cost, NextHop)

Shortest Path Routing - Algorithm

L'algoritmo implementato negli switch funziona nel modo seguente:

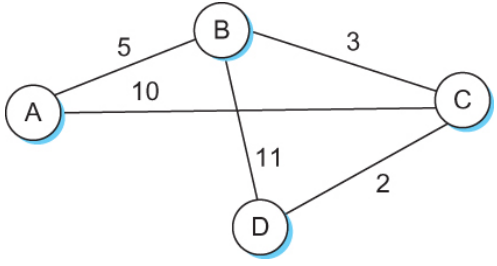
nodo S (START)

1. Inizializzo la lista Confirmed con il dato relativo a me stesso, con costo 0.
2. Seleziono il pacchetto LSP relativo al nodo (chiamato Next) appena inserito nella lista Confirmed nel passo precedente.
3. Per ciascun vicino (Neighbor) di Next, calcolo il costo (Cost) necessario per raggiungere Neighbor come somma del costo da me stesso a Next e del costo da Next a Neighbor.
    a. Se Neighbor non è nella lista Confirmed né nella lista Tentative, aggiungo alla lista Tentative, dove NextHop è la direzione verso cui devo andare per raggiungere Next.
    b. Se Neighbor è nella lista Tentative e Cost è minore del costo attualmente associato a Neighbor, sostituisco il dato attualmente memorizzato per Neighbor con, dove NextHop è la direzione verso cui devo andare per raggiungere il Next.
4. Se la lista Tentative è vuota, l'algoritmo termina; altrimenti scelgo dalla lista Tentative il dato avente il costo inferiore, spostandolo nella lista Confirmed e tornando al passo 2.

Shortest Path Routing

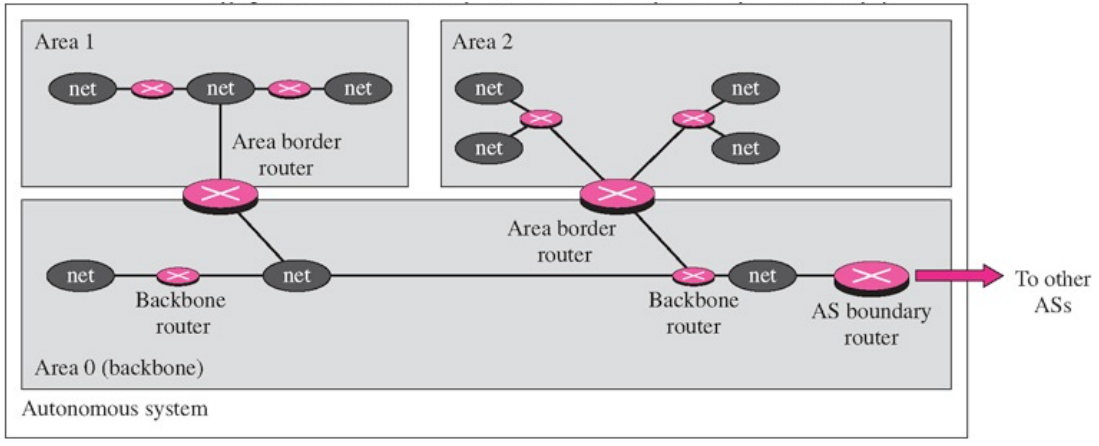
ESEMPIO:

Step	Confirmed	Tentative	Comments
1	(D,0,–)		Since D is the only new member of the confirmed list, look at its LSP.
2	(D,0,–)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better than anything else on either list, so put it on Tentative list; same for C.
3	(D,0,–) (C,2,C)	(B,11,B)	Put lowest-cost member of Tentative (C) onto Confirmed list. Next, examine LSP of newly confirmed member (C).
4	(D,0,–) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
5	(D,0,–) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to Confirmed, then look at its LSP.
6	(D,0,–) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative entry.
7	(D,0,–) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative (A) to Confirmed, and we are all done.



Open Shortest Path First (OSPF)

- Protocoll Open Shortest Path First
  - implemented on *nix by gated(8)
  - directly over IP (no UDP), with authentication
- adopts the Link State approach
- The AS is divided in *areas* and *backbone*
- each area is connected to backbone or other areas by a **border router**

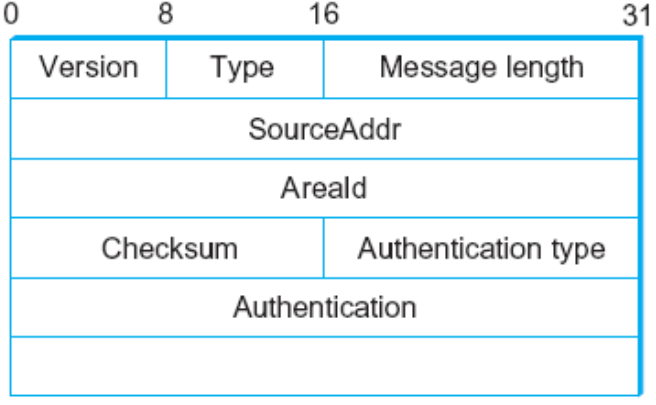


Open Shortest Path First (OSPF)

- Administrator can define link costs in various ways
  - speed, cost, load charge, latency, reliability, …
- OSPF allows LSP to carry different kind of costs, even for the same links
- A router can keep several routing tables (and correspondingly several forwarding tables), one for each intended kind of service: it can make different forwarding choices for different services
- which table to apply is decided upon
  - TOS/DSCP (rare)
  - QoS, priority (if supported)
  - flow tag and priority in IPv6
  - Automatic flow kind detection (traffic shaping)

Open Shortest Path First (OSPF)

- OSPF Header Format



OSPF Link State Advertisement

LS Age		Options	Type = 1
Link-state ID			
Advertising router			
LS sequence number			
LS checksum		Length	
0	Flags	0	Number of links
Link ID			
Link data			
Link type	Num_TOS	Metric	
Optional TOS information			
More links			

Summary

- We have looked at some of the issues involved in building scalable and heterogeneous networks by using switches and routers to interconnect links and networks.
- To deal with heterogeneous networks, we have discussed in details the service model of Internetworking Protocol (IP) which forms the basis of today's routers.
- We have seen several ancillary protocols (ICMP, NAT, DHCP)
- We have seen IPv6, next generation internet protocol
- We have discussed in details two major classes of routing algorithms
  - Distance Vector
  - Link State