



Politecnico  
di Torino

# Tecnologie e Servizi di Rete

Computer Engineering

Marco Lampis

23 febbraio 2023

# Indice

<b>0 Informazioni</b>	<b>1</b>
0.1 Contributi . . . . .	2
<b>1 IPv4 Summary</b>	<b>3</b>
1.1 Terminologia per l'indirizzamento . . . . .	3
1.2 Indirizzi speciali . . . . .	3
1.3 Indirizzamento con classi . . . . .	3
1.4 Indirizzamento IP senza classi (CIDR) . . . . .	4
1.5 IP routing . . . . .	5
1.6 IP addressing methodology . . . . .	7
1.6.1 Esercizi . . . . .	9
1.7 Multicast . . . . .	19
1.7.1 Status . . . . .	21
<b>2 IPv6</b>	<b>23</b>
2.1 Perché IPv4 non basta e soluzioni . . . . .	23
2.2 Chi assegna gli indirizzi IP . . . . .	24
2.3 Address pool status e scalabilità . . . . .	24
2.4 Notazione . . . . .	25
2.5 Routing . . . . .	26
2.6 Multicast . . . . .	27
2.7 Unicast . . . . .	28
2.7.1 Link local/site local Addresses . . . . .	29
2.7.2 Unique Local Addresses . . . . .	30
2.7.3 IPv4 Embedded Addresses . . . . .	30
2.7.4 Loopback Addresses . . . . .	31
2.7.5 Unspecified Addresses . . . . .	31
2.8 Anycast Addresses . . . . .	31
2.9 Protocolli utilizzati . . . . .	32
2.10 Packet Header Format . . . . .	32
2.10.1 Hop-by-Hop Extension Header . . . . .	35

2.10.2 Routing Extension Header . . . . .	35
2.10.3 Altre estensioni . . . . .	36
2.11 Interfacciarsi con i livelli più bassi . . . . .	37
2.11.1 Incapsulamento . . . . .	37
2.11.2 Address mapping . . . . .	37
2.11.3 IPv6 Multicast transmission . . . . .	37
2.12 Neighbor Discovery and Address Resolution . . . . .	38
2.12.1 Solicited-Node Multicast Address . . . . .	38
2.12.2 Risoluzione di un indirizzo . . . . .	39
2.13 La transizione tra IPv4 e IPv6 . . . . .	40
2.14 ICMPv6 . . . . .	40
2.14.1 Formato del messaggio . . . . .	42
2.14.2 Multicast Group Management . . . . .	44
2.14.3 Host Membership Discovery . . . . .	45
2.15 Device Configuration in IPv6 . . . . .	45
2.15.1 Privacy extension Algorithm . . . . .	47
2.15.2 Indirizzi . . . . .	48
2.15.3 ICMP Redirect . . . . .	49
2.15.4 Duplicate Address Detection (DAD) . . . . .	51
2.15.5 Fasi di una configurazione Stateless . . . . .	51
2.16 Scoped Addresses . . . . .	51
2.17 Routing Protocols . . . . .	52
2.18 La transizione da IPv4 a IPv6 . . . . .	53
2.18.1 Host centered solutions . . . . .	54
2.18.2 Network center solution . . . . .	56
2.19 Scalable, Carrier-grade Solutions . . . . .	58
2.19.1 AFTR: Address Family Transition Router . . . . .	59
2.19.2 DS-Lite . . . . .	60
2.19.3 A+P (Address plus Port) . . . . .	61
2.19.4 Mapping Address and Port (MAP) . . . . .	61
2.19.5 Port Set . . . . .	62
2.19.6 Mapping Rule . . . . .	63
2.19.7 Border Relay . . . . .	63
2.20 NAT64 + DNS64 . . . . .	63
<b>3 Reti wireless e cellulari</b>	<b>67</b>
3.1 Introduzione . . . . .	67

3.2	Wireless LAN . . . . .	69
3.2.1	CSMA/CA . . . . .	70
3.3	Reti cellulari . . . . .	73
3.3.1	Cluster . . . . .	74
3.3.2	Power Control . . . . .	76
3.3.3	Allocazione della frequenza . . . . .	77
3.3.4	Architettura di rete . . . . .	77
3.4	Evoluzione della rete cellulare . . . . .	80
3.4.1	GSM - Seconda generazione . . . . .	80
3.4.2	4G/LTE - quarta generazione . . . . .	85
3.5	Modalità di sleep . . . . .	91
3.5.1	5G - quinta generazione . . . . .	92
3.6	Mobilità nel 4G/5G . . . . .	94
<b>4</b>	<b>Principi del modern Lan Design</b>	<b>99</b>
4.1	Ripetitori . . . . .	99
4.2	Bridge . . . . .	101
4.3	LAN Moderne . . . . .	102
4.3.1	Transparent bridges . . . . .	103
4.3.2	Filtering database . . . . .	104
4.4	Routers . . . . .	106
4.5	VLAN . . . . .	106
<b>5</b>	<b>VPN</b>	<b>113</b>
5.1	Modalità di distribuzione . . . . .	117
5.1.1	Site to Site VPN Tunneling (s2s) . . . . .	117
5.1.2	End to End VPN Tunneling (e2e) . . . . .	117
5.1.3	Remote VPN Tunneling . . . . .	118
5.1.4	Overlay Model . . . . .	118
5.1.5	Peer Model . . . . .	118
5.1.6	Customer Provisioned VPN . . . . .	118
5.1.7	Provider Provisioned VPN . . . . .	119
5.1.8	Access VPN Customer Provisioned . . . . .	119
5.1.9	Tunneling . . . . .	121
5.2	Topologie . . . . .	121
5.3	Livelli . . . . .	122
5.3.1	Livello 2 . . . . .	122
5.3.2	Livello 3 . . . . .	122

5.3.3	Livello 4 . . . . .	123
5.4	Generic Routing Encapsulation (GRE) . . . . .	123
5.4.1	Enhanced GRE (version 1) . . . . .	124
5.5	Protocolli di livello 2 . . . . .	125
5.5.1	L2TP . . . . .	125
5.5.2	Point to Point Tunneling Protocol (PPTP) . . . . .	128
5.6	IPsec . . . . .	129
5.7	SSL VPN . . . . .	131
5.7.1	Protocolli con SSL . . . . .	133
5.7.2	Application Translation . . . . .	133
5.7.3	Application Proxying . . . . .	134
5.7.4	Port Forwarding . . . . .	134
5.8	VPN Gateway Positioning & anomalies . . . . .	134
5.9	Anomalie e posizionamento . . . . .	134
5.9.1	Monitorability anomaly . . . . .	135
5.9.2	Skewed Channel anomaly . . . . .	137
<b>6</b>	<b>Routing</b>	<b>139</b>
6.1	Introduzione . . . . .	139
6.1.1	Proactive routing . . . . .	139
6.1.2	On the fly routing . . . . .	139
6.2	Algoritmi per il proactive routing . . . . .	140
6.2.1	Non adaptive algorithms . . . . .	140
6.2.2	Adaptive algorithms . . . . .	140
6.3	Distance vector (Bellman-Ford) . . . . .	141
6.4	Path Vector . . . . .	146
6.5	Link State Routing Algorithm . . . . .	146
6.5.1	Algoritmo di Dijkstra . . . . .	147
6.6	Internet Routing Architecture . . . . .	148
6.6.1	Autonomous System . . . . .	148
6.7	Protocolli di routing . . . . .	150
6.7.1	Algoritmi IGP . . . . .	151
6.7.2	Algoritmi EGP . . . . .	153
<b>7</b>	<b>MPLS</b>	<b>155</b>
7.1	Architettura di rete . . . . .	156
7.2	Storia di MPLS . . . . .	158
7.3	Header MPLS . . . . .	159

7.4	LSP setup - selezione del path e delle etichette . . . . .	160
7.4.1	Label Binding . . . . .	160
7.4.2	Label Mapping . . . . .	160
7.4.3	Label Distribution . . . . .	160
7.4.4	Label Binding statico (e mapping) . . . . .	161
7.4.5	Label Binding dinamico . . . . .	161
7.4.6	Protocolli per la Label Distribution . . . . .	161
7.5	Protocolli di routing . . . . .	162
7.5.1	Modalità di Routing . . . . .	162
7.6	Traffic Engineering . . . . .	164
7.7	CoS e QoS . . . . .	165
7.7.1	Class of Service (CoS) . . . . .	165
7.7.2	Quality of Service (QoS) . . . . .	165
7.8	Fast fault recovery . . . . .	166
7.9	Gerarchia e scalabilità . . . . .	166
7.10	Penultimate Hop Popping (PHP) . . . . .	167
7.11	MPLS VPN . . . . .	167
7.11.1	PWE3 . . . . .	168
7.11.2	MPLS-based Layer 3 VPNs . . . . .	168
7.11.3	Componenti . . . . .	169
7.11.4	Benefici . . . . .	169
7.11.5	MPLS VPN basate su BGP . . . . .	170
7.11.6	MPLS Virtual Router VPNS . . . . .	170
7.12	6PE . . . . .	171
<b>8</b>	<b>Rete Ottica</b> . . . . .	<b>173</b>
8.1	Switching Core . . . . .	174
8.1.1	Optical Core . . . . .	174
8.1.2	Electronic Core . . . . .	175
8.2	Switching Dynamics . . . . .	175
8.2.1	Cross Connect . . . . .	175
8.2.2	Fiber Cross Connect . . . . .	175
8.2.3	Wavelength Cross Connect . . . . .	176
8.3	Wavelength Conversion . . . . .	176
8.4	Combinazioni comuni . . . . .	176
8.4.1	Wavelength cross-connect con Wavelength Conversion . . . . .	176
8.4.2	Dynamic Optical Switching . . . . .	176
8.5	Distribuzione . . . . .	177

8.6 Control Plane . . . . .	177
8.7 Routing . . . . .	178
8.8 Data Transport and Protocol Stack . . . . .	179
<b>9 Quality of Service</b>	<b>181</b>
9.1 Requisiti . . . . .	181
9.2 Contromisure . . . . .	182
9.2.1 Classificazione . . . . .	182
9.2.2 Scheduling . . . . .	183
9.3 Controllo del traffico . . . . .	183
9.3.1 Call Admission Control . . . . .	185
9.4 Routing . . . . .	185
9.5 Frameworks . . . . .	185
9.5.1 IntServ . . . . .	185
9.5.2 DiffServ . . . . .	185

# 0 Informazioni

La seguente dispensa è stata realizzata nell'anno accademico 2022-2023 durante il corso di *Tecnologie e Servizi di Rete*. Il materiale **non** è ufficiale e non è revisionato da alcun docente, motivo per cui non mi assumo responsabilità per eventuali errori o imprecisioni.

Per qualsiasi suggerimento o correzione non esitate a contattarmi o a eseguire una pull request su GitHub.

E' possibile riutilizzare il materiale con le seguenti limitazioni:

- Utilizzo non commerciale
- Citazione dell'autore
- Riferimento all'opera originale

E' per tanto possibile:

- Modificare parzialmente o interamente il contenuto

Questi appunti sono disponibili su GitHub al seguente link:

1 [https://github.com/Guray00/polito\\_lectures](https://github.com/Guray00/polito_lectures)



**Figura 1:** Repository GitHub

La seguente dispensa è rilasciata sotto la Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License.



## 0.1 Contributi

La condivisione è alla base del successo di qualsiasi progetto, citando:

*“Open source is about collaborating; not competing. ~ Kelsey Hightower”*

La seguente dispensa ha avuto il prezioso contributo di:

- Marco Lampis

# 1 IPv4 Summary

In questo capitolo viene fatto un ripasso generico su quanto visto nei corsi precedenti relativo al **IPv4**, con particolare riferimento a *Reti Informatiche* (o equivalenti).

## 1.1 Terminologia per l'indirizzamento

Gli indirizzi /P in **IPv4** hanno una lunghezza pari a **32 bit** e vengono utilizzati per identificare le interfacce per router e host. Questi sono composti da due parti:

- **network part**: sono i bit più alti, identificano la rete in cui un host si trova.
- **host part**: sono i bit più bassi, identificano l'host all'interno della rete.

Un insieme di dispositivi con interfacce IP identificano una *IP Network*, caratterizzata dalla medesima network part, a cui tutti i dispositivi sono connessi al medesimo *physical network* (link layer).

## 1.2 Indirizzi speciali

In IPv4, oltre ai “classici” indirizzi, sono presenti alcuni indirizzi speciali:

- tutti i bit a 1: indirizzo di **broadcast**, non può essere assegnato
- 127.x.x.x: indirizzo di **loopback**, è una classe di indirizzi e servono a identificare l'host stesso e per tale motivo vengono solitamente utilizzati per debug.

**Ricorda:** Spesso al giorno d'oggi non è consentito l'invio di messaggi in broadcast per motivi di sicurezza.

## 1.3 Indirizzamento con classi

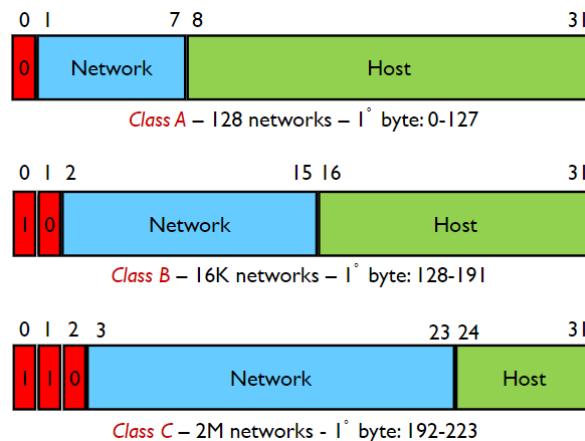
Le rappresentazioni possono essere **classes** (a classe) o **classless** (senza l'utilizzo di classi), in modo da sapere quali bit individuano la rete e quali gli host.

**Figura 1.1:** Indirizzi Speciali

La suddivisione in classi sulle seguenti tipologie:

- A:** il MSB identifica la classe, i 7 bit seguenti l'indirizzo di rete, i rimanenti sono per i dispositivi. Il totale degli indirizzi è  $2^7$  (128) per la rete e  $2^{24}$  per i dispositivi.
- B:** i 2 MSB identificano la classe, i 14 bit seguenti la rete e 16 bit per i dispositivi.
- C:** 3 bit per la classe, 21 bit per la rete e 8 bit per gli host.
- D:** 4 bit per la classe, 28 bit per la rete e 4 bit per gli host. Questi indirizzi sono riservati per i *multicast*.

Basta guardare i primi bit per capire la classe di un indirizzo.

**Figura 1.2:** Classi

## 1.4 Indirizzamento IP senza classi (CIDR)

Il sistema **Classless InterDomain Routing** permette di indirizzare in modo più preciso tra rete e dispositivi, rendendo la porzione di rete di lunghezza **arbitraria**. Il formato con cui può essere rappresentato un indirizzo è il seguente: **networkID + prefix length** oppure **netmask**.

Il **prefix length**, specificato con / $x$ , è il numero di bit della rete.

La netmask è identificata da una serie di bit posti a 1 che determinano quali bit identificano la rete, attraverso un **and** bit a bit.

*Esempio:*

```
1 200.23.16.0/23          # prefix length
2 200.23.16.0 255.255.255.254.0 # netmask
```

L'indirizzo viene espresso attraverso gruppi di **8 bit**, rappresentanti in modo decimale puntato (4 gruppi in quanto 32 bit totali). Ogni raggruppamento avrà un valore compreso tra 0 e 255.

In realtà **non tutti i valori sono permessi**, il più piccolo è **252**. Questo è dovuto al fatto che è sempre presente e non assegnabile l'indirizzo della sottorete (*Network ID*) e l'indirizzo del *inter-broadcast*.

Un modo per verificare se un indirizzo è scritto in modo corretto è prendere il prefix length / $x$  e controllare che l'ultimo numero puntato sia multiplo di  $2^{(32-x)}$ .

*Esempi:*

```
1 130.192.1.4/30 => 4%2^(32-30) = 4%4 = 0, si!
2 130.192.1.16/30 => 16%2^(32-30) = 16%4 = 0, si!
3 130.192.1.16/29 => 16%2^(32-29) = 16%8 = 0, si!
4
5 130.192.1.1/30 => 1%2^(32-30) = 1%4 != 0, no!
6 130.192.1.1/29 => 1%2^(32-29) = 1%8 != 0, no!
7 130.192.1.1/28 => 1%2^(32-28) = 1%16 != 0, no!
```

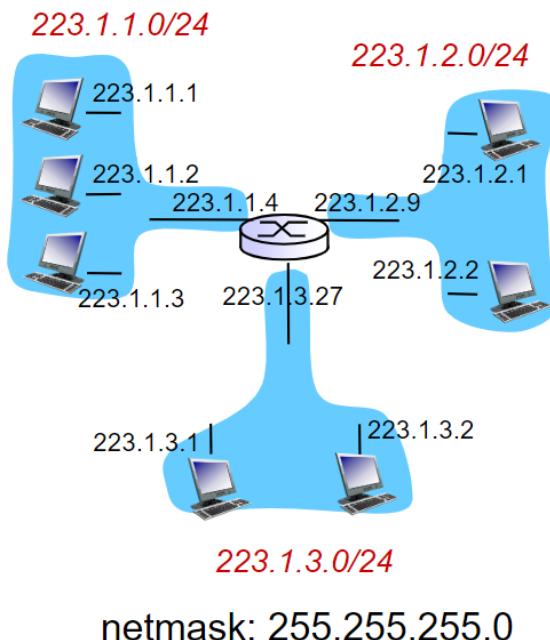
**Ricorda:** prefix length e netmask sono due modi equivalenti per rappresentare un indirizzo.

Per il ragionamento di sopra appare evidente che un indirizzo che termina con .1 **non sarà mai un indirizzo corretto**, in quanto ritornerà sempre un resto.

## 1.5 IP routing

Il routing degli host avviene attraverso la **routing table**, caratterizzata da due colonne che identificano:

- **destinazione:** indirizzi ip
- **interfaccia:** eth0, wlan etc...



**Figura 1.3:** Esempio di rete

Quando viene inviato un pacchetto, si cerca un match all'interno della tabella per identificare dove l'indirizzo IP di destinazione. Se è presente più di un match, viene considerato quello con il **prefisso più lungo** attraverso la tecnica del **longest prefix matching**.

**Nota:** i router sono identificati solitamente con un cerchio con dentro una x.

Di seguito è mostrato un esempio di routing:

Sono presenti in totale 7 sottoreti, di cui 3 reti locali e 4 reti punto punto. Tutta la sottorete ha come indirizzo quello raffigurato in alto a sinistra. Gli indirizzi di ciascuna di queste sono come segue:

Scriviamo la routing table del router identificando le reti direttamente connesse e raggiungibili. Prendiamo come riferimento **R1**:

Destination	Next	Type
130.192.3.0/30	130.192.3.1	direct
130.192.3.4/30	130.192.3.5	direct
130.192.2.0/24	130.192.2.1	direct
80.105.10.0/30	80.105.10.1	direct

Destination	Next	Type
0.0.0.0/0	80.105.10.2	static
130.192.0.0/24	130.192.3.2	static
130.192.1.0/24	130.192.3.2	static
130.192.3.8/30	130.192.3.2	static

## 1.6 IP addressing methodology

Preso come esempio la rete che segue, la metodologia da adoperare è la seguente:

1. Localizzare le reti IP, *in questo caso 3.*
2. Individuare il numero di indirizzi richiesti, *in questo caso nel router in alto a destra è sufficiente /30 perché ne sono richiesti 4 ( $2^2$ ), /26 a sinistra ( $2^6$ ) e /25 in basso a destra ( $2^7$ ).*
3. Calcolare quanti indirizzi è possibile allocare.
4. Verificare il range di validità degli indirizzi, *in questo caso /26, /25 e /30 dunque mi basterebbe o tutti e 3, o due /25 o infine un solo /24.*
5. Calcolare la netmask / prefix length.
6. Calcolare address range.
7. Calcolare gli indirizzi degli host.

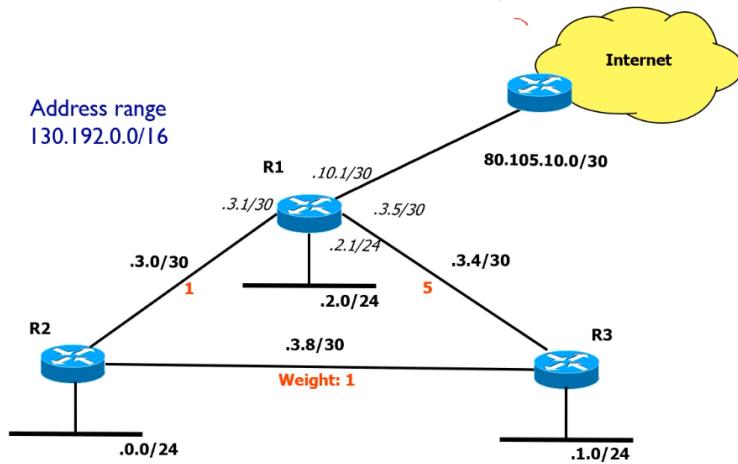
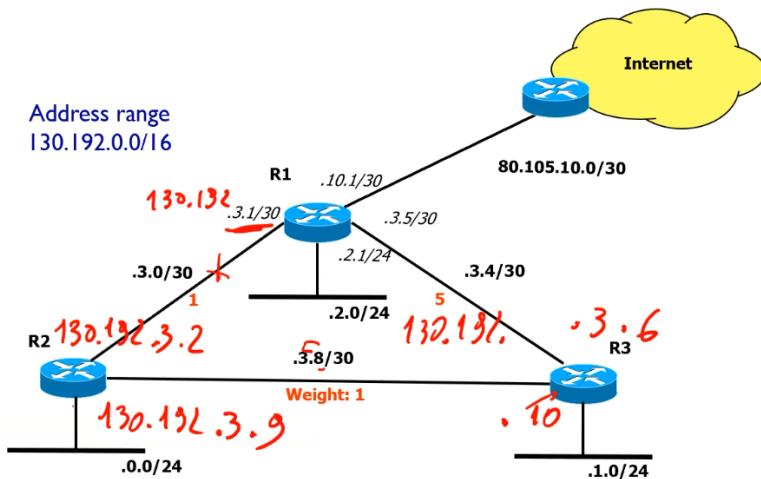
**Nota:** in basso a sinistra sono richiesti 43 indirizzi per 40 dispositivi. Ciò è dovuto al fatto che oltre ai 40 richiesti serve l'indirizzo di rete, l'indirizzo di broadcast e l'indirizzo del router.

Per riuscire a trovare le sottoreti, si prosegue in ordine dal più grande (*ovvero il valore minore*):

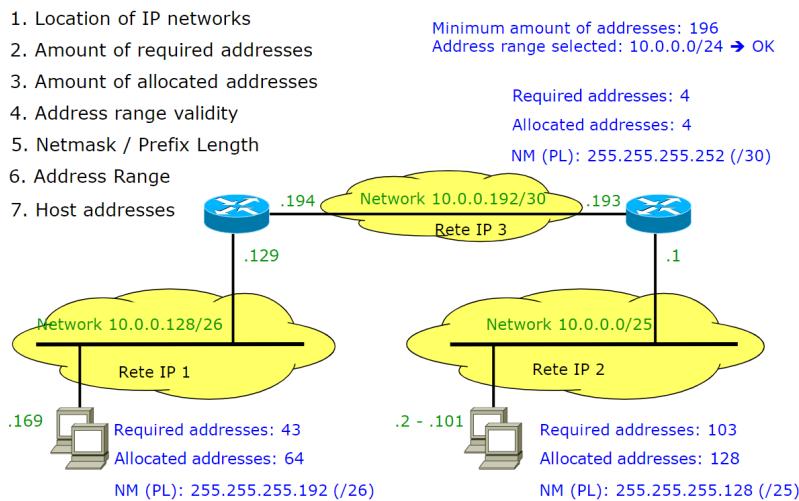
```

1 # tutta la rete (/24)
2 10.0.0.0/24
3
4 # subnet2 (/25), 32-25 = 7 => 2^7 = 128 indirizzi
5 # range: 0-127
6 10.0.0.0/25 <- primo
7 10.0.0.127 <- ultimo
8
9 # subnet3 (/26), 32-26 = 6 => 2^6 = 64 indirizzi
10 # range: 128-191
11 10.0.0.128/26 <- primo
12 10.0.0.191 <- ultimo
13
14 #subnet4 (/30), punto punto

```

**Figura 1.4:** routing**Figura 1.5:** routing2

## IP Addressing: methodology



**Figura 1.6:** Rete di esempio

15 10.0.0.192/30

Suggerimento: quando calcoli i bit per la maschera, vedi quanti zeri rimangono e fai  $256 - 2^{n\_zeri}$ .

**Ricorda:** Quando lasci lo spazio per gli indirizzi è sempre necessario riservarne 2 per l'indirizzo di rete e l'indirizzo di broadcast. Per questo motivo nelle connessioni punto punto (/30) devi comunque riservare 4.

### 1.6.1 Esercizi

#### 1.6.1.1 Esercizio 1

Assuming a classless addressing plan, define the netmask and the prefix length that have to be assigned to possible networks in order to contain the given number of hosts

Numero di hosts	NetMask	Prefix Length	Available Addresses
2	255.255.255.252	(32-2) -> /30	$2^2 - 2 = 2$
27	255.255.255.224	(32-5) -> /27	$2^5 - 2 = 30$

Numero di hosts	NetMask	Prefix Length	Available Addresses
5	255.255.255.248	(32-3) -> /29	$2^3 - 2 = 6$
100	255.255.255.128	(32-7) -> /25	$2^7 - 2 = 126$
10	255.255.255.240	(32-4) -> /28	$2^4 - 2 = 14$
300	255.255.254.000	(32-9) -> /23	$2^9 - 2 = 510$
1010	255.255.252.000	(32-10) -> /22	$2^{10} - 2 = 1022$
55	255.255.255.192	(32-6) -> /26	$2^6 - 2 = 62$
167	255.255.255.000	(32-8) -> /24	$2^8 - 2 = 254$
1540	255.255.248.000	(32-11) -> /21	$2^{11} - 2 = 2046$

**Nota:** per calcolare la netmask, si esegue  $256 - 2^{\text{bit}}$

### 1.6.1.2 Esercizio 2

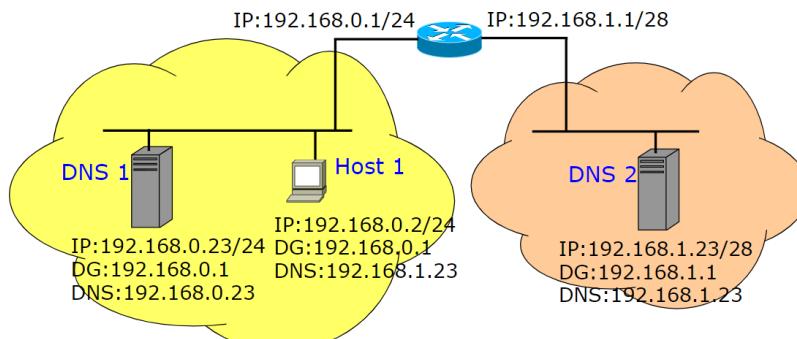
Verifica se i seguenti indirizzi sono validi o meno.

IP / Prefix Length pair	Valido?
192.168.5.0/24	Si, $0 \bmod 2^{(32-24)} = 0$
192.168.2.36/30	Si, $36 \bmod 2^{(32-30)} = 0$
192.168.2.36/29	No, $36 \bmod 2^{(32-29)} \neq 0$
192.168.2.32/28	Si, $32 \bmod 2^{(32-28)} = 0$
192.168.2.32/27	Si, $32 \bmod 2^{(32-27)} = 0$
192.168.3.0/23	No, $3 \bmod 2^{(1)} \neq 0$
192.168.2.0/31	No, /31 non ha senso
192.168.2.0/23	Si, $2 \bmod 2^{(1)} \neq 0$
192.168.16.0/21	Si, $16 \bmod 2^3 = 0$
192.168.12.0/21	No, $12 \bmod 2^3 \neq 0$

**Consiglio:** quando devi verificare la validità con prefix length che supera 8, significa che il controllo è da fare sul gruppo precedente (e così via), quindi puoi fare  $2^{(32-x)-8}$ . Stesso ragionamento quando si supera 16, 24, ecc...

### 1.6.1.3 Esercizio 3

Trova l'errore di configurazione nella rete indicata di seguito e spiega il motivo per cui questa non funziona come dovrebbe.



**Figura 1.7:** Configurazione errata

Il problema è relativo al fatto che il router non si trova nella medesima rete della rete arancione, in quanto essendo una /28 il range di indirizzi vanno da 192.168.1.0 a 192.168.1.15. In realtà quelli utilizzabili però sono da .1 a .14 in quanto i due rimanenti sono riservati per broadcast (.15) e rete (.0).

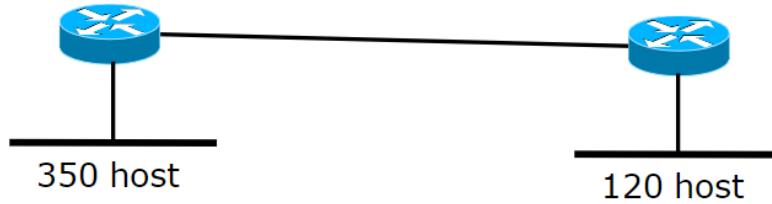
Le soluzioni sono due:

- utilizzare un /27 invece del /28 in modo da arrivare fino a .31, rendendo .23 corretto
- cambiare l'indirizzo del dns, ad esempio con 192.168.1.10

### 1.6.1.4 Esercizio 4

Definire un piano di indirizzamento IP per la rete in figura. Considerare entrambi i tipi di indirizzamento: "tradizionale" (senza minimizzare) e una soluzione che minimizzi il numero di indirizzi IP utilizzati. si assume di utilizzare il range 10.0.0.0/16.

Partiamo evidenziando come il router a sinistra, al fine di servire 350 host, ha in realtà bisogno di 353 indirizzi: 350 host + 1 indirizzo di rete + 1 indirizzo di broadcast + 1 indirizzo del router, dunque /23. Stesso ragionamento è applicabile al router di destra, che ha bisogno di 123 indirizzi dunque /25.

**Figura 1.8:** Rete

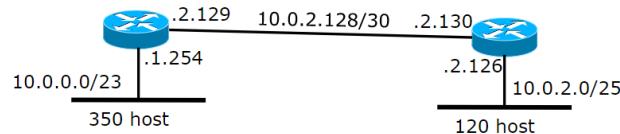
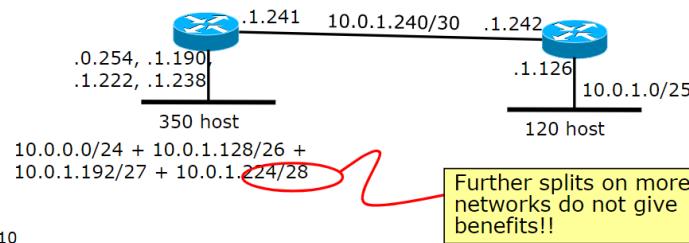
Troviamo così che  $10.0.0.0/23$  è la rete A (sinistra). Il suo indirizzo di broadcast sarà  $10.0.1.255$  in quanto adoperiamo 9 bit (*quindi gli ultimi 8 bit a 1 e il primo bit del terzo gruppo a 1*).

La sottorete C (destra) sarà identificata da  $10.0.2.0/25$  in quanto l'indirizzo immediatamente successivo. Il suo indirizzo di broadcast sarà  $10.0.2.127$ .

La sottorete B (centrale) sarà identificata da  $10.0.2.128/30$ , con /30 proveniente dal fatto che è una sottorete punto punto.

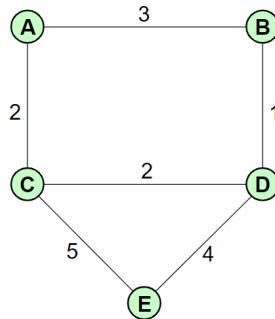
Questa soluzione comporta un grosso spreco, in quanto c'è un /25 che non viene utilizzato.

La seconda soluzione prevede l'utilizzo di più sottoreti per non sprecare indirizzi, in particolare un /24, /26, /27, /28 per un totale di  $256 + 64 + 32 + 16 = 368$  indirizzi.

**Solution1****Solution2****Figura 1.9:** Soluzioni

### 1.6.1.5 Esercizio 5

Definisci un albero di routing per tutti i nodi della rete mostrata di seguito.



**Figura 1.10:** Rete esercizio 5

L'**albero di instradamento** è quello che, a partire da un router della rete, stabilisce i percorsi minimi per raggiungere tutti i nodi. Per calcolarlo si prende un router come riferimento, ad esempio **A**, ei si calcolano tutte le distanze dagli altri nodi.

dest	next
B	3 (ramo dx)
C	2 (ramo inf)
D	4 (sia dx che inf)
E	7 (ramo inf)

La stessa procedura dovrà essere poi eseguita per tutti i nodi rimanenti, minimizzando le distanze. A parità di distanza solitamente ci sono motivi differenti per cui si scegli un percorso piuttosto che un altro (*ad esempio router più nuovi*).

Node A		Node B		Node C	
Destination	Next-hop	Destination	Next-hop	Destination	Next-hop
B	B	A	A	A	A
C	C	C	D	B	D
D	B/C	D	D	D	D
E	C	E	D	E	E

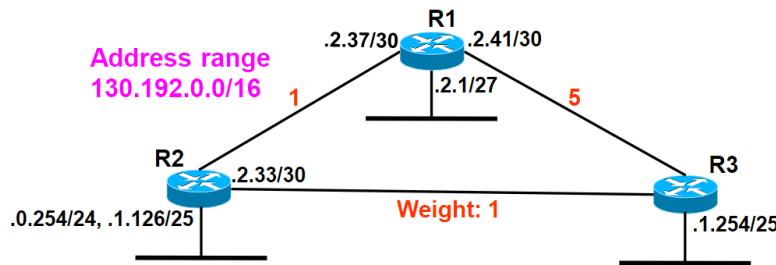
  

Node D		Node E	
Destination	Next-hop	Destination	Next-hop
A	B/C	A	C
B	B	B	D
C	C	C	C
E	E	D	D

**Figura 1.11:** Soluzione esercizio 5

### 1.6.1.6 Esercizio 6

Data la rete mostrata di seguito, definire la routing table di R1. La route aggregation deve essere massimizzata. Gli indirizzi ip mostrati in figura sono relativi all'interfaccia del router più vicino.



**Figura 1.12:** Esercizio 6

Cominciamo scrivendo la routing table di **R1**:

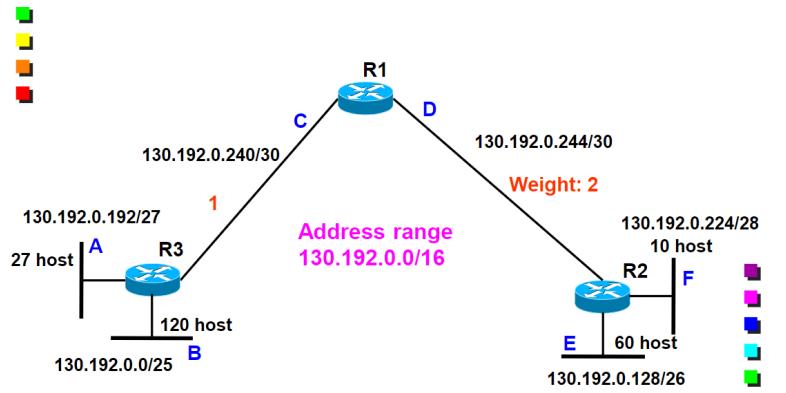
dest	next hop	Type
130.192.2.36/30 (A)	130.192.2.37	D
130.192.2.0/30 (B)	130.192.2.1	D
130.192.2.40/30 (C)	130.192.2.41	D
130.192.1.126/30 (D)	130.192.2.38	S
130.192.0.0/24 (E)	130.192.2.38	S
130.192.1.128/25 (F)	130.192.2.38	S
130.192.2.32/30 (G)	130.192.2.38	S

**D** ed **F** possono essere accorpati con 130.192.1.0/24, che a sua volta può essere aggregato con **E** ottenendo l'indirizzo 130.192.0.0/23 avendo il valore di broadcast pari a 130.192.1.255, per includere anche **G** è possibile usare 130.192.0.0/22. Dobbiamo però stare attenti a controllare come questi si rapportano con le entry statiche. In questo caso le include tutte, e non è un problema.

**Nota:** l'indirizzo 130.192.2.38 è l'indirizzo del router R2, 130.192.2.36 è l'indirizzo della sottorete (scelto prendendo il più alto multiplo di  $2^{(32-30)}$  minore di 36), mentre 130.192.2.37 è l'indirizzo dell'interfaccia di R1 per comunicare con R2.

### 1.6.1.7 Esercizio 7

Realizzare un piano di indirizzamento che minimizza il numero di indirizzi necessari.



**Figura 1.13:** Esercizio 7

Troviamo la routing table di **R1**, analizzando ogni nodo a partire dai collegamenti diretti:

- Nella sottorete **A** sono presenti 27 host, per cui sono necessari  $27+3$  indirizzi e un prefix length di  $(32 - 5) = 27$ .
- Nella sottorete **B** sono invece necessari  $120+3$  indirizzi, per cui un prefix length di  $(32 - 7) = 25$ .
- Le sottorete **C** e **D** sono invece una sottoreti punto punto, per cui è necessario un prefix length di 30.
- La sottorete **E** ha bisogno di  $60+3$  indirizzi, per cui un prefix length di  $(32 - 6) = 26$ . Infine la sottorete **F** ha bisogno di  $10+3$  indirizzi, per cui un prefix length di  $(32 - 4) = 28$ .

Troviamo adesso quali sono gli indirizzi delle sottoreti, partendo da quella di dimensione maggiore (**B**, in quanto /25).

- B:** 130.192.0.0/25, con indirizzo di broadcast 130.192.0.127 in quanto gli ultimi 7 bit sono a 1.
- E:** 130.192.0.128/26 con indirizzo di broadcast 130.192.0.191
- A:** 130.192.0.192/27, con indirizzo di broadcast 130.192.0.223
- F:** 130.192.0.224/28, con indirizzo di broadcast 130.192.0.239
- C:** 130.192.0.240/30, con indirizzo di broadcast 130.192.0.243
- C:** 130.192.0.244/30, con indirizzo di broadcast 130.192.0.247

E' ora possibile calcolare gli indirizzi dei next hop, prendendo come riferimento il router più vicino:

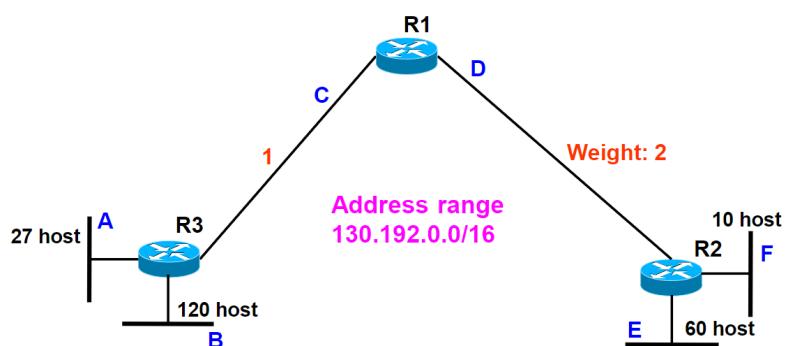
dest	Gateway	Type
130.192.0.240/30 (C)	130.192.0.241	D
130.192.0.244/30 (D)	130.192.0.245	D
130.192.0.192/27 (A)	130.192.0.242	S
130.192.0.0/25 (B)	130.192.0.242	S
130.192.0.128/26 (E)	130.192.0.246	S
130.192.0.224/28 (F)	130.192.0.246	S

Di queste entry bisogna valutare se è possibile fare qualche aggregazione. E' possibile farlo con **E** ed **F** in quanto: avendo /26 e 28, possono essere racchiusi in un /25 (quindi  $2^7$ ) con il medesimo indirizzo di **E** (130.192.0.128/25 è valido perché  $128 \% 128 = 0$ ). La soluzione risulta comunque inefficiente perché non abbiamo ottenuto solo una entry.

**Ricorda:** Il piano di indirizzamento si fa sempre partendo dalla sottorete più grande, ovvero l'intero minore.

#### 1.6.1.8 Esercizio 8

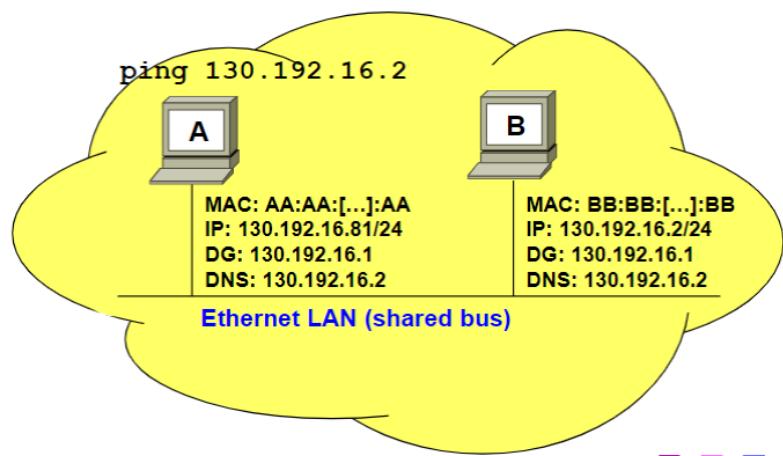
Realizzare un piano di indirizzamento che minimizza il numero di indirizzi necessari. Utilizzare il risultato della routing table di R1.



**Figura 1.14:** Esercizio 9

#### 1.6.1.9 Esercizio 9

Assumendo di avere interamente la cache libera, indicare il numero e il tipo di frames catturati da uno sniffer localizzato nella rete cablata dell'host A.



**Figura 1.15:** Esercizio 10

In una macchina Windows il ping viene eseguito 4 volte.

Bisogna innanzitutto verificare che le due macchine siano effettivamente nella stessa rete, lo si fa vedendo se hanno la stessa sottorete (in questo caso sì, entrambi coerenti sulla 130.192.16.0/24).

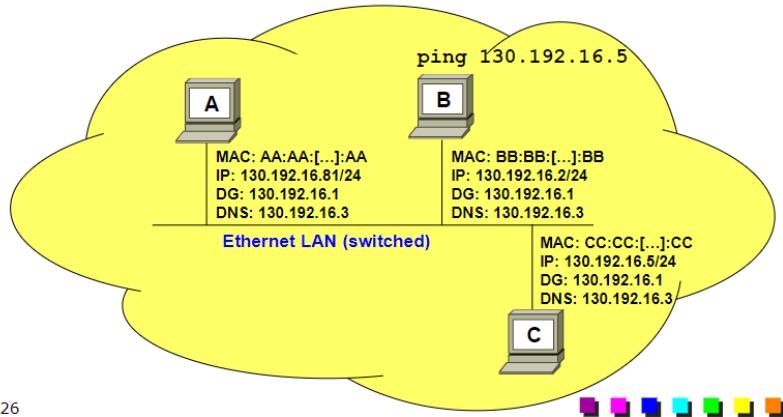
Scriviamo ora la tabella:

ID	MACS	MACD	IPS	IPD	DESCRIZIONE
1	MACA	broadcast	-	-	ARP Request
2	MACB	MACA	-	-	ARP Response
3	MACA	MACB	IPA	IPB	ICMP echo request
4	MACB	MACA	IPB	IPA	ICMP echo response

Il passaggio 3 e 4 sono quelli eseguiti 4 volte.

#### 1.6.1.10 Esercizio 10

Assuming that all caches are empty, indicate the number and the type of the frames captured by a sniffer located sulla rete dell'host A.

**Figura 1.16:** Esercizio 10

L'indirizzo IP del DNS è in realtà l'indirizzo di un host in quanto l'indirizzo della sottorete, con prefix length pari a /23 abbiamo 130.192.16.0/23 (osservando il router). Il relativo indirizzo di broadcast viene calcolato sapendo di avere gli ultimi 9 bit a 1, quindi 130.192.17.255, quindi l'indirizzo fornito è incluso.

La sottorete di A ha indirizzo della sottorete pari a 130.192.16.0, è errato il prefix length in quanto viene indicato /24 invece di /23.

A quando comunica per parlare con il DNS, che è all'esterno della sua sottorete, parla con il suo default gateway.

ID	MACS	MACD	IPS	IPD	DESCRIZIONE
1	MACA	broadcast	-	-	ARP Request
2	MACDG	MACA	-	-	ARP Response
3	MACA	MACDG	IPA	IPDNS	DNS request
4	MACDG	broadcast	-	-	ARP request
5	MACDNS	MACDG	-	-	ARP response
6	MACDG	MACDNS	IPA	IPDNS	DNS request
7	MACDNS	broadcast	-	-	ARP request
8	MACA	MACDNS	-	-	ARP response
9	MACDNS	MACA	IPDNS	IPA	DNS response
10	MACA	MACDG	IPA	IP google	ICMP echo request

ID	MACS	MACD	IPS	IPD	DESCRIZIONE
11	MACDG	MACA	IP google	IPA	ICMP echo response

Essendo uno shared bus tutti i pacchetti sono condivisi, solo che chi non è interessato ai pacchetti che riceve li scarta. *Nota: DG viene utilizzato per indicare default gateway; arp è di livello 2.* Il traffico viene ottenuto prima che entri nel nodo A.

Il passaggio 10 e 11 sono quelli eseguiti 4 volte.

## 1.7 Multicast

Il **multicast** è un concetto che sta nel mezzo tra una comunicazione *unicast* (*1 a 1*) e *broadcast* (*1 a tutti*). I pacchetti vengono indirizzati da una sorgente verso multiple destinazioni, quindi verso solo *alcuni* host. Sono dunque presenti dei gruppi a cui degli host possono entrare o uscire. E' vantaggioso in quanto l'alternativa sarebbe mandare pacchetti uno ad uno in modo molto più lento. Nel multicast viene inviato un solo pacchetto, che viene poi instradato correttamente dal router ai destinatari utilizzando meno traffico (nel broadcast è sempre un pacchetto, ma viene poi mandato anche ai non interessati).

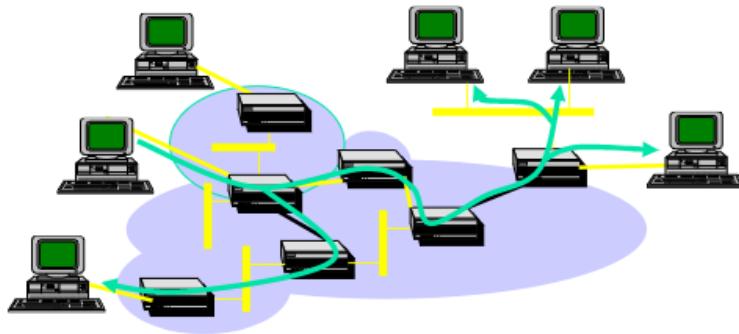
Il multicast è implementato in *IPv4* attraverso l'utilizzo del protocollo aggiuntivo **IGMP**, il quale consente a un router *IPv4* di scoprire quali gruppi multicast sono presenti in una rete ad esso direttamente connessa e permette a un host di comunicare ad altri *router* il proprio interesse nel ricevere il traffico di un determinato gruppo multicast.

In *IPv4* viene utilizzato poco a causa dei problemi con l'indirizzamento, ma al contrario è ampiamente utilizzato in *IPv6* dove risulta chiave per la comunicazioni tra gruppi (videoconferenze, video broadcast ecc.).

La richiesta da parte di un host di ricevere le informazioni di un dato gruppo multicast è mandata ai **router** e non direttamente agli host.

A ogni gruppo multicast viene associato un indirizzo *IPv4* di classe **D** identificato dai primi bit posti a **1110**. Fanno parte del range **224.0.0.0 - 239.255.255.255** che essendo riservati è necessario acquistarne uno per utilizzarli.

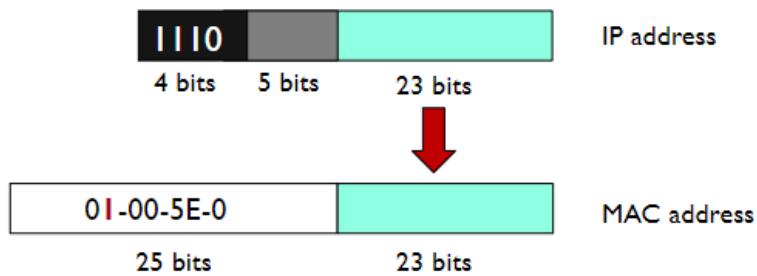
Il protocollo prevede che il *group delivery* venga eseguito al livello 2, occupandosi di scartare i pacchetti che non sono di interesse. Nonostante ciò è comunque possibile associare un indirizzo di livello 2 a uno di livello 3 in modo che possa essere scartato successivamente. Questo avviene attraverso un



**Figura 1.17:** Multicast

mapping: L'indirizzo *MAC* è formato da 48 bit di cui la parte alta, solitamente riservata al produttore, ha invece la costante 01-00-5E-0 che identifica la mappatura per un totale di 25 bit (l'ultimo gruppo è solo un bit), mentre la parte bassa è formata dai 23 bit meno significativi del *IP address* (non comprendendo tutti i casi ma cercando di ridurre il numero di collisioni).

I gruppi multicast vengono identificati da particolari indirizzi IP che non possono essere assegnati alle singole stazioni.



**Figura 1.18:** Mappatura IP a MAC

I router trovano i dispositivi su ciascuna *LAN* mediante il protocollo **IGMP** (Internet Group Management Protocol), annunciando gli host di un gruppo ai rimanenti mediante dei protocolli di routing appositi per il multicasting. Si forma così tra i router un albero di distribuzione per ogni gruppo verso tutte le *LAN* con almeno un partecipante.

Una stazione **è sempre raggiunta** da un pacchetto multicast relativo a un particolare gruppo anche se non vi è iscritta, in quanto in tal caso verrà scartato a livello 2. **Non è quindi vero** che una stazione consegna sempre a livello applicazione tutti i pacchetti multicast ricevuti.

### 1.7.1 Status

Attualmente, il multicast in IPv4 non è molto supportato in quanto non è in grado di rispondere alle esigenze comuni di *control engineering* e *traffic engineering*.

Spesso l'utilizzo avviene in ambienti limitati, come il video broadcasting su soluzioni IP.



## 2 IPv6

**IPv6** nasce per soddisfare le esigenze di un **maggior numero di indirizzi**, superando i limiti di *IPv4*. La nuova versione del protocollo risulta superiore sotto molti punti di vista, ma nonostante ciò *IPv4* è ancora largamente utilizzato e non è stato completamente sostituito e anzi, al contrario, nel corso degli anni è stato ulteriormente esteso e migliorato.

Altre motivazioni che hanno portato alla nascita di *IPv6* sono:

- Più **efficiente** sulle LAN
- Supporto di **Multicast** e **Anycast**
- Sicurezza
- Policy routing
- Plug and Play
- Traffic Differentiation
- Mobility
- Supporto alla Quality of Service

Riuscire a definire il protocollo *IPv6* ha richiesto molto tempo, attualmente è in una fase di migrazione (utilizzando soluzioni temporanee applicate su *IPv4*).

E' dunque lecito dire che ciò che ha comportato la nascita di *IPv6* è l'inefficienza del piano di indirizzamento di *IPv4*.

### 2.1 Perché *IPv4* non basta e soluzioni

Il protocollo *IPv4* ha indirizzi di lunghezza pari a **32 bit**, con un totale di circa **4 miliardi** di indirizzi. Nonostante ciò, solo parte di questi indirizzi possono essere effettivamente utilizzati a causa dell'utilizzo di classi, multicast, ecc... Inoltre, molti di questi sono utilizzati in modo gerarchico: il prefisso usato in una rete fisica non può essere usato in una differente. Infine, molti indirizzi IP risultano non utilizzati, causando un grande spreco.

Alcune delle soluzioni utilizzate per risolvere tali problemi sono:

- Introduzione di reti “su misura” mediante l’utilizzo di netmask.
- Utilizzo di indirizzi privati (intranet), ma non è abbastanza da risolvere il problema.
- *NAT*, che però annulla la connessione end to end aumentando il carico dei gateway e la relativa complessità.
- *ALG* (Application Layer Gateway).

## 2.2 Chi assegna gli indirizzi IP

Gli indirizzi IP vengono assegnati da parte dell’organizzazione **IANA**, che fornisce a ciascun *Regional Internet Registry (RIR)* un blocco di /8 indirizzi ip:

- *AFRINIC*: Africa
- *APNIC*: East Asia, Australia and Oceania
- *ARIN*: USA, Canada and some Caribbean islands
- *LACNIC*: South America, Mexico and some Caribbean islands
- *RIPE NCC*: Europe, Middle East and Central Asia

Successivamente, le *RIR* dividono i blocchi in blocchetti più piccoli di dimensione minore da assegnare alle *National Internet Registries (NIR)* e alle *Local Internet Registries (LIR)*.

## 2.3 Address pool status e scalabilità

Ogni singolo indirizzo IPv4 può essere in uno dei seguenti stati:

- far parte del pool di indirizzi **non allocati da IANA**
- far parte del pool di indirizzi **non allocati da RIR**
- assegnato a un *end user entity* ma non annunciato dal *BGP (Border Gateway Protocol)*
- assegnato e annunciato dal *BGP*

Ciò comporta dei problemi anche in termini di scalabilità, dovuti:

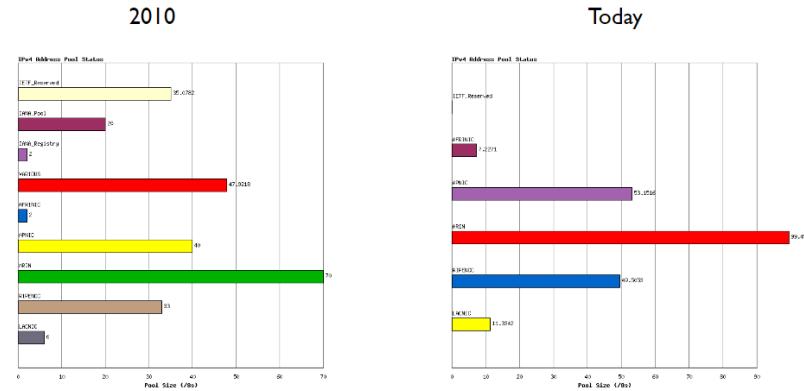
- **dimensione delle routing table**: ogni subnet network deve essere advertised.
- **Risorse** dei router **limitate**: troppe informazioni da gestire.
- **Limitazioni** dei **protocolli** di routing: spesso i router cambiano e con loro anche i protocolli (perlopiù riguarda i *router backbone*).

Sono state tentate alcune soluzioni, come:

- aggregazione di router mediante prefissi più piccoli

- *CIDR* (Classless Inter-Domain Routing)
- Limitazione di assegnamento di prefissi IP “non razionali” e indirizzi IP (es vendita di /8)

Ma nonostante ciò il problema persiste, in particolare la scalabilità dei protocolli di routing risulta attualmente non risolvibile.



**Figura 2.1:** Status degli address pool

## 2.4 Notazione

E' stato scelto, attraverso un approccio scientifico e con un focus sull'efficienza, l'utilizzo di indirizzi di lunghezza pari a **128 bit**, con un totale di  $2^{128}$  indirizzi.

La notazione utilizzata non è più puntata, ma utilizza gruppi di **2 byte** (*4 cifre esadecimali*) separati dal carattere **:**.

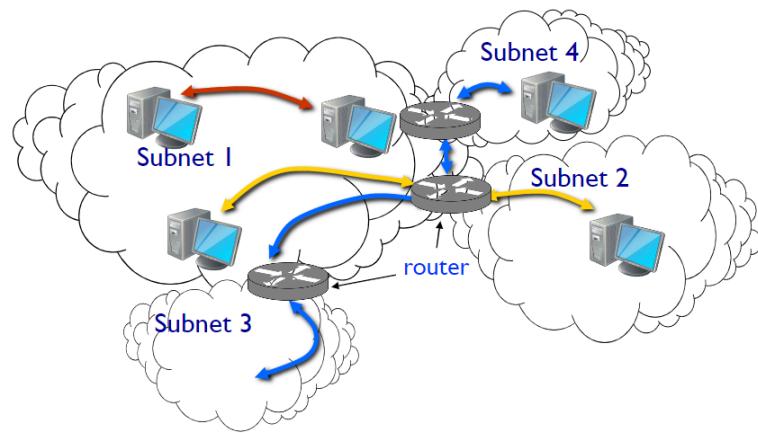
Tale notazione può essere resa più compatta nei seguenti modi:

- è possibile rimuovere i gruppi pari a 0000 comprimendoli in 0 (o gruppi aventi degli zeri all'inizio). Esempio: da 1080:0000:0000:0007:0200:**A00C**:3423:**A089** a 1080:0:0:0:7:200:**A00C**:3423:**A089**.
- e' possibile omettere un gruppo di soli zeri inserendo **::** (1080::7:200:**A00C**:3423:**A089**), ma **solo una volta**. Questo perché in caso contrario non sarebbe possibile sapere il numero di zeri omessi.

Se mettessimo **FEDC::0876:45FA:0562::3DAF:BB01** avremmo raffigurati 12 dei 18 byte, ma non saremmo in grado di dedurre in che modo sono distribuiti i 6 byte mancanti tra i due **::**.

## 2.5 Routing

Il routing *IPv6* è stato pensato in modo da **non modificare** la struttura adoperata in *IPv4*, a eccezione della lunghezza degli indirizzi.



**Figura 2.2:** Routing

Per dividere la parte del prefisso di rete e la parte dell'interfaccia si è deciso, per il momento, di applicare una separazione a metà con un prefisso di rete pari ad  $n=64$ , ma è previsto che in futuro potremmo aver bisogno di un prefisso di rete più lungo.

Il concetto di aggregazione rimane il medesimo, è infatti possibile utilizzare il *prefix length* come già visto, ad esempio: **FEDC:0123:8700::100/40**. Non è più necessario l'utilizzo di classi.

**Nota:** il prefix length non sarà, per quanto detto precedentemente, superiore a 64. Per quanto attualmente sia fisso a 64 è comunque pensato per essere **flessibile**.

Nonostante ora sia definito solo a 64, rimane comunque vero che la flessibilità.



$n=64$

**Figura 2.3:** Struttura dell'indirizzo

Sono però presenti alcune differenze in quanto a terminologia:

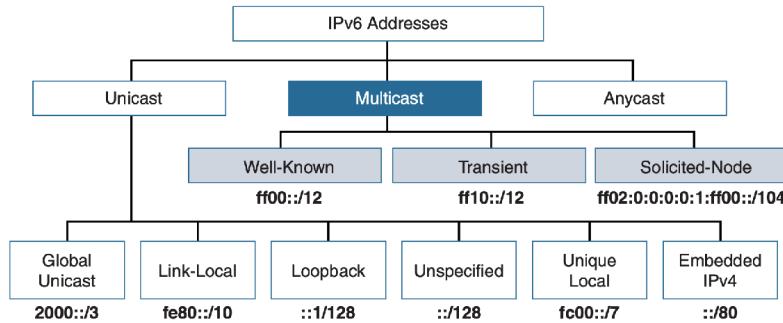
- **Link:** physical network.

- **Subnetwork:** Link, set di host con lo stesso prefisso.

Dividiamo le comunicazioni in:

- **On-link:** gli host hanno lo *stesso prefisso*, comunicano direttamente tra loro all'interno della stessa sottorete.
- **Off-link:** gli host hanno un *prefisso diverso*, comunicano attraverso un router.

A loro volta è possibile ulteriormente suddividere gli indirizzi di rete:



**Figura 2.4:** Spazio di indirizzamento

## 2.6 Multicast

L'equivalente dell'indirizzo multicast IPv4 `224.0.0.0/4` è `FF00::/8`, che si suddivide in:

- **Well-known Multicast:** `FF00::/12`, utilizzato per comunicazioni di servizio e vengono assegnati a gruppi di dispositivi, sono riservati. Un esempio è l'indirizzo di *Google*.
- **Transient:** `FF10::/12`, indirizzi transitori, assegnati dinamicamente da applicativi multicast (*corrispettivo della vecchia modalità multicast in IPv4*).
- **Solicited-node Multicast:** `FF02:0:0:0:1:FF00::/104`, simile a un indirizzo IP broadcast in ARP.

Una caratteristica importante è la **scomparsa in IPv6 l'utilizzo del broadcast**, che in seguito alle evoluzioni ha dimostrato essere un rischio per la sicurezza.

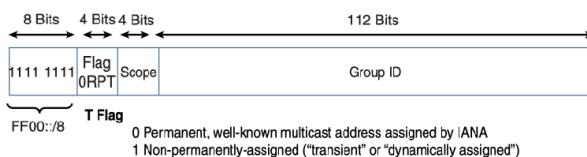
L'indirizzo si scomponete in:

- **8 bit** iniziali, identificano che è un indirizzo multicast (tutti i bit sono posti a 1).
- **4 bit** per il **T flag**, specifica se è *well known* (permanente) o *transient* (non permanente), viene assegnato da IANA.
- **4 bit** per lo *scope*, consente ai dispositivi di definire il range dei pacchetti multicast.

- **112 bit** per il group ID.

**Non è vero** che in *multicast* il pacchetto viene inoltrato su tutte le porte tranne quella da cui arriva (sarebbe broadcast).

Il router manda un pacchetto *multicast* solo sulle porte in cui è possibile raggiungere i dispositivi appartenenti al gruppo *multicast*.



**Figura 2.5:** Struttura indirizzo multicast

## 2.7 Unicast

Gli indirizzi **Unicast**, anche denominati *aggregatable global unicast addresses*, continuano a essere disponibili in IPv6, si suddividono in:

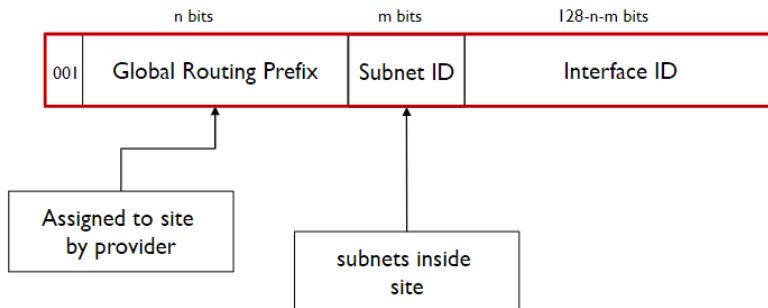
- 2000::/3, **Global Unicast**
- FE80::/10, **Link-Local**
- ::1/128, *Loopback* (in IPv4 era 127.0.0.1)
- ::/128, *Unspecified*
- FC00::/7, *Unique Local*
- ::80, *Embedded IPv4*

Sono indirizzi di tipo aggregato, utilizzati in modo equivalente agli indirizzi pubblici in IPv4. Hanno la caratteristica di essere **raggiungibili e indirizzabili globalmente**, oltre a essere *plug and play*. Attualmente sono disponibili in un range definito tra 3FFF:: e 2000::. Questi indirizzi hanno i primi 3 bit (più significativi) posti a 001.

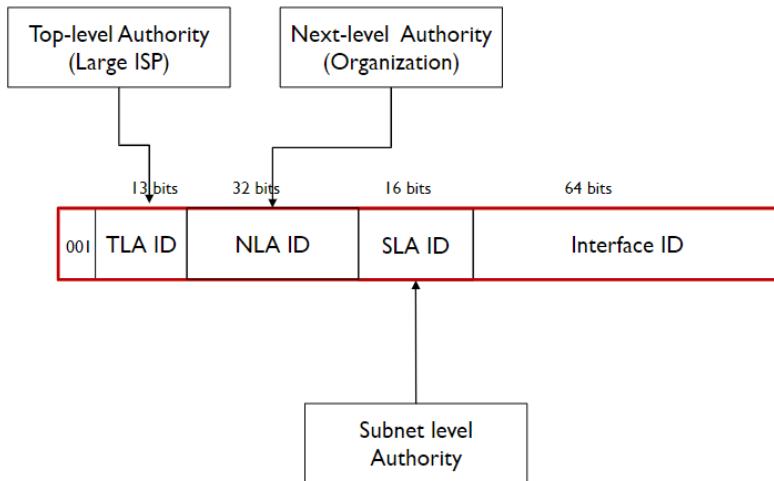
Hanno la caratteristica di essere distribuiti geograficamente in modo **gerarchico**.

I prefissi per il *Global Routing* sono formalmente assegnati da *multi-level authorities*:

- **3 bit**, tipologia (001).
- **13 bit**, TLA ID (*Top Level Authority, grandi ISP*)

**Figura 2.6:** Global Unicast Addresses

- **32 bit**, NLA ID (*Next-level Authority, organizzazioni*)
- **16 bit**, SLA ID
- **64 bit**, Interface ID

**Figura 2.7:** Global Routing Prefix

### 2.7.1 Link local/site local Addresses

I **link local/site local** sono un gruppo di indirizzi compresi tra **FE80** ed **FEBF** e vengono assegnati in **automatico** ai link quando viene acceso un router.

Gli indirizzi **Link local**, identificati nella rete **FE80 :: /64**, vengono assegnati quando più router devono parlare tra di loro oppure devono annunciarsi a un router vicino, oltre a consentire una configurazione automatica o quando un router non è presente. Sono normalmente assegnati automaticamente dalla stazione a partire dall'indirizzo **MAC** della scheda di rete, a cui si pre-pende un prefisso predefinito.

Gli indirizzi **site local** sono nella rete **FEC0 :: /10** e sono ormai ritenuti **deprecati** perché pensati come vecchi indirizzi privati riconfigurabili, possono avere assegnati i router nelle comunicazioni (tipo stella, mesh ecc...). Utilizzano comunicazioni dirette e possono essere assegnati solo a indirizzi di rete.

Quando il dispositivo si accende, dunque, prenderà in automatico un indirizzo link local dipendente dal MAC della propria scheda di rete.

## 2.7.2 Unique Local Addresses

Gli **Unique Local Addresses** (*ULA*) possono essere utilizzati in modo simile agli indirizzi globali *unicast*, ma sono per un utilizzo privato e non per l'indirizzamento sull'internet. Sono identificati da **FFC00 :: /7** e vengono utilizzati dai dispositivi che non hanno necessità di connettersi ad internet o di essere raggiungibili dall'esterno. Sono indirizzi privati che possono comunicare su internet grazie ad operazioni di tunneling.

L'**ottavo** bit è il **Local (L) Flag**, che divide in:

- **FC00 :: /8**, se L flag è 0, potrebbe essere assegnato in futuro
- **FD00 :: /8**, se L flag è 1, l'indirizzo è assegnato localmente

Attualmente gli indirizzi **FD00 :: /8** sono gli unici indirizzi *ULA* validi. Sono dunque privati e non utilizzati da altri dispositivi.

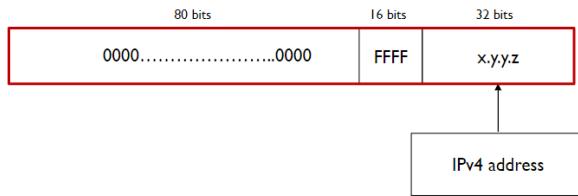


**Figura 2.8:** Unique Local Addresses

Dopo i primi 8 bit, sono presenti 40 bit generati casualmente in modo da non avere collisioni con altri indirizzi.

## 2.7.3 IPv4 Embedded Addresses

Gli **IPv4 embedded addresses** sono utilizzati per rappresentare indirizzi *IPv4* all'interno di un indirizzo *IPv6*. Vengono utilizzati per facilitare la transizione tra i due protocolli. L'indirizzo *IPv4* è inserito negli ultimi 32 bit (low order) mentre i primi 80 devono necessariamente essere pari a 0, a cui seguono 16 bit dal valore di **FFFF** (sedici bit posti a 1).



**Figura 2.9:** Struttura indirizzi IPv4 Embedded

#### 2.7.4 Loopback Addresses

L'indirizzo di **loopback** viene utilizzato per finalità di test e consiste nel inviare un pacchetto IPv6 a se stesso. È identificato da `::1` (equivalente di `127.0.0.1` di *IPv4*) ed è subordinato alle medesime regole della versine precedente:

- Non può essere assegnato a un'interfaccia fisica.
- Pacchetti con un indirizzo di loopback non dovrebbero mai essere trasmessi oltre il dispositivo.
- I router non devono mai fare il forwarding di un pacchetto contenente un indirizzo di loopback.
- Il dispositivo deve fare il drop di pacchetti ricevuti da un'interfaccia se il destinatario è un indirizzo di loopback.

#### 2.7.5 Unspecified Addresses

Un indirizzo unicast non specificato è tale da contenere solo 0. Tale indirizzo viene utilizzato come sorgente per indicare l'assenza di un indirizzo. Non può essere assegnato a un'interfaccia e viene utilizzato per il duplicate address detection in *ICMPv6*.

### 2.8 Anycast Addresses

Gli indirizzi **anycast** possono essere assegnati a più di un'interfaccia (tipicamente su dispositivi differenti), dando dunque la possibilità di avere su dispositivi differenti lo stesso indirizzo *anycast*. Un pacchetto che viene inviato a un indirizzo *anycast* viene reindirizzato all'interfaccia più vicina avente quel indirizzo. Questo permette di avere un indirizzo unico per un servizio, ma che può essere raggiunto da più dispositivi.

Inizialmente venne realizzato per il *DNS*, ma è ancora in uno stato sperimentale.

**Nota:** molto utile, ma non è ancora utilizzato.

## 2.9 Protocolli utilizzati

L'architettura del protocollo *IPv6* è molto simile a quella di *IPv4*, ma presenta alcune differenze:

- **IP**: utilizzato, salvo alcune modifiche.
- **ICMP**: viene utilizzato *ICMPv6*.
- **ARP**: non più utilizzato, inglobato in *ICMPv6*.
- **IGMP**: non più utilizzato, inglobato in *ICMPv6*.

Sono invece stati aggiornati senza modifiche essenziali:

- DNS (type AAAA record)
- RIP e OSPF
- BGP e IDRP
- TCP e UDP
- Socket interface

**Attenzione:** non è più possibile utilizzare ARP E IGMP per risolvere gli indirizzi *IPv6*.

Il protocollo **IGMP**, non più presente in *IPv6*, permette ad un router *IPv4* di scoprire quali gruppi *multicast* sono presenti in una rete ad esso direttamente connessa.

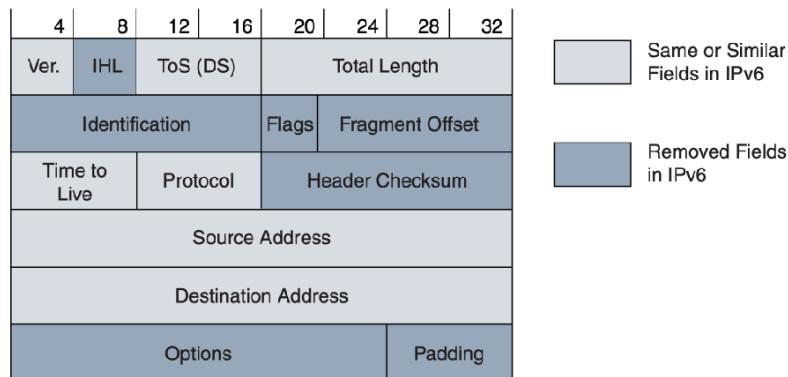
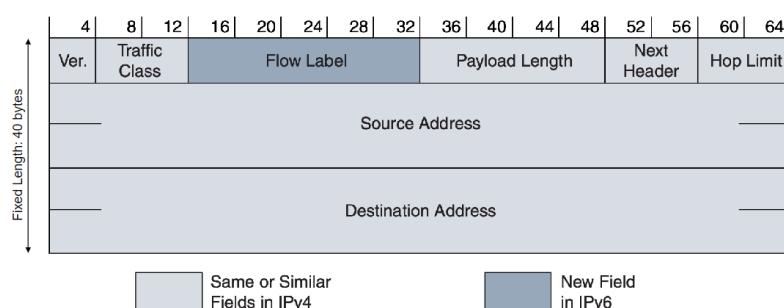
## 2.10 Packet Header Format

L'header è stato modificato in modo sostanziale in seguito all'introduzione di *IPv6*. Ciò è stato fatto al fine di avere un header il più snello possibile, ottenendo una lunghezza di **40 byte**.

L'header utilizzato in *IPv6* è invece il seguente:

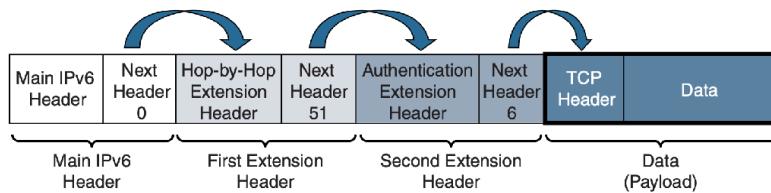
Osservando le immagini si può notare come alcune informazioni siano state rimosse:

- **Header Checksum**: Non è più presente. Veniva utilizzato per verificare se il dato trasmesso è corrotto, ma non è più necessario in quanto a livello *data link* (2) vengono eseguiti già controlli di errore. Analogamente, i protocolli di livello superiore come *UDP* e *TCP* hanno i propri meccanismi di checksum.
- **Frammentazione**: I router *IPv6* non frammentano i pacchetti a meno che non siano loro la sorgente. I pacchetti più grandi di *MTU* vengono scartati e viene restituito alla sorgente un messaggio di errore **ICMPv6 Packet Too big**.

**Figura 2.10:** Header IPv4**Figura 2.11:** Header IPv6

**Nota:** Il checksum su *UDP* diventa opzionale in *IPv6*.

L'header può essere ulteriormente esteso attraverso il campo **next header**, che consente di puntare a un altro header con ulteriori informazioni creando una catena di header. Funzionano in modo simile al campo "protocol" di *IPv4*.



**Figura 2.12:** Chaining

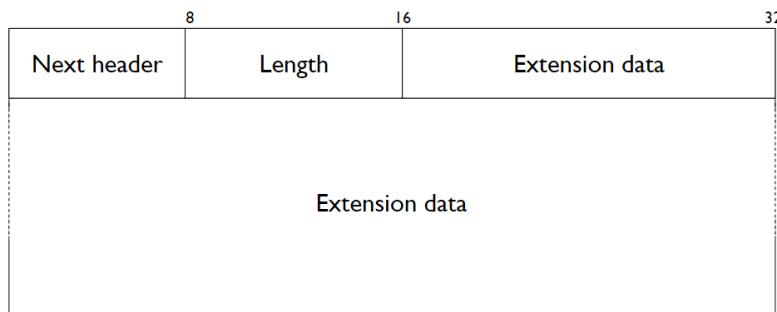
Inoltre, sono presenti:

- **version:** versione del protocollo.
- **traffic class:** permette di indicare la priorità del traffico (*quality of service*).
- **flow label:** permette di indicare il flusso di dati (nuovo campo), associando un'etichetta a un certo tipo di traffico (*label routing*). Un esempio è il caso di un datore di lavoro che non si fida dei suoi dipendenti e vuole che tutto il loro traffico passi per un dispositivo di sicurezza che lo analizzi.
- **payload length:** lunghezza del payload.
- **hop limit:** numero di router che possono essere attraversati prima che il pacchetto venga scartato. Se il valore è 0, il pacchetto viene scartato. Se il valore è 1, il pacchetto viene inviato al destinatario senza essere ulteriormente inoltrato. Se il valore è 255, il pacchetto non viene scartato mai.

Il formato del campo **next header** è il seguente:

- **next header:** indica il tipo di header successivo.
- **length:** lunghezza del header successivo.
- **extension header:** header successivo.
- **extension data:** dati dell'header successivo.

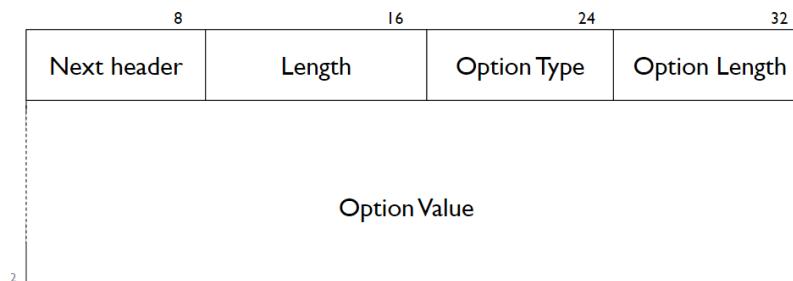
**Nota:** Header length non serve più! Viene eseguita la frammentazione attraverso il next header.

**Figura 2.13:** Extension Header Format

### 2.10.1 Hop-by-Hop Extension Header

L'**Hop-by-Hop Extension Header** è utilizzato per andare a inserire dei campi/vincoli che servono all'hop per capire se il pacchetto deve essere scartato o meno (strumento di analisi). Se è presente, è indicato immediatamente dopo l'header IPv6. Questo header viene utilizzato per inserire dei campi opzionali. Ogni opzione ha un set di:

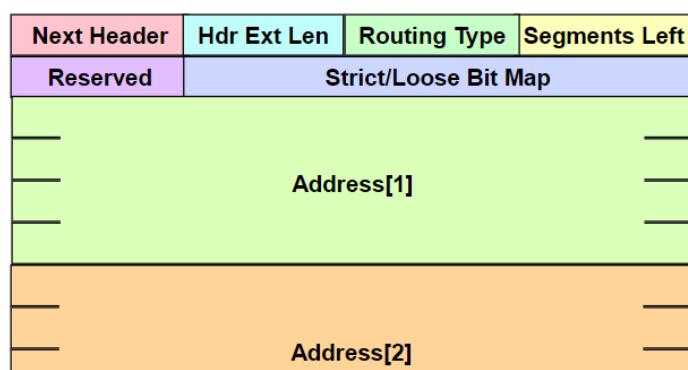
- **option type:** indica il tipo di opzione.
- **option length:** lunghezza dell'opzione.
- **option value:** valore dell'opzione.

**Figura 2.14:** Hop-by-Hop Extension Header

**Nota:** si ottiene una tripletta **TLV** (type-length-value).

### 2.10.2 Routing Extension Header

Il **Routing Extension Header** permette alla sorgente di un pacchetto di specificare il percorso di destinazione, indicando uno o più router intermedi. Viene utilizzato per il supporto alla mobilità in IPv6.



**Figura 2.15:** Routing Extension Header

### 2.10.3 Altre estensioni

Sono possibili altri due tipi di estensioni a seconda delle necessità:

- **Fragment Extension Header**
- **Authentication and Encapsulating Security Payload Extension Headers:** utilizzato da IPsec (una suite di protocolli per la sicurezza).

#### 2.10.3.1 fragmentation header

Il **Fragmentation header** viene utilizzato per la frammentazione dei pacchetti ognuno dei quali ha un proprio header IPv6 e un frammento di extension header. Il ricevente del pacchetto deve riunire i frammenti in un unico pacchetto. A differenza di IPv4, il protocollo IPv6 non frammenta un pacchetto a meno che non sia la sorgente del pacchetto.

#### 2.10.3.2 Authentication and Encapsulation Header

Gli header **Authentication and Encapsulation Header** vengono utilizzati per la sicurezza, sono usati da IPsec: una suite di protocolli per l'invio in sicurezza dei pacchetti in una rete IP. *Authentication Header* (AH) è utilizzato per l'autenticità e la integrità dei pacchetti mentre *Encapsulating Security Payload* (ESP) è utilizzato per la cifratura, autenticazione e integrità dei pacchetti.

## 2.11 Interfacciarsi con i livelli più bassi

### 2.11.1 Incapsulamento

La prima cosa che risulta evidente appena si approccia *IPv6* è che lo stack *ISO/OSI* prevede un campo in cui viene specificato il contenuto del livello superiore. Questo approccio è detto **dual stack**: creando uno nuovo stack è possibile far funzionare i dispositivi sia in *IPv4* che in *IPv6* (lo trattiamo come un nuovo protocollo), senza alterare il funzionamento nel protocollo precedente.

I pacchetti *IPv6* sono incapsulati nel frame di *livello 2*, ad esempio per ethernet il tipo è 86DD.

Un dispositivo con un interfaccia *IPv6* e una *IPv4* può ricevere pacchetti *IPv4* e *IPv6*, come avverrebbe normalmente.

### 2.11.2 Address mapping

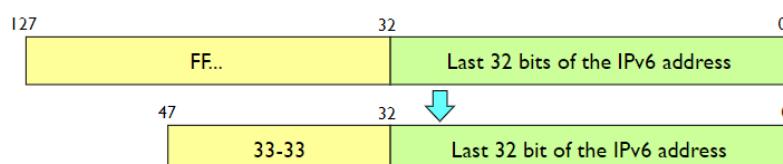
Un indirizzo di un pacchetto *IPv6* viene associato a un *MAC* di destinazione attraverso:

- **IP unicast address**: mediante discovery procedurale (protocol based) o *neighbor discovery*.
- **IP multicast address**: utilizza un algoritmo di mapping.

### 2.11.3 IPv6 Multicast transmission

La trasmissione **Multicast** si basa sul **ethernet multicast**, e a differenza del *ethernet broadcast* può essere filtrato dalla scheda di rete (*NIC*).

Gli indirizzi multicast *IPv6* vengono associati ad indirizzi *MAC*, in particolare è riservato l'indirizzo *MAC Ethernet* 33-33-**xx-xx-xx-xx** per il trasporto di pacchetti *multicast IPv6*. Questo viene fatto ponendo i **4 byte** (32 bit) meno significativi dell'indirizzo *IPv6* in un indirizzo *MAC Ethernet multicast* 33-33.



**Figura 2.16:** Multicast Transmission

Un esempio può essere il seguente: quando viene inviato un pacchetto all'indirizzo IP multicast FF0C : :89 : AABB : CCDD, questo viene incapsulato in un *MAC frame* con indirizzo 33 : 33 : AA : BB : CC : DD .

**Nota:** abbiamo FF all'inizio dell'indirizzo proprio perché è multicast.

## 2.12 Neighbor Discovery and Address Resolution

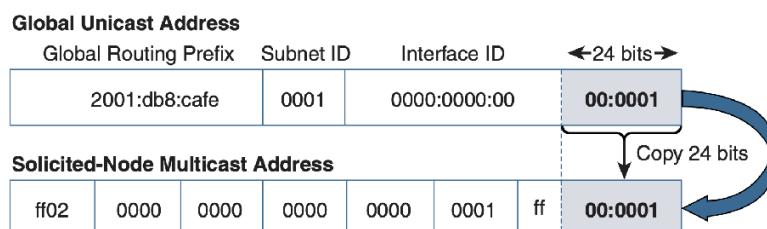
**ICMPv6** sostituisce completamente il protocollo **ARP**. E' basato su multicast e sfrutta il **Solicited-Node Multicast Address**. A causa di come il *multicast solicited address* è realizzato, per lo più solo un nodo viene coinvolto.

**Attenzione:** l'inoltre dei pacchetti *IPv6* su una *LAN* non utilizza meccanismi di *neighbor discovery* in quanto esiste una regola per mappare gli indirizzi *IPv6* in indirizzi *MAC* (4 byte meno significativi).

### 2.12.1 Solicited-Node Multicast Address

Mediante il **Solicited-Node Multicast Address**, gli indirizzi vengono automaticamente creati per ogni indirizzo *unicast* dell'interfaccia. Tutti gli host si iscrivono e vengono mappati nel seguente modo: FF : 02 :: 1 : FF / 104 | 24 ip meno significativi, in modo da ottenere per lo più un host per gruppo.

Il *solicited-node Multicast Address* viene utilizzato come indirizzo di destinazione in un pacchetto di *Neighbor Solicitation*.

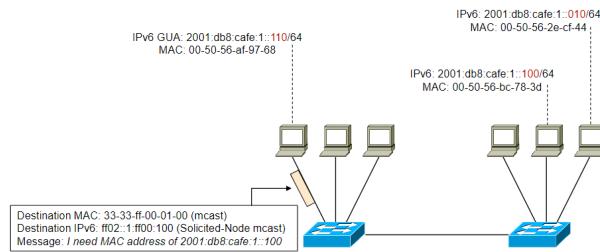


**Figura 2.17:** Mappatura indirizzo

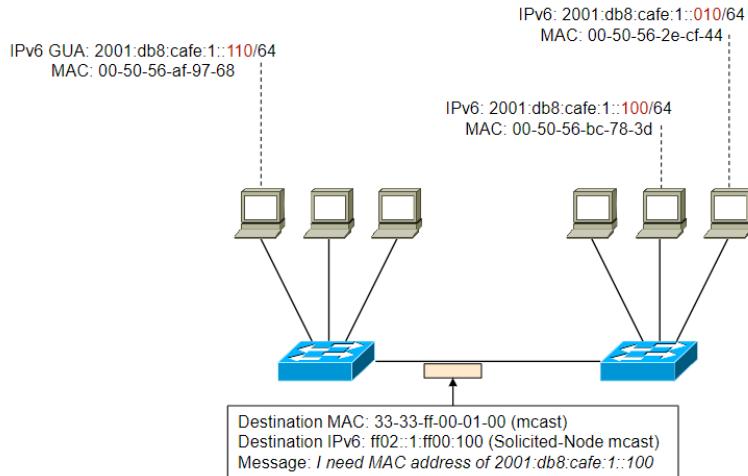
## 2.12.2 Risoluzione di un indirizzo

La risoluzione di un indirizzo avviene attraverso **ICMP Neighbor Solicitation**:

1. Il richiedente invia un frame al *Solicited Node Multicast Address* contenente l'indirizzo *IPv6* del host *target*.
2. Viene inviata dal target una risposta **ICMP Neighbor Advertisement**, attraverso la quale viene spedita indietro all'indirizzo unicast del richiedente la risposta. La mappatura tra *IPv6* e *MAC* address viene memorizzata nella cache dell'host (in modo equivalente alla cache *ARP*).

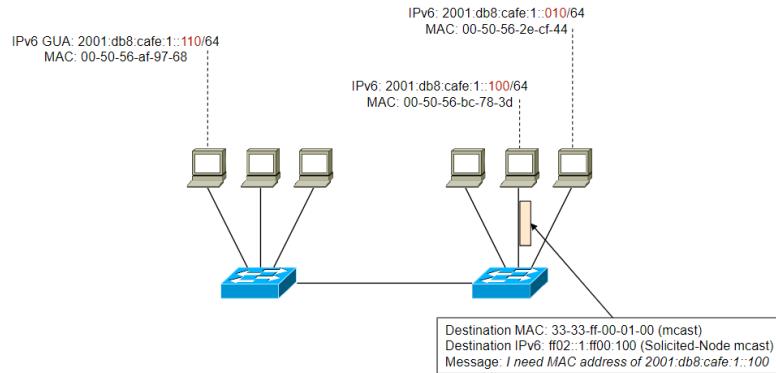


**Figura 2.18:** Risoluzione dell'indirizzo



**Figura 2.19:** Risoluzione dell'indirizzo

A causa della mancanza degli indirizzi *broadcast* il numero di *MAC* aumenta molto, per questo motivo è necessario che il router sia in grado di rispondere alle richieste di risoluzione di un indirizzo.



**Figura 2.20:** Risoluzione dell'indirizzo

## 2.13 La transizione tra IPv4 e IPv6

La transizione da *IPv4* a *IPv6* sta avvenendo in modo **incrementale**, in quanto non è stato stabilito un limite entro cui forzare il passaggio ma al contrario sarà stabilito automaticamente quando sarà, nel pratico, il più utilizzato. Questo approccio trasparente e graduale mira a far prendere piede a *IPv6* nel corso di molto tempo ma in modo **seamless** (ovvero senza cambiamenti) utilizzando approcci di tipo **dual stack**.

Questo risultato viene ottenuto attraverso tre meccanismi:

- **Address Mapping**
- **Tunneling**
- **Translation mechanisms**

Quando è nato *IPv6* erano presenti poche reti **dual stack** e una forte presenza di *IPv4* nella *backbone* (dorsale, architettura di rete).

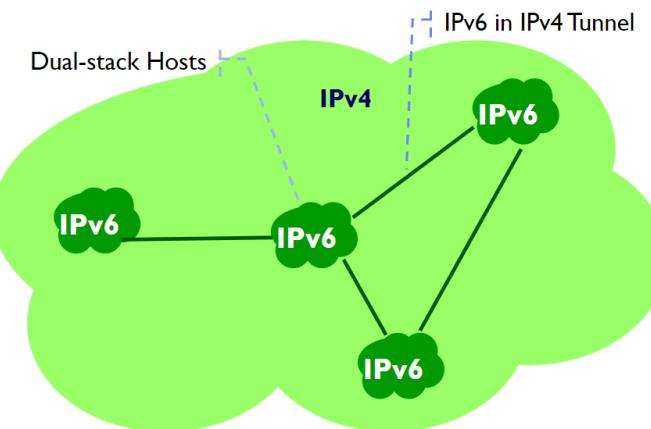
Nel corso del tempo le infrastrutture si sono adeguate al passaggio, aumentando il numero di host con comunicazioni on-link in *IPv6*.

L'obiettivo è quello di riuscire a creare una rete maggioritaria su *IPv6* con solo poche connessioni *IPv4* (in realtà le infrastrutture per eseguire il passaggio sono già pronte).

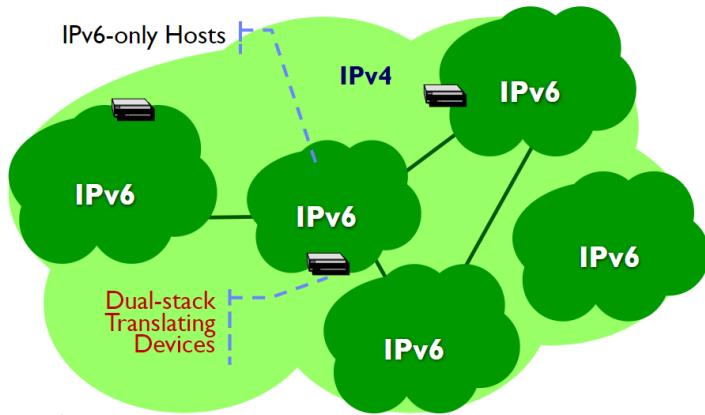
## 2.14 ICMPv6

Il protocollo **ICMPv6** permette di eseguire operazioni di:

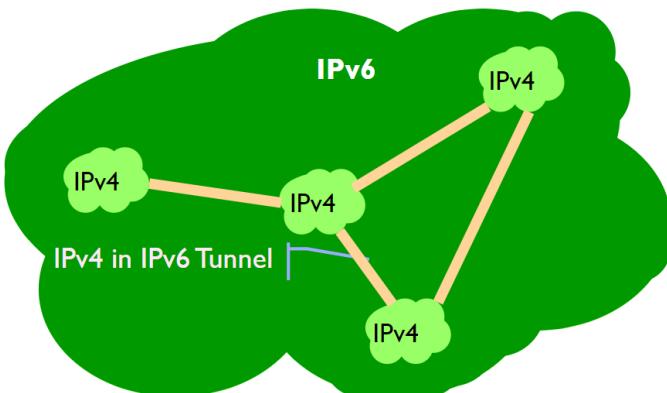
- diagnostica



**Figura 2.21:** Pochi host IPv6



**Figura 2.22:** Aumento di host IPv6



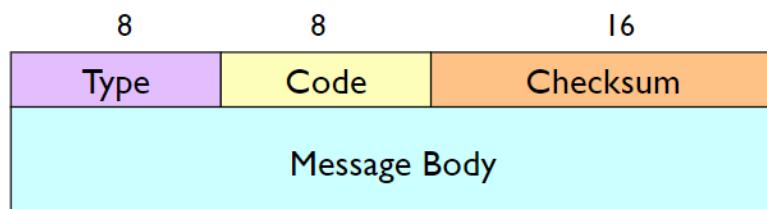
**Figura 2.23:** Maggioranza IPv6

- *neighbor discovery*
- *Multicast group management*
- *issue notification*

Inoltre, include alcune funzionalità che in IPv4 erano delegate ad **ARP** (*Address Resolution Protocol*) e **IGMP** (*Internet Group Membership Protocol*).

### 2.14.1 Formato del messaggio

Il messaggio è incapsulato in pacchetti IPv6 con `next header = 58`, che permette di identificare il nuovo header di tipo **ICPMv6**, che avrà al più **576 byte**.



**Figura 2.24:** Formato del messaggio

Code	Spiegazione	tipo
1	Destination Unreachable	Errore
2	Packet too big	Errore
3	Time exceeded	Errore
4	Parameter Problem	Errore
128	Echo Request	Informativo
129	Echo Reply	Informativo
130	Multicast Listener Query	Informativo
131	Multicast Listener Report	Informativo
132	Multicast Listener Done	Informativo
133	Router Solicitation	Informativo
134	Router Advertisement	Informativo
135	Neighbor Solicitation	Informativo

Code	Spiegazione	tipo
136	Neighbor Advertisement	Informativo
137	Redirect	Informativo

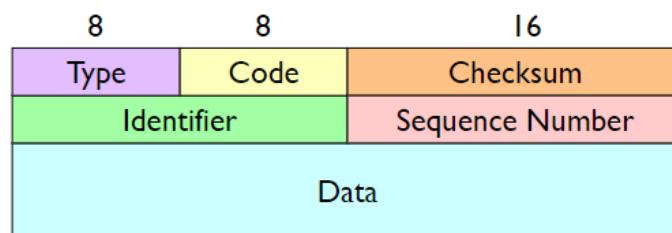
### 2.14.1.1 Messaggi di errore

Analizzando più nel dettaglio i messaggi di errore:

- **Destination unreachable** (*tipo 1*): solitamente generato dal router o firewall, nel campo “code” viene fornita la motivazione (nessuna route, scope errato, indirizzo/porta non raggiungibile).
- **Packet too big** (*tipo 2*): IPv6 non fa più la frammentazione dei pacchetti.
- **Time exceeded** (*tipo 3*): avviene quando il router riceve un pacchetto con `Hop Limit = 0`.
- **Parameter Problem** (*tipo 4*): generato quando un dispositivo trova un problema con un campo del header IPv6 main o con un extension header. Un esempio è un valore non valido del campo *Next Header*.

### 2.14.1.2 Messaggi informativi

**2.14.1.2.1 Echo** La richiesta di *echo* ha tipo 128 mentre la *echo reply* ha 129; viene utilizzato, ad esempio, da `ping`.

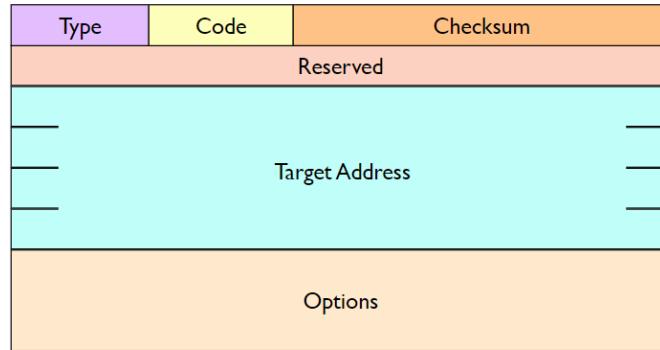


**Figura 2.25:** Echo request

### 2.14.1.2.2 Neighbor Solicitation

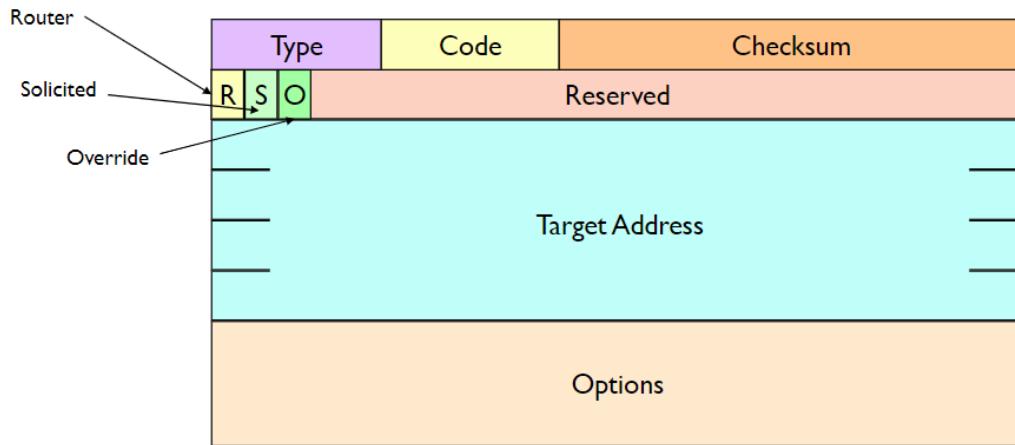
**2.14.1.2.3 Neighbor Advertisement** E’ importante evidenziare la presenza di flag aggiuntivi:

- `R router flag`, se `true` arriva da un router.
- `S solicited flag`, se arriva da un nodo che ha fatto una richiesta di risoluzione.



**Figura 2.26:** Neighbor Solicitation

- 0 **override flag**, se la host cache deve essere aggiornata o meno.



**Figura 2.27:** Neighbor Advertisement

**Nota:** non è presente un campo MAC, in quanto è dato per scontato sia presente nelle opzioni. Viene invece specificato l'ip, anche se ridondante, in quanto potrebbe essere sia un nodo che un router.

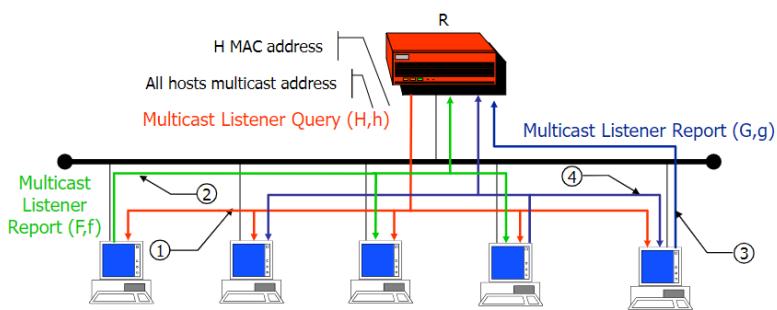
### 2.14.2 Multicast Group Management

Quando si ha un collegamento che fa affidamento al *data link layer multicasting services*, è necessario eseguire il *mapping* di un indirizzo *multicast IPv6* su un indirizzo *MAC*. Questo deve essere eseguito tra i link e i pacchetti inviati dai router in modo che *ICMPv6* sappia i membri *on-link* (ovvero gli host interessati a ricevere i pacchetti).

Inoltre consente ai protocolli di *multicast routing* di sapere quando sono presenti membri *off-link*.

### 2.14.3 Host Membership Discovery

La **Multicast Listener Query** è una domanda che il router manda ai suoi host per verificare se sono interessati a far parte di un gruppo *multicast*, ponendosi in attesa di una risposta. La risposta con la quale un host comunica al router tale interesse è detto **Multicast Listener Report**.



**Figura 2.28:** Host Membership Discovery

- **Multicast listener query** (`type=130`): il router manda una query per capire se un host è interessato a ricevere i pacchetti *multicast*.
- **Multicast Listener Report** (`type=131`): l'host risponde al router dicendo che è interessato a ricevere i pacchetti *multicast*.
- **Multicast Listener Done** (`type=132`): l'host manda un messaggio di fine per avvisare che non è più interessato a ricevere i pacchetti *multicast*.

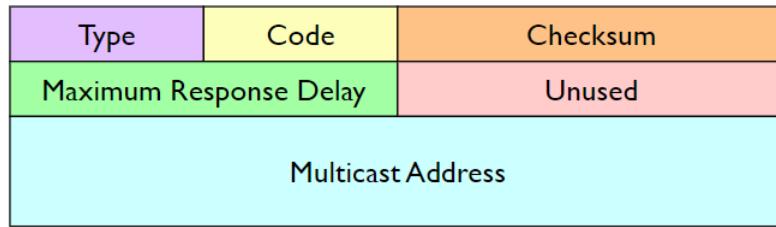
Il messaggio di *done* è importante, perché se un host esce da un gruppo, il router deve essere informato. Potrebbe succedere che il messaggio non venga inviato: in questo caso il router prevede dei timer, se dopo un intervallo di tempo (*maximum response delay*) l'host non manda un messaggio di interesse verso un gruppo, allora verranno inoltrati i pacchetti *multicast*.

Adesso la gestione del *multicast* viene rappresentato solo a livello 3 (quindi diviene compito del router e non più dello switch).

## 2.15 Device Configuration in IPv6

Le informazioni necessarie per eseguire la configurazione di un dispositivo sono:

- Address prefix



**Figura 2.29:** Formato richiesta

- Interface identifier
- Default gateway
- DNS server
- Hostname
- Domain name
- MTU (Maximum Transmission Unit)
- ...

Molte di queste informazioni vengono recuperate automaticamente in modo da rendere gli host plug and play.

Le configurazioni possono essere:

- **Manual configuration:** configurazione manuale.
- **Stateful configuration:** tutte le informazioni recuperate mediante *DHCPv6*.
- **Stateless configuration:** generate automaticamente, con il prefisso dell'indirizzo ottenuto dal router.
- **Hybrid** (Stateless DHCP): ulteriori informazioni oltre l'indirizzo recuperate mediante *DHCPv6*.

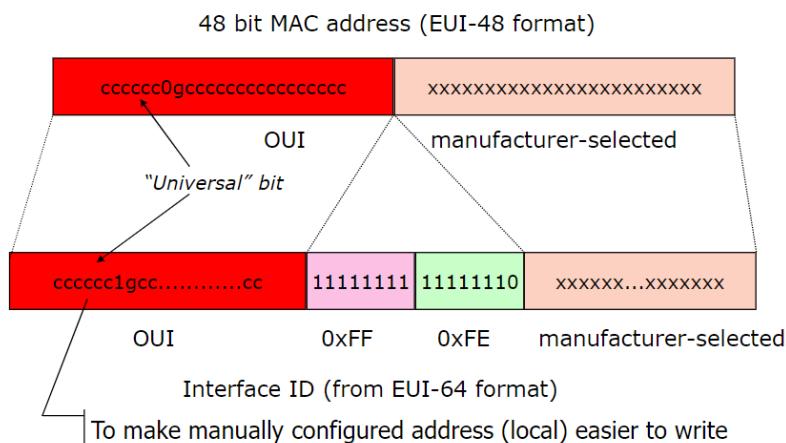
L'identificatore dell'interfaccia (*64 bit bassi*) può essere ottenuto in più modi:

- configurazione manuale
- ottenuto tramite *DHCPv6*
- generato automaticamente da *EUI-64 MAC address (privacy aware)*

Ci sarà in realtà un ulteriore meccanismo che si assicura che l'indirizzo utilizzato sia unico all'interno della rete.

**EUI-48 to EIU-64** (Extended Unique Identifier) estende un indirizzo *MAC* da 48 bit a 64 bit, aggiungendo i bit **11111110** e **11111111** in posizione 1 e 2.

La rete **FE80 :: /64** dovrebbe essere utilizzata solo per indirizzi di tipo link local (cioè per comunicare sulla stessa rete locale senza uscire sulla rete pubblica), per ottenere i 64 bit bassi di interface ID è



**Figura 2.30:** EUI-48 to EUI-64 mapping

necessario prendere il MAC address(48 bit), separare i 24 bit alti dai 24 bassi e al centro inserire 16 bit pari a **FFFE**, inoltre per convenzione il *settimo bit* deve essere post a 1 nel caso in cui l'indirizzo mac sia stato configurato manualmente.

Dal punto di vista della tracciabilità, i 64 bit meno significativi di un indirizzo IPv6 di un'interfaccia non cambiano mai quando viene utilizzato un MAC address.

Un esempio è il seguente: **FE80::0201:06FF:FEA5:3A4C** potrebbe avere come MAC associato **00:01:06:A5:3A:4C**.

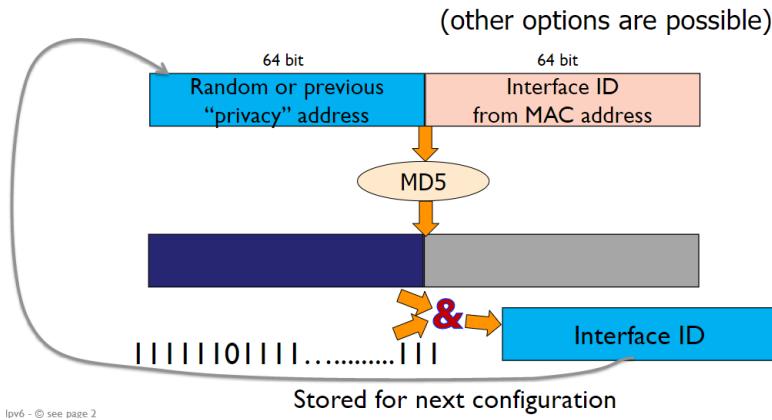
### 2.15.1 Privacy extension Algorithm

Per quanto detto fino ad adesso, riuscire a tracciare un dispositivo non è complicato in quanto i 64 bit meno significativi dell'indirizzo IPv6 non cambiano mai quando il MAC address è utilizzato.

Per questo motivo è stato implementato il **Privacy Extensions for Stateless Address Autoconfiguration in IPv6**, in modo da consentire una configurazione automatica e non tracciabile dei dispositivi.

Il **Privacy extension Algorithm** garantisce la privacy al livello 3 (network layer), in quanto non è possibile dai 64 bit ricavare l'indirizzo. Questo consiste nel unire i 64 bit del MAC address con i 64 bit di un *random number* (parte più significativa) dato poi in pasto a un algoritmo di *hashing* (come *md5*) in modo da non poter ricavare il *MAC* iniziale. Viene infine fatto una & bit a bit con 64 bit posti ad 1 ad esclusione del settimo bit più significativo che viene posto a 0 trovando un nuovo *random number* per la prossima configurazione.

Ormai da qualche anno, non viene più utilizzato MD5.



**Figura 2.31:** Privacy extension Algorithm

## 2.15.2 Indirizzi

Un host potrebbe avere più di un indirizzo IPv6 (*default* o *privacy aware*). Questi possono essere utilizzati per accettare o iniziare connessioni. Solo una un numero selezionato di indirizzi potrebbe essere disponibile per un user o una applicazione.

Il prefisso di un indirizzo può essere configurato manualmente, ottenuto tramite *DHCPv6*, generato automaticamente (link local) oppure ottenuto dal router.

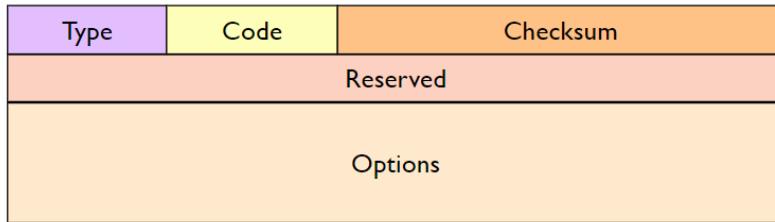
Come faccio a capire quali sono i 64 bit alti che ha comprato il mio amministratore di rete? Dal router. In particolare sono di nostro interesse il **router prefix discovery**, **router solicitation** e il **router advertisement**.

### 2.15.2.1 Router Prefix Discovery

Attraverso la **Router/Prefix Discovery** è possibile introdurre una “*sincronia*”: se l’host non ha chiesto un messaggio potrebbe essere direttamente il router a mandare l’informazione tempestivamente senza che venga richiesta un *solecition*.

### 2.15.2.2 Router Solicitation

Una **router solicitation** viene viene mandata solamente ai router, dunque non `all node` ma bensì `all routers (FF01::2)`.



**Figura 2.32:** Router Solicitation

### 2.15.2.3 Router Advertisement

Nel messaggio di advertisement sono rilevanti alcuni parametri:

- **M flag (Managed address Configuration)**: se è settato a 1 significa che l'indirizzo è stato configurato tramite DHCPv6.
- **O flag (other configuration)**: se è settato a 1 sono presenti altre configurazioni, ad esempio DNS server.
- **reachable time**: tempo in millisecondi che il router impiega per raggiungere un host.
- **retrans timer**: intervallo di tempo per cui ritenere l'indirizzo valido.
- **Option**: sono presenti delle opzioni, in formato generico ovvero type, length (multipli di 8) e value.

tra le opzioni c'è il prefix information option che ha sempre:

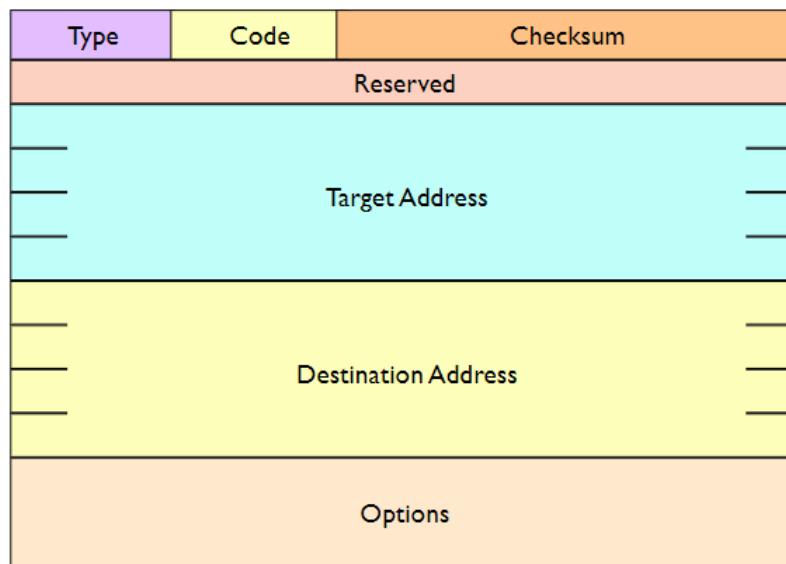
- **lifetime**: tempo di vita dell'indirizzo.
- **preferred lifetime**: periodo in cui non dovrei più utilizzarlo.
- **L**: se è utilizzato all'interno di un on-link.
- **A**: il prefisso può essere utilizzato per una configurazione automatica.
- **prefix**: il prefisso.

Link layer address option: indirizzo MAC del mio default gateway. Se il default gateway invia il messaggio perché lo inserisco? per comodità dello stack iso/osi.

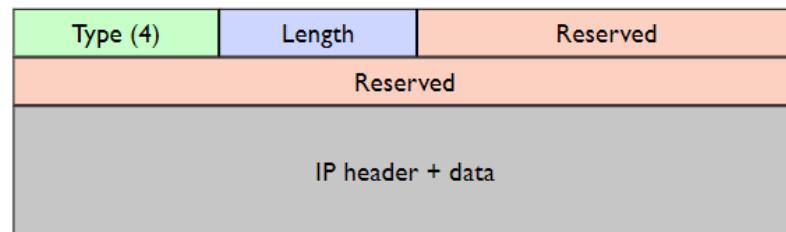
### 2.15.3 ICMP Redirect

Il **redirect** viene utilizzato per informare, all'interno di una stessa sottorete, un host **A** che per raggiungere un determinato host **B** è più conveniente utilizzare un altro router.

Se la comunicazione è a livello globale questo solitamente non avviene.



**Figura 2.33:** Message Format



**Figura 2.34:** header Option

### 2.15.4 Duplicate Address Detection (DAD)

Il **Duplicate Address Detection** (DAD) è un meccanismo che permette di verificare che un indirizzo IPv6 sia unico all'interno della rete.

Il funzionamento è molto semplice: l'host manda un messaggio *ICMPv6* a tutti gli host con destinazione `all nodes` al *IPv6 Solicited Node Multicast Address* e al corrispondente *MAC multicast address*, con il payload contenente l'indirizzo che si vuole utilizzare. Se non vi è nessuna risposta entro 1 secondo, l'indirizzo viene considerato valido, se invece vi è risposta significa che l'indirizzo è già utilizzato in quanto un host ha risposto con un messaggio *ICMPv6* di tipo **DAD** con il *payload* contenente l'indirizzo da utilizzare.

### 2.15.5 Fasi di una configurazione Stateless

La configurazione *stateless* di un nuovo dispositivo permette il non utilizzo del protocollo DHCP per la configurazione automatica dei dispositivi. Avviene nei seguenti passaggi:

- Generazione di un indirizzo link local.
- Verifica dell'unicità dell'indirizzo (**DAD**).
- il dispositivo si pone in ascolto di un messaggio di *router advertisement* o manda una *solicitation* per scoprire le informazioni sull'indirizzo privato.

Una volta scoperta la parte alta:

- si verifica se anche all'interno della sotto rete che l'indirizzo sia unico (di nuovo).
- iscrizione al corrispondente IPv6 Solicited Node Multicast Address, configurando la ricezione del multicast MAC corrispondente e inviando un ICP MULTicast Listener Report.
- La comunicazione on-link è abilitata.

Un vantaggio è quello del **renumbering**, che consente un funzionamento *plug and play*. Tramite l'*advertisement* vengono riconfigurati tutti i dispositivi in modo automatico. Questi rimangono in ascolto per il *Router Advertisement* e quando arriva un messaggio con un nuovo prefisso, cambiano indirizzo GI, in modo da poterli riconfigurare in qualsiasi momento. Si identificano così indirizzi “*preferred*” e “*deprecated*” permettendo di cambiare *ISP* senza aggiornare tutti gli indirizzi.

## 2.16 Scoped Addresses

Un dispositivo può avere più interfacce con il medesimo indirizzo, per cui un determinato pacchetto viene mandato su un interfaccia piuttosto che un'altra in base allo **scopo** e al programma che lo ha

generato (concetto di scopo). Un indirizzo scoped è composto da un indirizzo IPv6 seguito da % e un numero che identifica l'interfaccia.

Ad esempio: **FE80::0237:00FF:FE02:A7FD%19**

**Attenzione:** il valore dello scopo è specifico per ogni implementazione.

**Attenzione:** Questo byte di scope non viene più utilizzato perché è utile solo per il sistema operativo.

## 2.17 Routing Protocols

Il routing in due tesi differenzia in due tipologie:

- **On the fly routing:** è il *forwarding*, usa le *routing tables* e permette, a fronte di un pacchetto entrante in un nodo di rete, di determinare qual è la migliore porta di uscita verso la destinazione.
- **Proactive routing:** crea le *routing tables* in modo da individuare un percorso valido per un pacchetto, dal mittente al destinatario.

La creazione di tali tabelle di tipo manuale prende il nome di **static routing**, in caso contrario la distribuzione delle informazioni all'interno della rete è ottenuta attraverso protocolli di routing.

Le routing table in IPv6 sono basate sul **longest prefix match** (come in IPv4), ma nonostante alcune peculiarità, IPv4 e IPv6 si comportano come due protocolli indipendenti (con routing table separate).

I protocolli di routing possono essere:

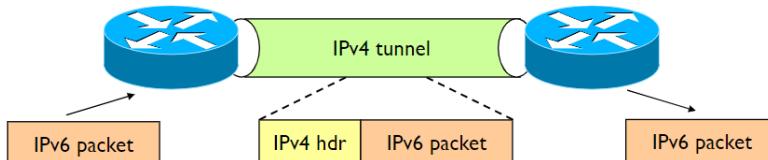
- **Integrated routing:** viene adoperato un singolo protocollo che informa i destinatari per entrambe le *protocol families* (sia IPv4 che IPv6). Ha come vantaggio quello di non avere meccanismi di duplicazione, ma è necessaria l'implementazione di un nuovo protocollo dedicato che potrebbe comportare bug con il funzionamento delle operazioni in IPv4. Inoltre, le topologie di rete tra IPv4 ed IPv6 potrebbero essere diverse e quindi il routing potrebbe non essere ottimale.
- **ships in the night:** ogni *family address* ha il suo protocollo di routing, con la caratteristica che tutti i protocolli sono indipendenti l'uno dall'altro. In questo modo è possibile utilizzare protocolli di routing differenti (scelti in base alla topologia o scenario). Il vantaggio è una più semplice integrazione e troubleshooting, ma comporta un inevitabile meccanismo di duplicazione.

*Esempi di routing protocol:*

Protocollo	Approccio
Static	Ships in the night
RIPng	Ships in the night
EIGRP	Ships in the night
OSPFv3	Ships in the night (Integrated routing is possible)
IS-IS	Integrated routing
MP-BGP	Both (configuration-dependent); “Integrated Routing” is the most commonly deployed because of practicality: BGP process identified by AS number, which is the same for both IPv4 and IPv6.

## 2.18 La transizione da IPv4 a IPv6

La transizione da *IPv4* e *IPv6*, come già detto, è tutt’ora in corso e molto lenta. In prima battuta, quando la maggior parte delle connessioni erano su *IPv4* si andava a utilizzare il tunneling di *IPv6*, il cui nome deriva dal fatto che *IPv6* veniva inserito in un header *IPv4* per compatibilità.



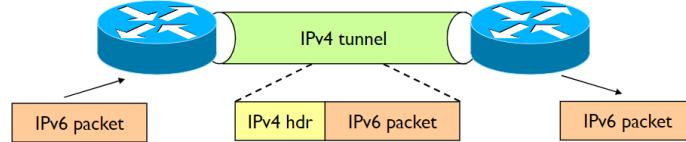
**Figura 2.35:** Esempio di Tunneling

L’approccio iniziale è stato di tipo **dual stack** con lo scopo di supportare le funzionalità di entrambi i protocolli, ma con la limitazione di non ridurre l’utilizzo di *IPv4* e di lasciare la responsabilità alle applicazioni di utilizzare *IPv6* o *IPv4*. Un aspetto però negativo era la completa duplicazione di tutto lo stack di protocolli necessari (routing protocols, routing table, access list).

Un differente approccio è quello di utilizzare il **tunneling**, ovvero incapsulare un pacchetto *IPv6* in uno *IPv4* al fine di emulare il link diretto tra dispositivi *IPv6* ma in una infrastruttura *IPv4*. Alcuni protocolli che implementano soluzioni di tipo tunneling sono:

- **GRE** (*Generic Routing Encapsulation*)
- **IPv6 in IPV4** (*protocollo di tipo 41*)

- setup manuale ed automatico



**Figura 2.36:** Tunneling

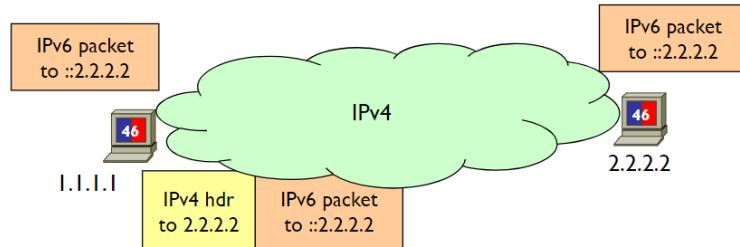
### 2.18.1 Host centered solutions

Una soluzione potrebbe essere di utilizzare un approccio di tipo *dual stack host*, ovvero un host che supporta sia *IPv4* che *IPv6*. In questo modo, il *tunneling* non è più necessario.

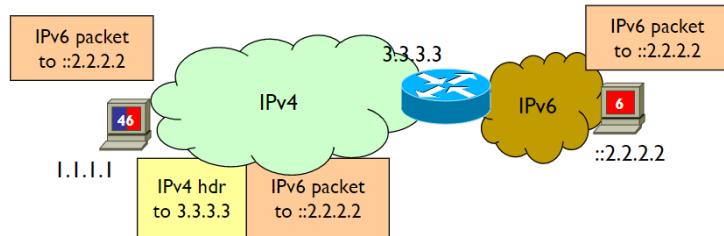
Per fare ciò, degli indirizzi *IPv6* devono essere riservati per la compatibilità con *IPv4*, in particolare quelli con il prefisso  $::/96$ , in modo da ignorare i bit più significativi e renderlo retrocompatibile.

Le applicazioni mandano pacchetti *IPv6* attraverso un indirizzo *IPv6*, ad esempio  $::2.2.2.2$  e vengono reindirizzati a  $::/96$  attraverso una *pseudo-interfaccia* (che fa tunneling automaticamente). Questa dunque incapsula i pacchetti *IPv6* in pacchetti *IPv4* e li invia.

**Nota:** devono essere riservati  $::/96$  perché consente di mantenere libero i 32 bit meno significativi per *IPv4*:  $128 - 96 = 32$ .



**Figura 2.37:** End-to-End-Tunneling



**Figura 2.38:** Dual stack router

### 2.18.1.1 6over4

Il protocollo **6over4** utilizza una rete *IPv4* per emulare una *virtual LAN*, attraverso il *broadcast multiple access data link* e l'*IP multicasting*.

Il *neighbor* e *router discovery* sono abilitati in modo da consentire l'individuazione di nodi e dei router vicini.

L'indirizzo *IPv4* è utilizzato per la generazione automatica di un *interface ID* *IPv6* dell'indirizzo *link local*.

**Nota:** Non è molto utilizzato a causa della poca diffusione del supporto *IPv4 multicast*.

### 2.18.1.2 ISATAP: Intra-site Automatic Tunnel Addressing Protocol

**ISATAP** si differenzia in quanto al posto di usare il multicast, utilizza una soluzione con un prefisso di rete **0000 : 5EFE**. La rete *IPv4* viene utilizzata come una *Non-Broadcast Multiple Access (NBMA)* data link, in questo modo non è necessario il supporto per IP multicast.

L'interface ID viene derivata dall'indirizzo *IPv4*.

Utilizza il protocollo *DNS*, ma ha come limitazione che ogni indirizzo deve avere associato un *hostname*. Quindi la richiesta non parte dall'indirizzo di *IPv6*, ma dal *hostname* (potrebbe essere un problema in alcuni casi).

Non è necessario eseguire *data-link address discovery* in quanto l'indirizzo *IPv4* è incluso nell'indirizzo *IPv6*, in particolare negli ultimi 4 byte.

Si rende necessario fornire una *PRL* (Potential Router List) in quanto la router discovery non è possibile. Può essere configurata manualmente oppure acquisita dal *DNS*.

### 2.18.1.3 Configurazione automatica

La configurazione automatica è diventata lo standard nel tempo. Vengono utilizzati indirizzi *IPv4*, indirizzi *DNS* e il nome del dominio viene ottenuto tramite *DHCPv4*.

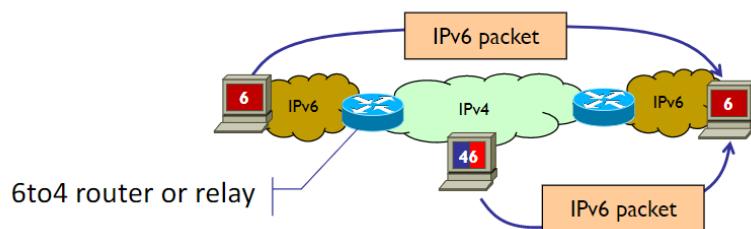
L'indirizzo *IPv6* link local viene generato automaticamente, mentre l'interface ID è ottenuto dall'indirizzo *IPv4*.

Per ottenere il *Potential Router List* si utilizza una query *DNS*, a meno che non sia fornita da *DHCPv4*.

Periodicamente viene eseguita una *router discovery* verso tutti i router su link prefixed per l'autoconfigurazione.

### 2.18.2 Network center solution

Si configurano intere reti *IPv6* all'interno di una struttura ancora *IPv4*, dovendo però rinunciare a parte delle funzionalità *IPv6*, inoltre il range di indirizzi continua a essere ridotto.



**Figura 2.39:** Network centered solution

#### 2.18.2.1 6to4

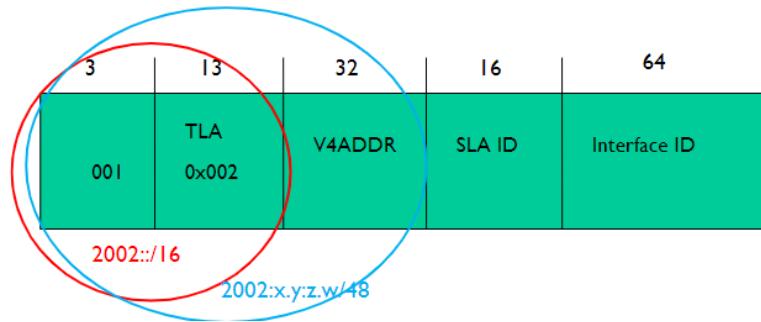
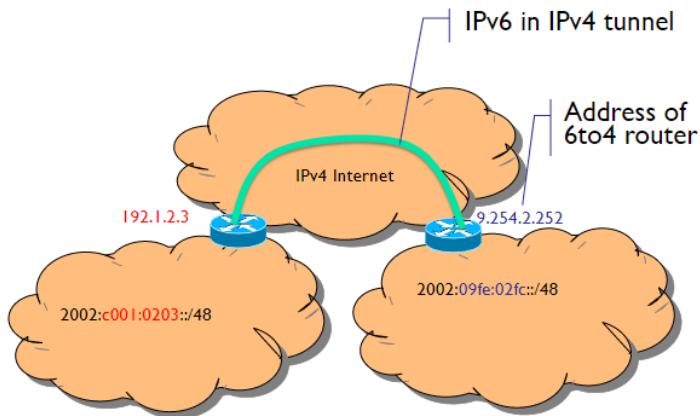
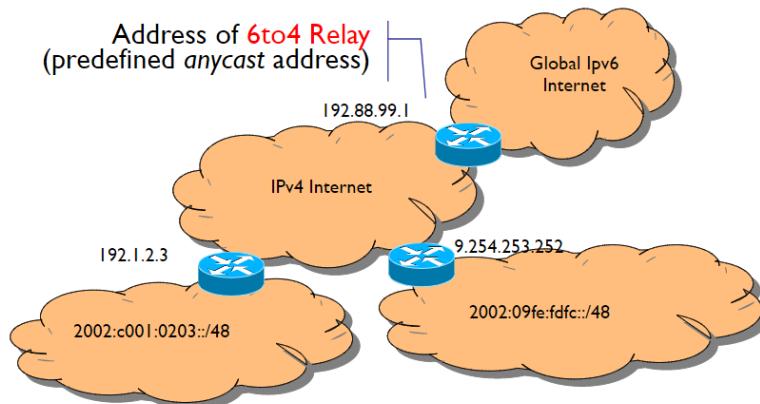
Attraverso il protocollo **6to4** gli indirizzi dei relay sono *embedded* in un prefisso *IPv6*. Iniziano con 2002 e sono indirizzi pubblici (inizia con 2).

Questo protocollo non è pensato per le comunicazioni da host *IPv4* a host *IPv6*.

Un relay 6to4 deve essere necessariamente il default gateway per i router 6to4.

#### 2.18.2.2 Tunnel broker

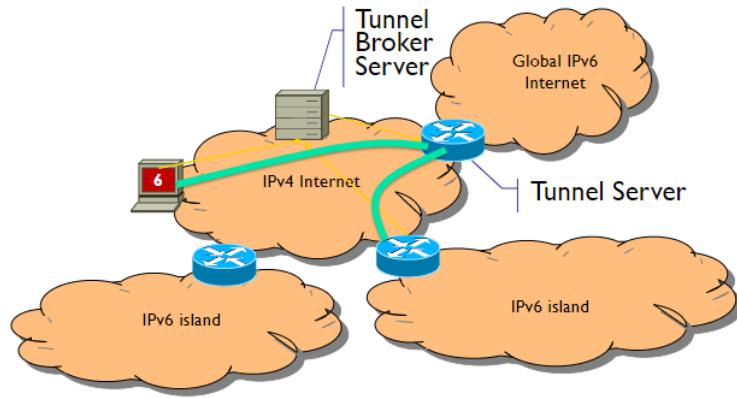
In questa modalità le comunicazioni avvengono attraverso un **tunnel broker server** che si occupa di individuare i *tunnel server* e fa da mediatore tra le configurazioni dei tunnel.

**Figura 2.40:** Schema indirizzo**Figura 2.41:** Scenario base**Figura 2.42:** Scenario misto

Vengono utilizzati tunnel IPv6 in IPv4 (a.k.a. proto-41).

Per eseguire la configurazione dei tunnel viene utilizzato il *Tunnel Setup Protocol* (TSP) o il *Tunnel Information Control* (TIC).

Questo tipo di soluzione è centralizzata.



**Figura 2.43:** Architettura tunnel broker

## 2.19 Scalable, Carrier-grade Solutions

Le soluzioni adottate dai grandi provider prevedono ancora il supporto per i server e i client IPv4 in modo che possano comunicare con host IPv6 e host IPv4. Le soluzioni più utilizzate sono:

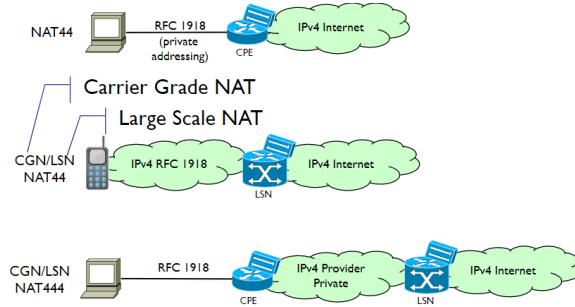
- **DS-Lite**
- **A+P (evoluzione di DS-Lite)**
- **MAP-T and MAP-E**
- **NAT64**
- **6PE (MPLS-based)**

Tutte queste soluzioni si basano sul concetto di *mapping* di un indirizzo IP, ovvero il NAT, eseguendo un mapping tra ipv4 e ipv4. Quello che viene fatto è associare una porta a un indirizzo privato.

Prende il nome di **LSN** il **Large Scale NAT**, utilizzato per gestire una grande quantità di richieste.

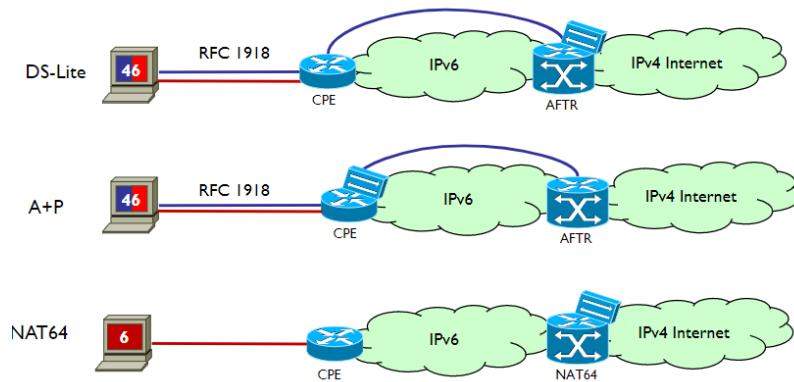
E' possibile avere più livelli di NAT ponendoli in cascata (*pratica piuttosto comune*).

Nonostante il grande utilizzo, non è da dimenticare il fatto che il NAT comporti un *single point of failure* e che a causa della scarsità di indirizzi possa essere difficile da implementare.



**Figura 2.44:** Nat è ampiamente utilizzato

E' necessario tenere a mente che nelle soluzioni proposte, anche se è previsto l'utilizzo del NAT, è comunque presente l'utilizzo di tunnel.



**Figura 2.45:** Stessa architettura con IPv6

E' importante notare la posizione del *NAT* nei protocolli esposti sopra:

- **DS-Lite:** il *NAT* è presente nel AFTR.
- **A+P:** il *NAT* è post sul CPE (*Customer Provided Equipment*).
- **NAT64:** il *NAT* è post sul *NAT64*.

**Attenzione:** notare dove le funzionalità di *NAT* sono presenti.

### 2.19.1 AFTR: Address Family Transition Router

L'utilizzo del **Address Family Transition Router** (AFTR) Abilita gli host *IPv4* a comunicare con altri host *IPv4* attraverso una rete *IPv6* (ad esempio la connessione residenziale fornita dagli attuali provider). Ha dunque come conseguenza il poter connettere strutture *IPv6* con una infrastruttura nel mezzo *IPv4*.

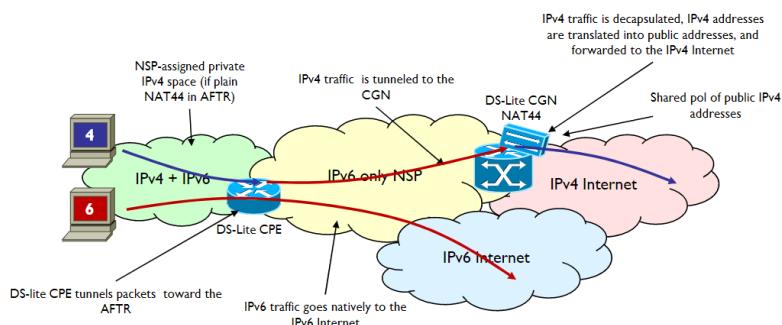
Ha due tipi di funzionalità:

- sia come *NAT*, in quanto gestisce richieste di *natting*.
- parte hardware che consentono le operazioni di tunneling.

**Nota:** Viene utilizzato da DS-Lite e A+P.

## 2.19.2 DS-Lite

La soluzione **Dual-Stack Lite** è caratterizzata da *Internet Service Provider* che utilizzano una backbone (infrastruttura di rete) /IPv6. Questo consente di avere solo parti /IPv4 o /IPv6 con altre sottoreti /IPv4 o /IPv6. Questa soluzione, rispetto a quelle già viste, sono molto articolate e consentono di coprire tutte le casistiche.



**Figura 2.46:** DS-Lite

Permette di ridurre il numero di indirizzi /IPv4 richiesti rispetto a un approccio *dual stack* (che aveva bisogno di un indirizzo pubblico per ogni host).

La soluzione adottata da *DS-Lite* funziona come segue: da /IPv4 privato si arriva sul *home gateway* (CPE) che effettua tunnel verso l'**AFTR** tramite una rete /IPv6 su cui è installato un *Large Scale NAT* (LSN), in questo modo un unico *NAT* gestisce tutti i clienti.

Ha però delle limitazioni:

- il cliente non ha controllo sul *NAT*.
- possono esserci problemi con i server in quanto *static mapping* e *port forwarding* non possono essere configurati.

Il *NAT esteso* consente l'indirizzamento assegnato dal cliente (ovvero sovrapposto).

Il NAT è posizionato sul AFTR.

### 2.19.3 A+P (Address plus Port)

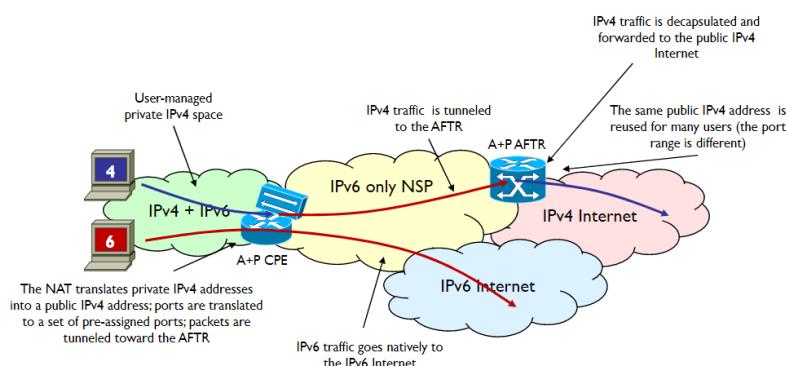
Il vantaggio di **A+P** risiede nella possibilità per il cliente di avere sotto controllo il *NAT*. Una ulteriore caratteristica è che il range di *TCP/UDP* è assegnato a ciascun customer (solo le porte sono utilizzate dal *NAT* in uscita).

La soluzione *A+P* parte da un indirizzo *IPv4* privato che arriva sul *CPE* e viene convertito in indirizzo pubblico e solo successivamente viene effettuato il tunnel *IPv6* verso l'*AFTR*. Quando il pacchetto esce dal tunnel è già stato trattato dal *NAT* e dunque può andare verso la rete pubblica.

Le features sono:

- nessun problema con la sovrapposizione degli indirizzi privati nello spazio di indirizzi dei customer.
- Le porte possono essere assegnate automaticamente al *CPE* utilizzando il Port Control Protocol (PCP), mentre il *CPE* può negoziare più porte in qualsiasi momento.
- AFTR è solo un tunnel terminator IPv4 in IPv6 (NAT44 non è più necessario in AFTR).

**Nota:** Il concetto alla base è di spostare la complessità sulle foglie.



**Figura 2.47:** A+P

### 2.19.4 Mapping Address and Port (MAP)

Il **Mapping Address and Port (MAP)** utilizza un approccio di tipo **stateless**. Questo sfrutta i vantaggi del *DHCP* e del *DNS* anche all'interno del sistema, non associando dei range di porte ma bensì dei **set**:

un set si differenzia dal fatto che ci sono più porte che non sono necessariamente contigue. Inoltre, il CPE utilizza la stessa rete pubblica *IPv4*, così non da non avere limitazioni.

La soluzione si basa sul cercare di inserire all'interno del pacchetto l'informazione di stato, in modo tale da diminuire l'informazione di stato che deve essere contenuta all'interno del Border Relay. Si cerca di mappare queste informazioni negli indirizzi IPv6 del CPE che vengono utilizzati. Così facendo il Border Relay può ricostruirsi le informazioni sul indirizzo IPv6 del CPE partendo dalle informazioni che ha (e che gli vengono fornite dal pacchetto IPv4 che arriva dalla rete internet).

L'indirizzo e la porta del client *IPv4* sono mappati in un unico indirizzo *IPv6* (prefix routed dal CPE).

L'indirizzo del server pubblico *IPv4* è anche questo mappato in un unico indirizzo *IPv6* (prefix routed dal Border Relay).

Esistono due tipi di *MAP*:

- **MAP-E:** *MAP with Encapsulation*, i pacchetti *IPv4* vengono tunnelizzati.
- **MAP-T:** *MAP with Translation*, i pacchetti *IPv4* sono tradotti in pacchetti *IPv6* e poi nuovamente in *IPv4*.

Quando però avviene la sostituzione di un header IPv6 con un header IPv4 è necessario fare attenzione a non perdere informazioni.

Si definisce **set** di porte un insieme di porte non sono necessariamente contigue.

## 2.19.5 Port Set

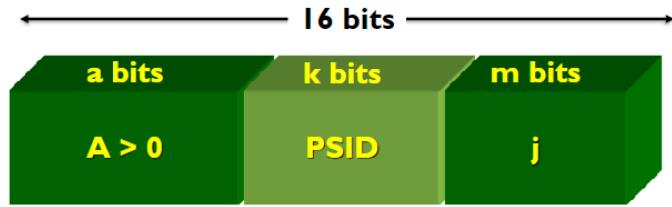
A ogni CPE viene assegnato un indirizzo pubblico *IPv4* un unico **PSID** (Port set Identifier) che identifica un set di porte.

Per creare un set di porte si utilizzano 16 bit, suddivisi in:

- *A*: identifica il dominio (ad esempio si assegna a un CPE le porte con i bit che iniziano con 1100).
- *PSID*: identifica il set di porte (lunghezza variabile).
- *j*: insieme delle porte.

**attenzione:** non porre i primi a bit a zero perché sennò diventa una *well known port*.

L'*embedded Address* (EA) contiene i bit di PSID e parzialmente l'indirizzo *IPv4* (che identificano univocamente il CPE).



**Figura 2.48:** Port set

### 2.19.6 Mapping Rule

Le regole per il mapping sono:

- IPv6 prefix rule
- IPv4 prefix rule: prefisso IPv4 di un indirizzo IPv4 utilizzato da un determinato CPE.
- EA bits length

Inoltre, un offset PSID (valore di a) viene settato per l'intero dominio di mappatura.

Tramite queste informazioni il CPE si calcola l'indirizzo IPv6 da utilizzare nell'interfaccia esterna.

### 2.19.7 Border Relay

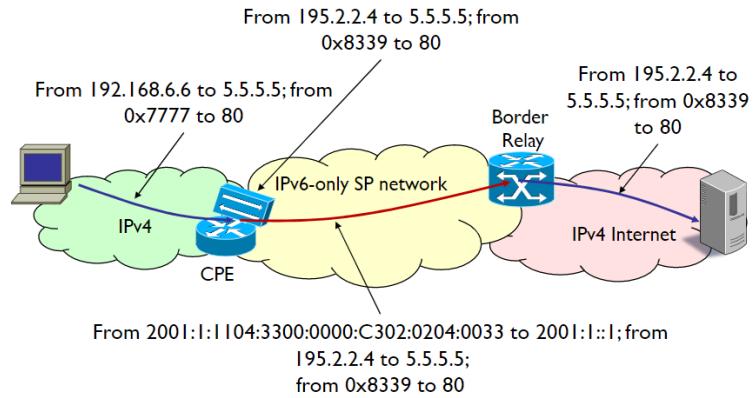
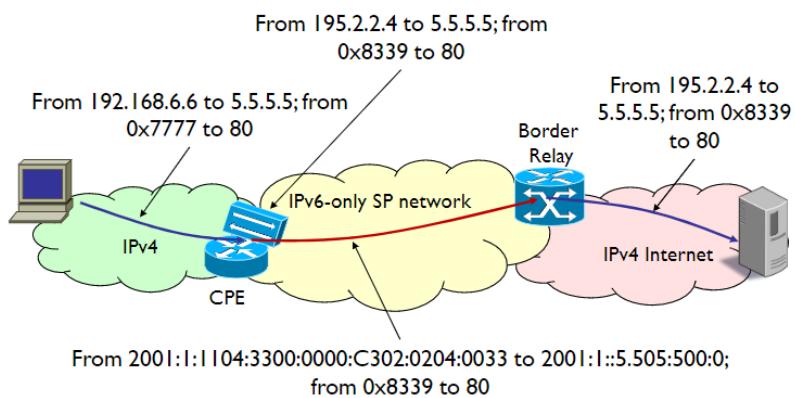
L'indirizzo del *border relay* deve essere conosciuto da tutti i *CPE*, anche se più *BR* possono avere lo stesso indirizzo (anycasting).

Mentre nel *MAP-E* il *BR* termina il tunnel, nel *MAP-T* il *BR* è responsabile della traduzione degli indirizzi *IPv4* verso l'esterno in quanto si occupa di sostituire l'header *IPv4* con un header *IPv6*. Il *BR* prefix viene advertised sul backbone (e potrebbe essere advertised da più *BR*).

**Non è vero** che viene sostituito l'indirizzo *IPv4* con un indirizzo *IPv6*, ma viene sostituito l'header *IPv4* con un header *IPv6*.

## 2.20 NAT64 + DNS64

Il **NAT64** è un meccanismo di transizione a *IPv6* che facilita la comunicazione tra *IPv4* ed *IPv6* utilizzando il *Network Address Translation* (NAT), che traduce indirizzi e pacchetti *IPv6* in *IPv4*, prendendo un indirizzo/porta *IPv4* liberi dal *pool* e realizzando un *NAT session entry*.

**Figura 2.49:** Vita di un pacchetto con MAP-E**Figura 2.50:** Vita di un pacchetto con MAP-T

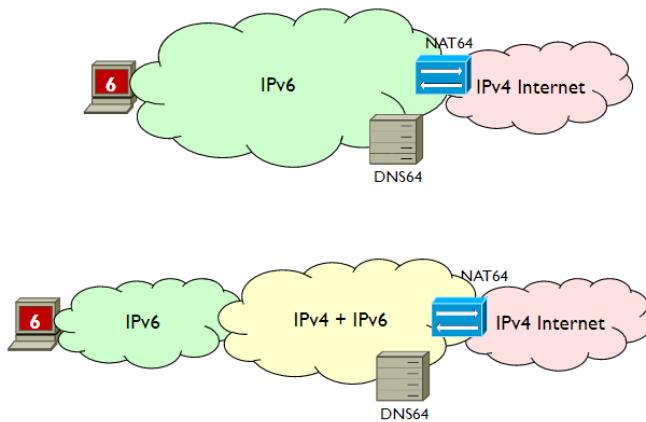
Questa tecnica risolve il problema della **rete IPv6 con host IPv6** che deve comunicare con la rete pubblica *IPv4*. In questo caso, la soluzione è il *NAT64* che rimuove l'header *IPv6* per inserire l'header *IPv4* ma per farlo viene utilizzato il *DNS64*.

Il vantaggio del *MAP* risiede nella possibilità di avere più *CPE* e maggiormente distribuite. Questa modalità rappresenta una forma semplificata, che può vedere il suo utilizzo su reti più piccole.

Un prefisso *IPv4* è dedicato per mappare indirizzi *IPv4*, comprensivi di *well-known* che di *network specific*. Il *DNS64* mappa un record in *AAAA* utilizzando un prefix *NAT64*, entrambi vengono poi forniti al client. Il router *NAT64* fa il *advertising* del prefisso in una rete *IPv6* per attirare il traffico verso gli host *IPv4*.

Il processo si divide nei seguenti passi:

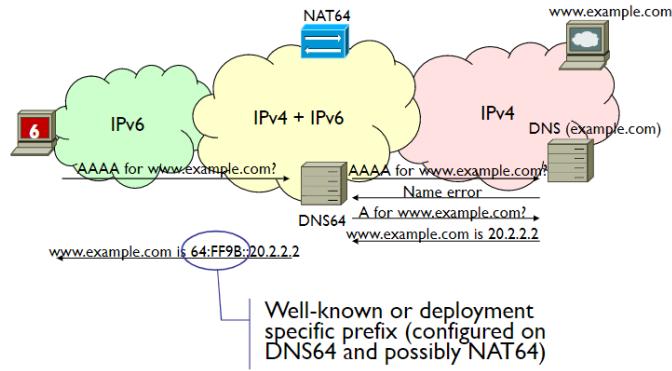
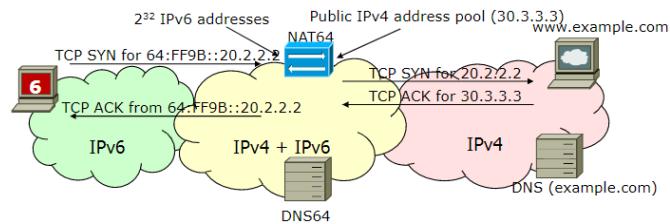
1. Un host *IPv6* esegue una query DNS di tipo *IPv6 AAAA*.
2. *DNS64* inoltra la query dal DNS autoritativo verso il dominio della richiesta.
3. Il DNS autoritativo (che si trova in una rete *IPv4*) non può ricevere la richiesta, per questo motivo viene ripetuta per un indirizzo *IPv4* (necessaria rete *IPv4+IPv6* intermedia).
4. *DNS64* riceve l'indirizzo *IPv4* che risolve il dominio, ne esegue l'*embedding* in un indirizzo *IPv6* con un prefisso di default 64 : **FF9B**.
5. L'indirizzo *IPv6* viene restituito al host che ha fatto la richiesta.



**Figura 2.51:** Deployment scenarios

Le limitazioni dovute al *NAT64 + DNS64* sono:

- Coinvolgimento del DNS (necessità del hostname)
- Non è possibile abilitare *DNSSEC* (ovvero la firma della risposta a un record DNS) perché DNS64 modifica i record.

**Figura 2.52:** Name resolution**Figura 2.53:** Packet forwarding

E' possibile risolvere indirizzi soltanto quando a questi sono associati dei nomi, dunque funzionerebbe con [www.example.com](http://www.example.com) ma non con [1.2.3.4](http://1.2.3.4).

# 3 Reti wireless e cellulari

## 3.1 Introduzione

Le reti **wireless** permettono la comunicazione tra dispositivi senza la necessità di un cavo fisico. Queste sono molto comuni oggi giorno, e sono presenti in molti dispositivi come cellulari, tablet, computer portatili, router, dispositivi di rete e molti altri. Un aspetto molto importante che ne deriva è la **mobilità**, anche se una parte rilevante di ogni rete wireless è in realtà la sua componente wired (oltre al wireless link).

I componenti principali delle reti wireless sono:

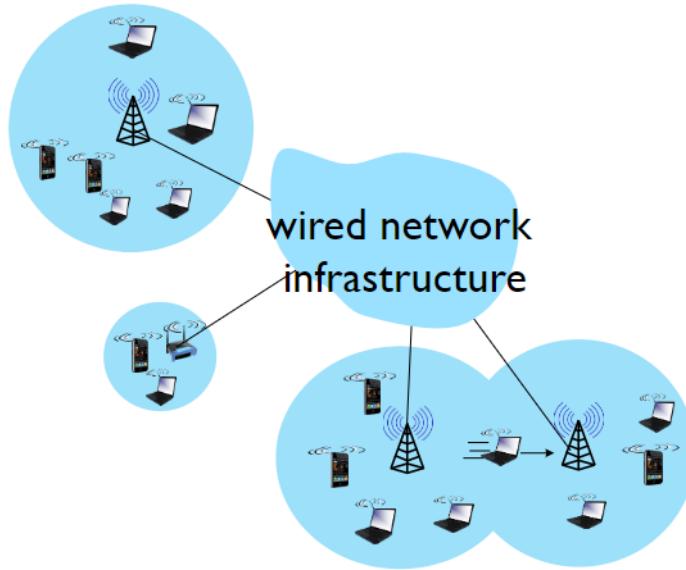
- **Wireless host:** dispositivi che possono trasmettere e ricevere dati, possono essere stazionari o mobili.
- **Base station:** responsabili di inviare pacchetti tra la rete cablata e quella wireless nella propria aria. Tipicamente sono connessi alla rete via cavo (esempio torri di telefonia mobile e access point).
- **Wireless link:** collegamento tra host e base station, tipicamente utilizzato per connettere un wireless host a una base station oppure come *backbone link*. L'accesso è controllato da protocolli ad accesso coordinato multiplo.

La rete cellulare può essere gestita mediante una infrastruttura attraverso cui le base stations connettono i wireless host alla rete cablata, con l'uso del **handoff** i wireless host cambiano la base station che fornisce una connessione alla rete cablata, oppure **ad hoc** in cui i wireless host si connettono direttamente tra di loro senza l'utilizzo di base station e organizzandosi autonomamente in una rete (i nodi trasmettono agli altri con un link coverage).

Nonostante i grandi vantaggi, il link wireless comportano alcuni svantaggi rispetto a un link cablato:

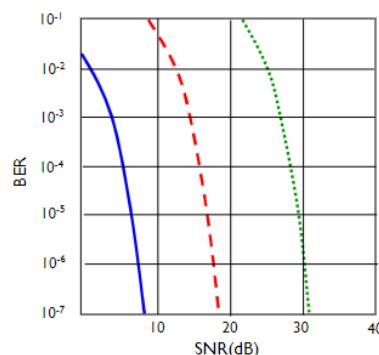
- Maggiore **degrado** del segnale dovuto all'attenuazione del segnale.
- **Interferenza** tra i dispositivi a causa dell'utilizzo delle stesse frequenze.
- **Multipath propagation** (fading): effetto dovuto ai rimbalzi del segnale sugli ostacoli.
- le **comunicazioni** tra punti diventa più **complicata**.

Un'altra importante caratteristica è il **Signal to Noise Ratio** (SNR), che esprime la relazione tra il segnale ricevuto e il rumore ed indica la qualità del segnale, più è alto più è semplice estrarre il segnale



**Figura 3.1:** Elementi di una rete wireless

dal rumore. Dato un livello fisico, aumentarne l'alimentazione comporta un aumento di SNR e una riduzione del *Bit Error Ratio* (BER), mentre dato un SNR è necessario scegliere un livello fisico che rispetta i requisiti di BER in modo da ottenere il massimo throughput. Il valore di SNR può cambiare a causa della mobilità, adattandosi dinamicamente al livello fisico.



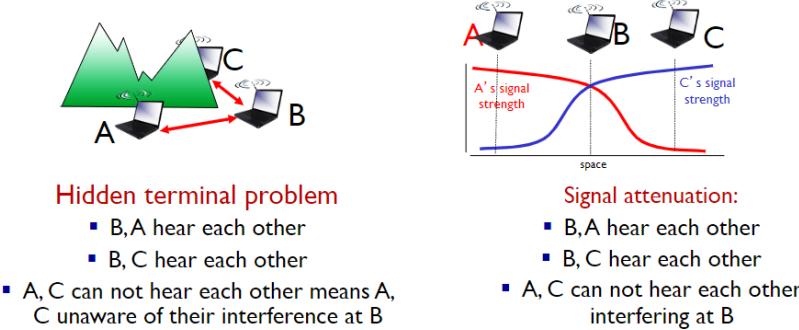
**Figura 3.2:** SNR e BER

La modulazione è il processo attraverso cui viene inviato un bit. Vi sono varie tipologie come:

- quam256
- quam16
- bpsk

Un problema che ritroviamo all'interno delle reti wireless è inherente al problema del **nodo (o termi-**

**nale) nascosto:** dati 3 nodi **a**, **b**, **c** se **b** comunica con entrambi i rimanenti, questi potrebbero però non essere a conoscenza della reciproca presenza e generare interferenze.



**Figura 3.3:** Problema del nodo nascosto

Inoltre le reti possono essere suddivise gerarchicamente, in particolare Una rete di primo livello (*Tier 1* network in inglese) è una rete IP (tipicamente ma non necessariamente un *Internet Service Provider*) che si connette all'intera Internet soltanto attraverso un'interconnessione **non** regolata da contratto, conosciuta anche come peering (dunque non a pagamento).

## 3.2 Wireless LAN

Nel corso degli anni lo standard 802.11 si è evoluto dando origine a vari standard, i quali utilizzano il protocollo *Carrier Sense Multiple Access*, **CSMA/CA**.

IEEE 802.11 standard	Year	Max data rate	Range	Frequency
802.11b	1999	11 Mbps	30m	2.4 Ghz
802.11g	2003	54 Mbps	30m	2.4 Ghz
802.11n (WiFi 4)	2009	600	70m	2.4, 5 Ghz
802.11ac (WiFi 5)	2013	3.47Gbps	70m	5 Ghz
802.11ax (WiFi 6)	2021	14 Gbps	70m	2.4, 5 Ghz
802.11af	2014	35 – 560 Mbps	1 Km	unused TV bands (54-790 MHz)
802.11ah	2017	347Mbps	1 Km	900 Mhz

Nelle reti wireless un wireless host comunica con una base station, ovvero un *access point* (AP).

Un **BSS** (*Basic Service Set*), ovvero una cella, se in modalità infrastruttura contiene un host wireless e una base station, mentre in modalità ad hoc solamente l'host.

Ogni rete wifi lavora su un canale differente ed è in grado di gestire fino a 16 frequenze (di cui utilizza solo una alla volta) per la trasmissione dei dati, con la possibilità che ci sia interferenza se il canale viene scelto male. La configurazione può essere automatica o manuale.

Ogni host che vuole connettersi esegue prima una scansione delle reti e rimane poi in attesa di un **beacon frame**: un frame speciale inviato dagli access point per effettuare la connessione contenente il nome dell'access point (SSID) e il MAC address. Il dispositivo si conterà al beacon frame più forte in modo da aumentare la qualità della connessione. Per poter iniziare a dialogare con la rete wifi potrebbe essere richiesta una autenticazione, a cui segue tipicamente una richiesta DHCP per recuperare l'indirizzo IP nella subnet del AP.

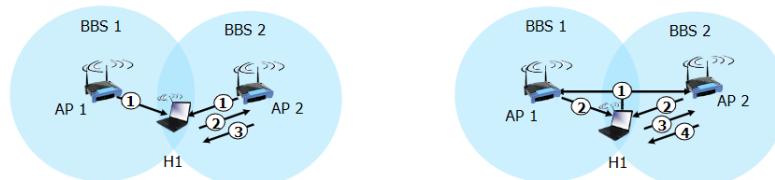
Esistono due tipologie di scanning eseguite da un host che si connette a una rete:

- **Passive scanning:**

1. il beacon frame viene inviato dagli access point all'host.
2. l'host manda una richiesta di associazione all'access point scelto.
3. l'access point conferma l'associazione mediante un *association response* verso l'host.

- **Active scanning:** l'host richiede il beacon frame all'access point, in 4 fasi che si dividono in:

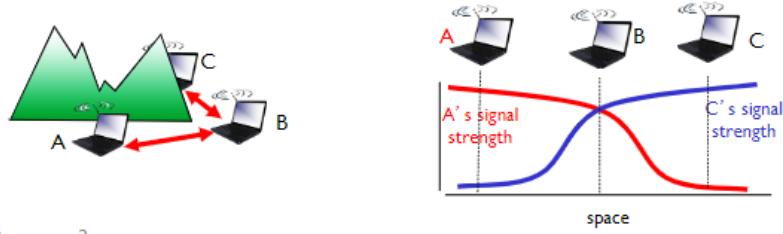
1. **probe request** dal host.
2. **probe response** dagli APs.
3. **association request** dal host verso l'access point scelto.
4. **association response** dal APs in questione.



**Figura 3.4:** A sinistra passive scanning e a destra active scanning

### 3.2.1 CSMA/CA

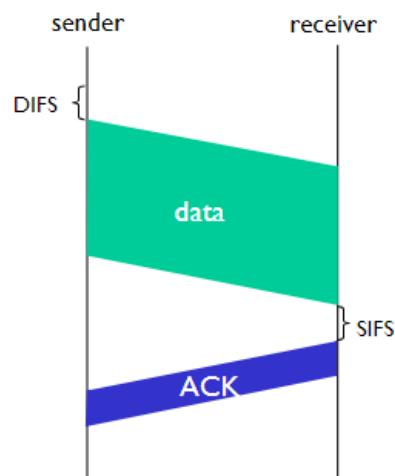
L'accesso di multipli dispositivi su un canale wireless è un problema molto complesso, che prevede l'utilizzo di **CSMA** per l'eliminazione delle collisioni tra due o più nodi che trasmettono contemporaneamente.

**Figura 3.5:** Accessi multipli

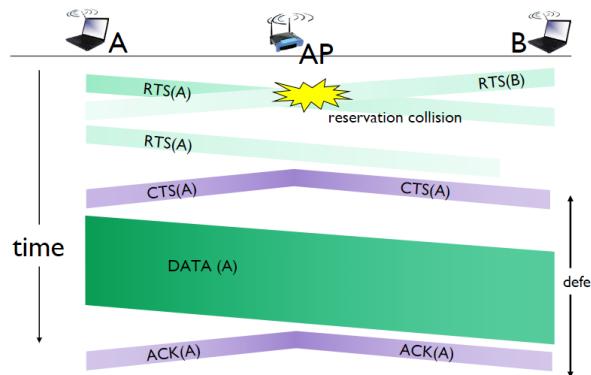
Mentre in ethernet viene utilizzato **CSMA/CD** (collision detection), in wireless viene utilizzato **CSMA/CA** (collision avoidance) con lo scopo di eseguire *sense before transmitting*, in modo di evitare le collisioni con la trasmissione già in corso di altri nodi.

Il funzionamento è il seguente:

- Il dispositivo che invia:
  1. Se il canale è in idle per **DIFS** tempo, allora il dispositivo inizia a trasmettere (no CD).
  2. Se il canale è occupato, viene avviato un *random backoff time* che lo pone in attesa prima del nuovo tentativo. Se anche al nuovo tentativo il canale è occupato, il dispositivo ripete il processo aumentando il *random backoff interval*.
- Il dispositivo che riceve:
  1. Se il frame è ricevuto correttamente, viene inviato un ACK frame dopo **SIFS** tempo (necessario per evitare il problema del terminale nascosto).

**Figura 3.6:** Schema di funzionamento

Il *Collision Avoidance* mostrato sopra non è però deterministico, per riuscire a renderlo tale è possibile utilizzare un sistema di “prenotazione” che riserva il canale per i data frame usando dei pacchetti di “prenotazione” (RTS/CTS) caratterizzati da trame piccole. Questi possono ancora collidere, ma sono molto più piccoli e quindi meno dannosi. Il pacchetto **RTS** (ready to send) viene inviato dal dispositivo che vuole trasmettere, mentre **CTS** (clear to send) viene inviato dal dispositivo che ha ricevuto il pacchetto RTS verso tutti i dispositivi in ascolto in modo da far partire la trasmissione da chi deve trasmettere e porre in attesa i rimanenti.



**Figura 3.7:** Schema temporale RTS-CTS

### 3.2.1.1 Frame addressing

Il frame contiene:

- frame control
- duration
- address 1: mac address del host wireless o Access Point che deve ricevere il frame
- address 2: MAC address del host wireless o Access Point che deve trasmettere il frame
- address 3: MAC address dell’interfaccia del router a cui l’access point è connesso
- seq control: necessari per gli ack
- address 4: usato solo in modalità ad hoc
- payload
- crc: controllo di errore

Dentro frame control troviamo ulteriori campi, tra cui ad esempio:

- protocol version
- tipo (RTS, CTS, ACK, data)
- sottotipo
- bit per il power management

### 3.2.1.2 Mobilità nella stessa sottorete

Soltamente per le reti wireless l'host rimane all'interno della stessa *subnet IP*, motivo per cui è possibile riutilizzare lo stesso indirizzo.

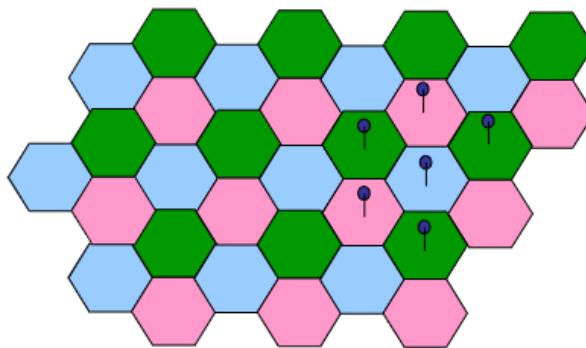
Spesso gli switch sono *self learning*, ovvero quando vedono un frame transitare verso un host ne memorizzano la porta a cui è stato inviato.

Dal punto di vista energetico, esiste il **node-to-AP** attraverso il quale l'Access Point viene a conoscenza del fatto che non deve inoltrare i frame al nodo, il quale si sveglierà prima del prossimo beacon frame (ha al suo interno la lista dei dispositivo con gli AP-to-mobile frames in attesa di essere inviati).

## 3.3 Reti cellulari

Le **reti cellulari** sono reti wireless che coprono aree geografiche molto vaste attraverso la definizione di zone adiacenti denominate **celle**. A differenza di altre reti, gli host si muovono anche attraverso lunghe distanza e diventa importante non far disconnettere l'utente attraverso la gestione della mobilità denominata **handover**.

La copertura cellulare è garantita mediante reti *isotopiche* e antenne direzionali da 120 gradi e posizionate a un estremo della cella. La forma non è esattamente esagonale e l'emissione non è *omni direzionale* a causa della presenza di ostacoli (montagne, edifici), altezza, il guadagno dell'antenna, la morfologia del territorio, la potenza dell'antenna e infine le condizioni di propagazione (atmosferici ecc...).



**Figura 3.8:** Copertura cellulare

Le celle si dividono in **macrocelle** e **microcelle** in base alle loro dimensioni e di conseguenza alla copertura.

Come nelle reti wireless, è nuovamente presente il problema di accesso multiplo condiviso sul canale, che viene risolto attraverso varie tecniche:

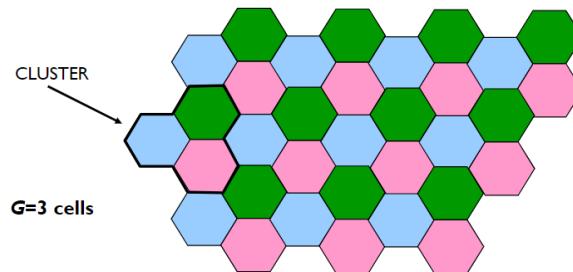
- **FDMA**: viene scelto una frequenza in cui trasmettere.
- **TDMA**: viene scelto uno slot temporale in cui trasmettere.
- **CDMA**: viene assegnato a ogni stazione un codice *ortogonale* agli altri, ovvero un gruppo di segnali da cui è possibile recuperare ogni singolo segnale.
- **SDMA**: ogni frequenza viene riutilizzata, a condizione che i luoghi siano fisicamente molto distanti tra loro.

Verranno riutilizzate le stesse frequenze in posti diversi in modo da non causare interferenze. Questo viene fatto a causa del ridotto numero di risorse, nel tentativo di coprire un'area più ampia e servire un maggior numero di utenti.

**Definizione:** Si definisce **handover** la gestione della mobilità di un dispositivo su una rete cellulare e il conseguente funzionamento di sgancio e riaggancio tra le celle. E' valido **se e solo se** la comunicazione è **attiva**.

### 3.3.1 Cluster

Un gruppo di celle viene definito **cluster**, come nell'esempio in figura.



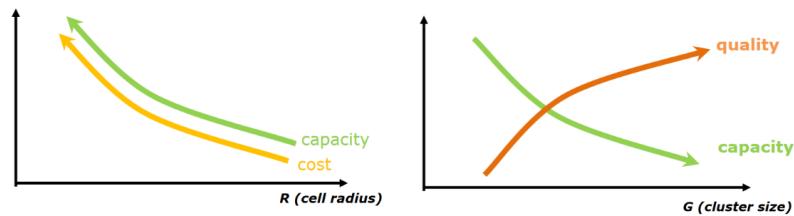
**Figura 3.9:** 3-Cell Cluster

Le celle *verdi*, *rosa* e *blu* usano un set differente di canali. Le celle dello stesso colore sono chiamate **“co-channel” cells**.

Con la variazione della dimensione delle celle *R* cambia la capacità, ovvero il numero di utenti che questa è in grado di soddisfare. Il numero di celle *G* impatta invece sul costo, in quanto un numero maggiore di celle ha dei costi maggiori. Aumentando il cluster aumenta la qualità, aumentando anche *G* aumenta la qualità ma diminuisce la capacità.

Fissando  $G$ , se si diminuisce il raggio  $R$  delle celle si guadagna in capacità ma aumentano anche i costi. Se si diminuisce il raggio, oltre che i costi possono aumentare anche le interferenze perché le *co-channel cells* saranno molto più vicine.

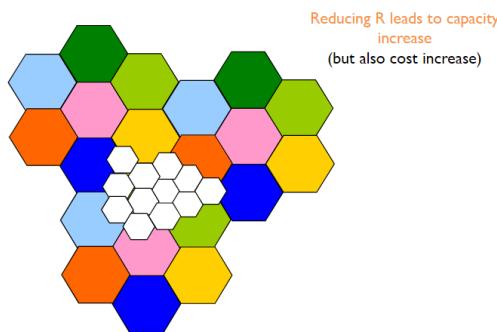
Fissando  $R$  e facendo variare  $G$  si nota che, aumentando il numero  $G$  del cluster diminuisce il numero di canali per cella e dunque la capacità decresce ma aumenta la distanza tra le co-channel cells e dunque le interferenze sono minori comportando un aumento di qualità.



**Figura 3.10:** A sinistra fissando  $G$ , a destra fissando  $R$

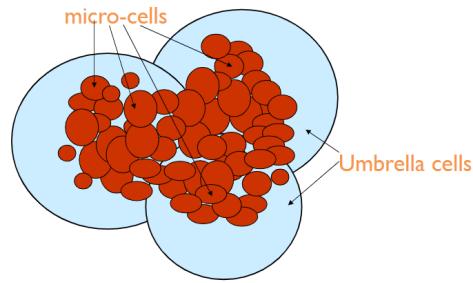
Non esiste una legge assoluta per definire i due parametri, ma è possibile sfruttare alcune tecniche per diminuire le interferenze ed aumentare la capacità:

- **splitting:** non utilizzare celle delle stesse dimensioni, ma basarsi sulle necessità specifiche.
- **sectoring:** utilizzare delle antenne non omnidirezionali per ridurre le interferenze e ridurre solo nelle direzioni in cui non è necessario.
- **tilting:** non usare un angolo a 90 gradi per la trasmissione cercando di variare l'angolo).
- **Creazione di femtocelle:** possiamo creare delle celle non fisse in base alle necessità (esempio stadio o concerti).



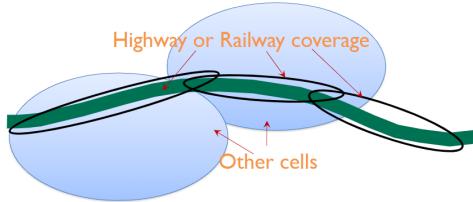
**Figura 3.11:** Splitting

Inoltre è possibile utilizzare antenne direzionali per avere celle con dimensioni e forme ad-hoc, oppure adoperare una copertura multi livello (umbrella coverage) o infine utilizzare microcelle che seguano l'utente dove si muove.



**Figura 3.12:** Shaping

Altri esempi sono possibile tenendo conto di strade oppure ferrovie, dove le celle cercano di seguire la forma della strada.



**Figura 3.13:** Shaping su strade

### 3.3.2 Power Control

Il **Power Control** mira al gestire al meglio le capacità delle batterie con l'obiettivo è di ridurre l'utilizzo di potenza in base alle necessità e in relazione alla qualità di trasmissione e alla distanza tra cellulare e antenna. Per effettuare la regolazione della potenza necessaria si utilizzano strategie di due tipi:

- **a catena aperta** (*open loop*): sistema senza reazione
- **a catena chiusa** (*closed loop*): sistema con reazione (*feedback*)

In particolare in *uplink* (da terminale a ripetitore) si utilizzano le seguenti strategie:

- *closed loop power control*
- *open loop power control*
- *outer loop power control*

Mentre in *downlink* (da ripetitore a terminale) si utilizza:

- Downlink power control

### 3.3.2.1 Open loop

Nella strategia **open loop** per l'*uplink* il sistema, non avendo a disposizione un feedback, analizza e misura la qualità del segnale ricevuto (da ripetitore a terminale, downlink) per valutare se aumentare o diminuire la potenza di trasmissione. Questo adattamento non è preciso e non è detto che ciò che succede su una frequenza sia uguale a un'altra. Non è molto accurato in quanto solitamente uplink e downlink trasmettono su canali differenti, dunque la qualità potrebbe essere differente.

Soltamente si divide in due fasi:

- l'utente misura la qualità del segnale che riceve dalla base station.
- l'utente utilizza poi un algoritmo per impostare la potenza di trasmissione in modo che la SINR (*Signal-to-interference-plus-noise ratio*) sia sopra una certa soglia.

In questa modalità il terminale “*si regola autonomamente*” sulla potenza di trasmissione.

### 3.3.3 Allocazione della frequenza

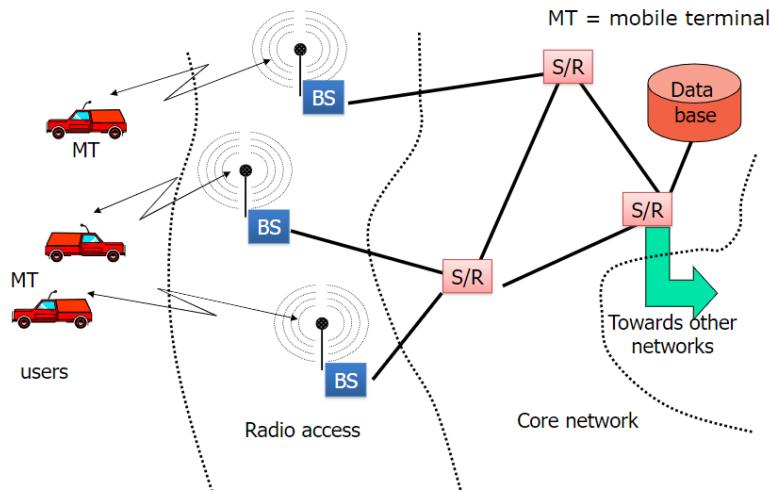
L'allocazione delle frequenze possono avvenire nei seguenti modi:

- **Fixed Channel Allocation (FCA)**: Basato sul concetto di cluster, le frequenze sono assegnate staticamente e vengono modificate raramente per aumentare performance e adattare piccole variazioni sull'utilizzo del traffico dell'utente.
- **Dynamic Channel Allocation (DCA)**: Le risorse sono assegnate da un controller centrale, quando necessarie. Il frequency plan varia nel tempo in modo da adattarsi allo stato del sistema.
- **Hybrid Channel allocation Scheme (HCS)**: Una porzione è allocata staticamente (FCA) mentre una dinamicamente (DCA)

### 3.3.4 Architettura di rete

Le reti sono costituite da *mobile terminal* (MT) che si connettono a delle *base station* (BS) radio che a loro volta si connettono a dei core network attraverso Switch Router (commutatori a pacchetto o circuito). I core network sono costituiti da un set di server che si occupano di gestire le connessioni e le risorse, in modalità cablata (*wired*). Il *database* viene utilizzato per memorizzare le informazioni degli utenti.

Il processo di **registrazione** permette a un terminale mobile di connettersi alla rete attraverso una registrazione che lo identifica e autentica. La procedura avviene periodicamente ogni volta che si deve accedere al servizio, oppure quando il terminale si accende e deve associarsi alla rete.



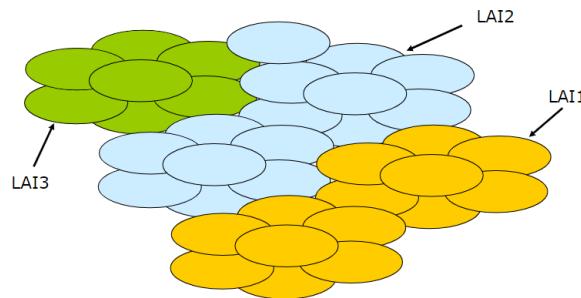
**Figura 3.14:** Architettura di rete

Un'altra procedura è quella del **Mobility Management**, utilizzata per gestire la mobilità e che a sua volta utilizza le seguenti procedure:

- Roaming
- Location updating
- Paging
- Handover

### 3.3.4.1 Roaming

Il **roaming** è la capacità di un terminale di essere tracciabile quando si sposta nella rete. Il sistema deve memorizzare la posizione in un database e localizzare l'utente quando necessario. Per salvare tali informazioni, la rete viene divisa in location areas (LAs), ovvero gruppi di celle adiacenti ciascuna con un identificativo univoco.



**Figura 3.15:** Roaming

### 3.3.4.2 Location updating

Il location updating è la procedura che avviene ogni volta che un utente si sposta verso un'altra location area.

Periodicamente l'utente deve comunicare la sua posizione alla rete, in modo da essere tracciato. Questa procedura è necessaria per mantenere aggiornate le informazioni sul database.

### 3.3.4.3 Paging

Il **Paging** è la procedura attraverso la quale il sistema notifica un terminale mobile di una chiamata o data delivery.

Il sistema manda la richiesta in broadcast a tutti i terminali della location area, e il terminale che riceve la richiesta risponde con un messaggio di conferma.

### 3.3.4.4 Handover

La procedura di **Handover** abilita il trasferimento di una **connessione attiva** da una cella verso un'altra, mentre il terminale mobile si sposta nella rete. Questa procedura è molto complessa e richiede una rete ben architettata, con protocolli e segnali adeguati.

Si classifica nei seguenti tipi:

- **Intra vs. Inter Cell:** Indica se l'handover avviene tra frequenze all'interno della stessa cella o di celle diverse.
- **Soft vs. Hard:** Indica se durante l'handover sono attivi entrambi i canali radio (soft) o solo uno alla volta è attivo (hard).
- **MT vs. BS initiated:** Indica se il primo messaggio di controllo per l'avvio di un handover è inviato dal terminale mobile (MT initiated) o dalla BS (BS initiated), ovvero quale entità esegue le misure per capire dove e quando deve essere eseguito un handover.
- **Backward vs. Forward:** Indica se la segnalazione di handover avviene tramite la BS di origine (backward) o la BS di destinazione (forwarding).

La connessione **deve essere attiva!**

### 3.4 Evoluzione della rete cellulare

Nel corso degli ultimi anni la rete cellulare ha subito una serie di evoluzioni che hanno portato ad una maggiore capacità di trasmissione e ad una maggiore efficienza energetica.

La prima generazione **GSM** era di tipo analogico, con ampio utilizzo di FDMA e trasportava traffico esclusivamente voce. La qualità del segnale era bassa e l'efficienza nel riutilizzo della frequenza era scarsa.

La seconda generazione ha comportato il passaggio al digitale, con il vantaggio in termini di servizi (sms), crittografia e voice coding avanzato per ridurre la banda necessaria. La seconda generazione estesa, **2.5G**, caratterizzata da **GPRS/EDGE** in europa e IS-95B in USA, vede l'introduzione del servizio dati con packet switched, 170kb/s in GPRS e 384kb/s in EDGE. Si ha il passaggio a tariffe basate sul traffico e non più sul tempo.

La terza generazione, **3G**, ha comportato dei miglioramenti in termini di data service (multimedia service), l'introduzione di CDMA e l'avvento di UMTS e CDMA2000. Il rate dati ha raggiunto i 2Mb/s ed possibile l'handover tra reti differenti oltre alla exploit spatial diversity. La generazione **3.5G** ha comportato una evoluzione di **UMTS** soprattutto sul livello fisico, con miglioramenti del trasferimento dati fino a 56Mb/s in download e 22Mb/s in upload.

La quarta generazione, conosciuta come **LTE**, ha raggiunto un rate di 250Mb/s. Utilizza MIMO (multiple input multiple output) che consentono performance di modulazione più elevate. Per la prima volta abbiamo una rete completamente IP con l'introduzione di VoLTE per consentire il passaggio della voce sulla rete dati.

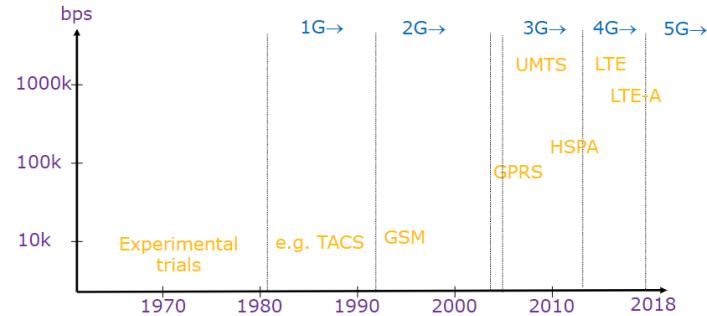
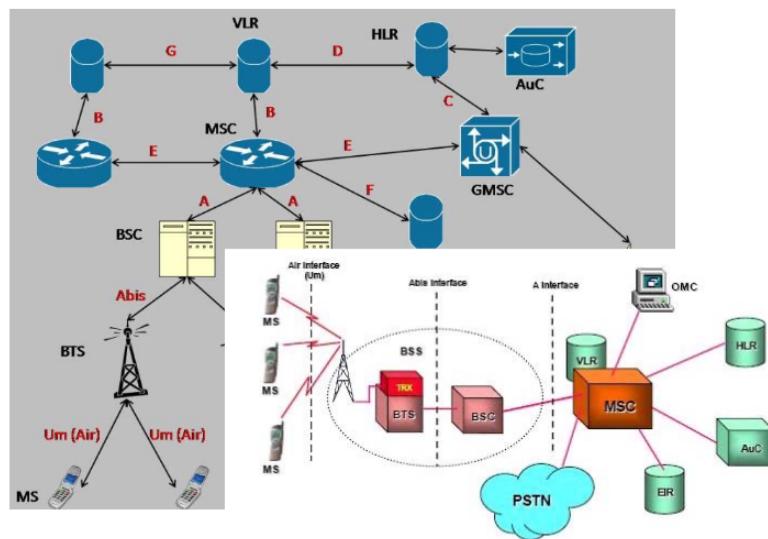
La quinta generazione, il **5G**, ha lo scopo di unificare le tecnologie di accesso wireless rimuovendo la differenza tra rete wireless e cellulare, attraverso mmWave che consentono trasmissioni ad alto throughput. Introduce il **NFV** (network function virtualization) che permette di virtualizzare le funzioni di rete, come il routing, il firewall, il load balancing, il caching, il DPI (deep packet inspection) e il DDoS (distributed denial of service) protection. Inoltre, anche il **SDN** (software defined networking) permette di virtualizzare il controllo della rete consentendo di utilizzare un hardware general purpose.

#### 3.4.1 GSM - Seconda generazione

Il GSM è una rete con full rate di 13 kbit/s e half rate di 6.5Kbit/s. Consente l'invio di SMS e servizi supplementari come call forward, recall, e busy tone.

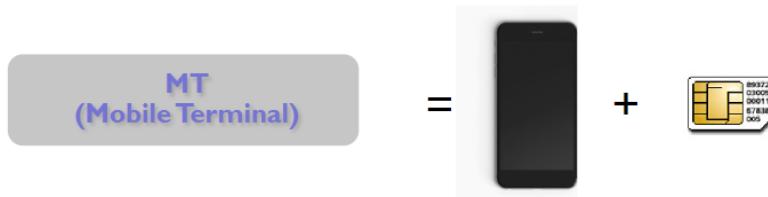
I **Mobile Station** (MS), ovvero i dispositivi, sono quelli in grado di connettersi alla rete GSM (come telefoni, antenne dei veicoli) ed hanno differenti potenze di trasmissione all'antenna:

- fino a 2W per i telefoni

**Figura 3.16:** Evoluzione della rete cellulare**Figura 3.17:** Architettura GSM

- fino a 8W per dispositivi mobili
- fino a 20W per le antenne dei veicoli

La MS è però unicamente hardware, per connettersi alla rete è necessaria una SIM, ovvero una smart card con un processore e una memoria in grado di memorizzare, crittografare, le informazioni dell'utente come il numero di telefono, i servizi accessibili, parametri di sicurezza ecc. L'identificativo univoco della SIM si chiama **MSI**.



**Figura 3.18:** Mobile Terminal

#### 3.4.1.1 Base Station Subsystem

La **Base Station Subsystem** (BSS) comprende:

- **Base Transceiver Station** (BTS): interfaccia fisica con il compito di trasmettere e ricevere. Rappresenta il *punto d'accesso* per i dispositivi e a differenza di altri sorgenti di segnale (ad esempio radio e TV) trasmette segnale solo verso gli utenti attivi. Arriva fino a 32 canali FDM per BTS.
- **Base station controller** (BSC): gestisce il controllo delle risorse sull'interfaccia radio

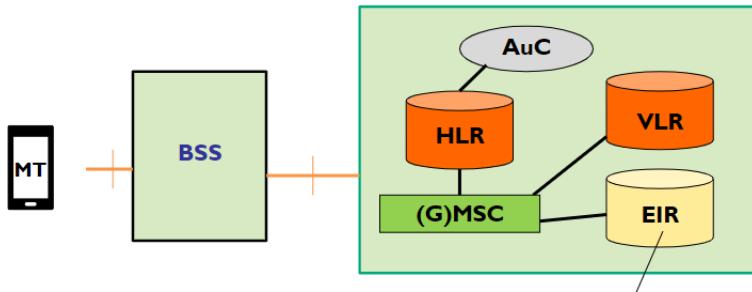
I BSC e i BTS comunicano mediante un collegamento cablato. Un *BSC controlla un alto numero* di *BTS (da decine a centinaia)*. Tipicamente, i BSC sono collocati con un MSC (Mobile Switching Center, ovvero switch a circuito per la creazione di un circuito *end-to-end*), invece di essere allocate vicino ai BTS.

Le funzionalità principali dei BSC comprendono:

- Eseguire il transcoding vocale a 13 kb/s / 64 kb/s
- Eseguire il paging
- Misurazione della qualità del segnale
- Gestione dell'handover tra BTS controllati dallo stesso BSC

#### 3.4.1.2 Network and Switching Subsystem

Il **network and switching subsystem** (NSS) ha il compito di gestire le chiamate, il service support, mobility support e l'autenticazione.

**Figura 3.19:** NSS

E' composto da:

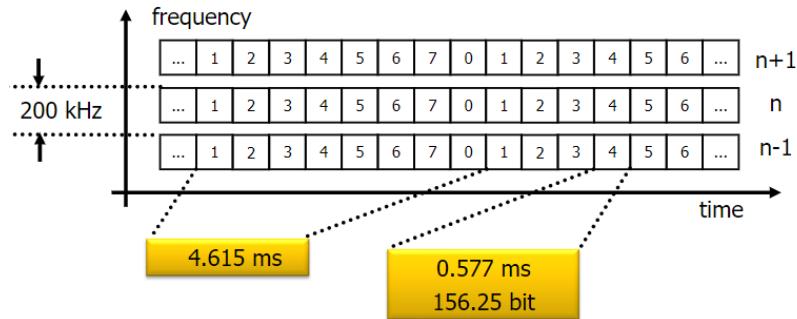
- **MSC:** *mobile switching center*, ha il compito di gestire la mobility support, call routing tra MT e GSMC (ovvero l'*interfaccia tra GSM e le altre reti*).
- **HLR:** *home location register*, si occupa di salvare le informazioni degli utenti nel database (anche permanenti come id, servizi abilitati, parametri di sicurezza) e dati dinamici per la gestione della user mobility (VLE identifier).
- **VLR:** *visitor location register*, salva nel database le informazioni relative a dove si trova il dispositivo (MT) attualmente nell'area controllata dal MSC (come id, stato on/of, LAI, informazioni di routing e sicurezza).
- **AUC:** *authentication center*, si occupa della autenticazione basata su un protocollo di tipo *challenge & response* con generazione di chiave crittografiche per comunicazioni *over-the-air*.
- **EIR:** *Equipment Identity Register*, memorizza le informazioni dei dispositivi rubati.

#### 3.4.1.3 Canali fisici

Le frequenze utilizzate per il GSM sono: 859, 900 1800, 1900 MHz, variano in base allo scopo (ricezione o trasmissione) e funzionano attraverso il sistema **FDD** (*frequency division duplex*).

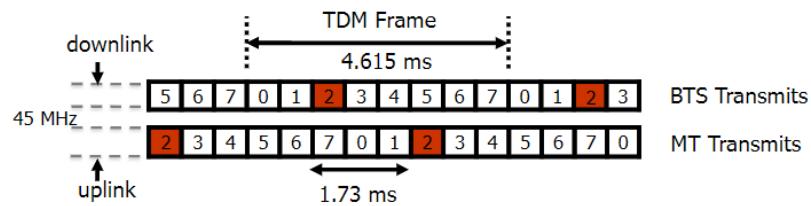
I canali GSM sono composti da una frequenza e uno slot, che identificano un canale fisico. Le trasmissioni sono organizzate in **burst** (da non confondere con pacchetti), ovvero blocchi di dati trasmessi su canali fisici. Sono simili ai pacchetti, ma funzionano su switching a circuito. La velocità di trasmissione è di 272 kbit/s. I canali possono essere acceduti con FDMA o TDMA mentre le frequenze sono divise in **FDM channels** (ciascuno largo 200kHz), che a loro volta sono divisi in **TDM frames** composti da 8 slot (ciascuno dalla durata di 0.577ms per un totale di 4.615ms).

**Nota:** Data una frequenza è uno time slot è possibile identificare un canale fisico, dunque frequenza + slot di tempo = canale fisico



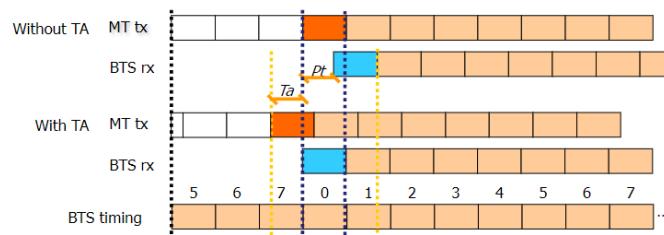
**Figura 3.20:** Accesso al canale

Il GSM non prevede una trasmissione simultanea (non è full duplex), per limitare i costi è presente un unico transceiver che consente la sola ricezione o trasmissione. Ogni MT trasmette per un *time slot* un *burst di dati* e rimane silenzioso per i rimanenti 7 slot. I frame su UL e DL sono sincronizzati in base ai time slot e shiftati di 3 slot.



**Figura 3.21:** GSM frame

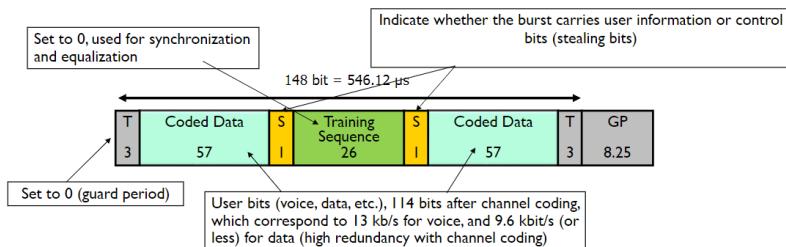
I tempi di propagazioni però non sono nulli, per cui possono nascere problemi nella struttura degli slot in quanto i burst trasmessi dai MT potrebbero arrivare al BTS quando lo slot è già finito, causando anche la possibilità di collisioni. La soluzione utilizzata è la **timing advance**: la trasmissione del MT comincia prima del reale inizio del timeslot. a inizio e fine burst sono presenti dei “bit di guardia” che permettono di sincronizzare i burst.



**Figura 3.22:** Timing advance

Analizzando più nel dettaglio la struttura di un burst, notiamo come questo è caratterizzato dai bit

di guardia, il coded data e infine lo stealing bit, il quale viene utilizzato per comunicare all'utente informazioni importanti.



**Figura 3.23:** Burst structure

I canali fisici del GSM sono composti da 8 canali, con timeslot da 0 a 7, mentre i canali logici mantengono le informazioni e specificano “cosa” è trasmesso. Sono mappati nel livello fisico in accordo a determinati criteri. I canali logici si dividono in **control channels**, i quali trasportano le informazioni di controllo (relative all'utente o alla rete), e traffic channels che trasportano le informazioni dell'utente.

### 3.4.2 4G/LTE - quarta generazione

Una delle caratteristiche di **LTE** è l'utilizzo del **FDMA** al posto del **CDMA**, che era stato pensato per gestire in modo efficiente il *fading* e sembrava essere una tecnologia migliore per il trasferimento dei dati. Il **CDMA** è però difficile da mantenere in termini tecnologici a causa del rapporto costi/benefici, oltre a essersi rivelato non sufficientemente buono. **FDMA** è un *FDM* con frequenze portanti più vicine e ortogonali (è possibile sovrapporre lo spettro) in modo da non generare interferenze.

Abbiamo una diffusione dei MIMO e il livello fisico è stato migliorato per arrivare ad downlink di 300Mb/s e uplink da 50Mb/s.

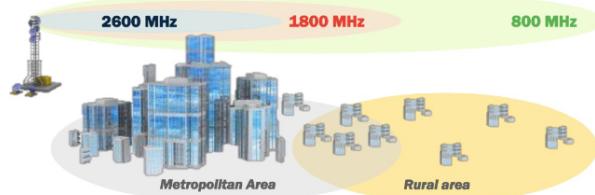
	Release 8 LTE	
	Downlink	Uplink
<b>Peak data rate</b>	300 Mbps (4x4 MIMO) 150 Mbps (2x2 MIMO)	75 Mbps (1x2 SIMO)
<b>Bandwidth</b>	Up to 20 MHz	Up to 20 MHz
<b>Peak Spectrum efficiency</b>	$\approx 16.3 \text{ bit/s/Hz}$	$\approx 4.3 \text{ bit/s/Hz}$ (1x2 SIMO)
<b>Average Spectrum efficiency [bit/s/Hz/cell]</b>	1.69 (2x2 MIMO) 1.87 (4x2 MIMO) 2.67 (4x4 MIMO)	0.74 (1x2 SIMO)
<b>Latency</b>	Data plane : 10 ms (round trip delay) Control plane : 100 ms (idle to active state)	

**Figura 3.24:** Statistiche del LTE

In LTE WCDMA è stato sostituito con **OFDMA** in downlink e **SC-FDM** uplink.

Le frequenze utilizzate sono differenti al variare della distanza:

- **2600 MHz** utilizzata per massimizzare la capacità in aree urbane.
- **1800 MHz** alta capacità ma limitata interferenza.
- **800 MHz** alta copertura e alta interferenza, per esempio nelle aree rurali.



**Figura 3.25:** Utilizzo delle frequenze

Prende il nome di downlink (DL) la parte della rete che trasmette dati dal BTS al MT, mentre uplink (UL) è la parte che trasmette dati dal MT al BTS.

In LTE è presente una separazione tra il piano controllo e il piano utente:

- **user plane:** comprende tutte le operazioni legate al trasporto dei dati degli utenti in DL o UL (*access stratum*).
- **control plane:** comprende tutte le operazioni legate al setup, controllo e mantenimento delle comunicazioni tra utente e la rete (*non access stratum*).

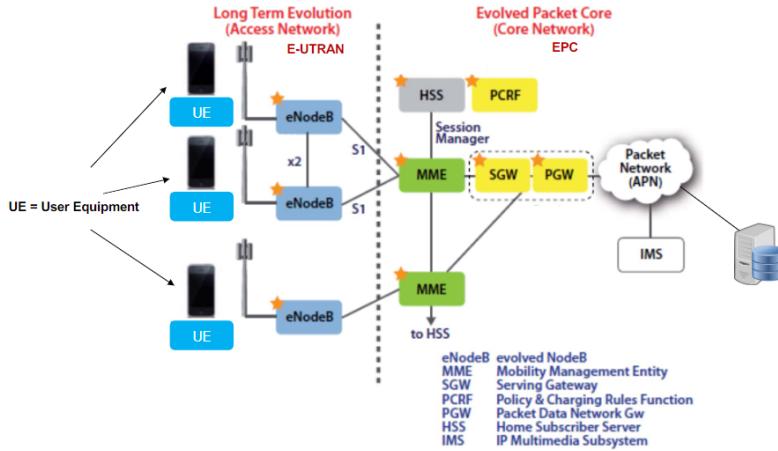
La **Radio Access Network** (RAN), la quale include tutti i dispositivi che interagiscono con i dispositivi utente, prende il nome di **E-UTRAN** mentre il **Core Network**, che include tutti i dispositivi responsabili del trasporto da/a internet verso gli utenti, viene denominato **EPC**.

Le *Base Station* prendono il nome di **eNodeB**.

### 3.4.2.1 Architettura di LTE

A differenza del GSM che utilizzava i *burst*, in LTE vengono utilizzati veri e propri pacchetti. La connessione alla rete avviene attraverso un **MME setup**, ovvero la configurazione di un *home tunnel* dalla rete di casa a quella dell'operatore.

Come mostrato nella figura di seguito, la rete si divide in **Long Term Evolution** (Access Network), ovvero E-UTRAN, ed **Evolved Packet Core** (core network) con l'acronimo di **EPC**, che rappresenta il cuore della rete e comprende tutti i nodi che forniscono funzioni di gestione della mobilità, autenticazione, session management, QoS e bearers configuration.



**Figura 3.26:** LTE architecture

**3.4.2.1.1 EPC** L'approccio adottato per **EPC** è di tipo *clean state design*, ovvero ripensato completamente da zero.

Adopera il **packet switching transport** per il traffico appartenente a tutte le classi QoS comprendente di conversazione, streaming, dirette, non in tempo reale e in background.

Viene utilizzato il **Radio resource management** per: end-to-end QoS, trasporto verso i livelli più alti, load sharing/balancing, policy management/enforcement tra differenti accessi a tecnologie radio.

Sono presenti integrazioni con le reti già esistenti 3GPP, 2G e 3G.

Le funzioni principali di EPC sono:

- **Network access control:** include network selection, authentication, authorization, admission control, policy e charge enforcement e infine lawful interception.
- **Routing e trasferimento** di pacchetti.
- **Sicurezza:** include cifratura, integrity protection e network interface physical link protection.
- **Gestione della mobilità** per tenere traccia della posizione corrente all'interno del User Equipment (UE).
- **Radio resource management** per assegnare, riassegnare e rilasciare le risorse radio prese dalle singole o multiple celle.
- **Gestione della rete** per operazioni di manutenzione.
- Funzionalità di **networking IP** per le connessioni di eNodeB, condivisione di E-UTRAN, supporto in condizioni di emergenza e altre.

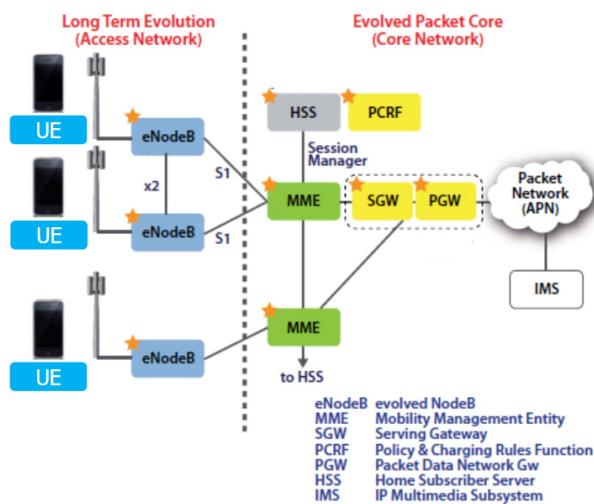
Le principali componenti sono:

- **Mobility Management Entity (MME):** si trova all'interno del control plane, supporta equipment

context, identity, authentication e authorization. Perlopiù esegue procedure di tipo *Non Access Stratum* che si dividono prevalentemente in funzioni relative al bearer management e funzioni relative alla connessione e alla gestione della mobilità.

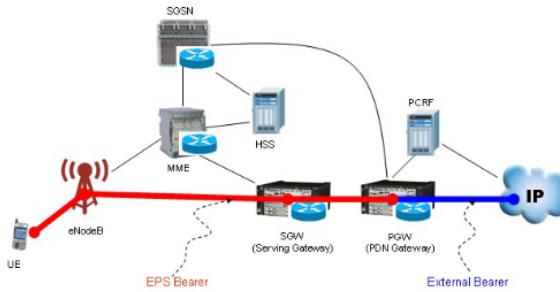
- **Serving Gateway (SGW)**: si trova all'interno del User Plane, riceve e invia i pacchetti tra gli eNodeB e la core network. Esegue il packet routing e forwarding tra gli EPC, oltre al lawful intercept. E' uno dei punti chiave per la *intra LTE-mobility*.
- **Packet Data Network Gateway (PGW)**: si trova all'interno del *user plane*, connette l'EPC con le reti esterne/internet ed esegue operazioni di assegnamento UE IP, user packet filtering e servizi di NAT. E' uno dei punti chiave per l'accesso di reti *non 3GPP*.
- **Home Subscriber Server (HSS)**: database di informazioni relative all'utente e agli iscritti. Viene utilizzato, insieme al MME, per l'autorizzazione. Funziona in modo simile al *HLR* dell'architettura *GSM*.

Nelle reti LTE, i pacchetti vengono indirizzati da/verso la access network dal *Serving Gateway* (SGW).



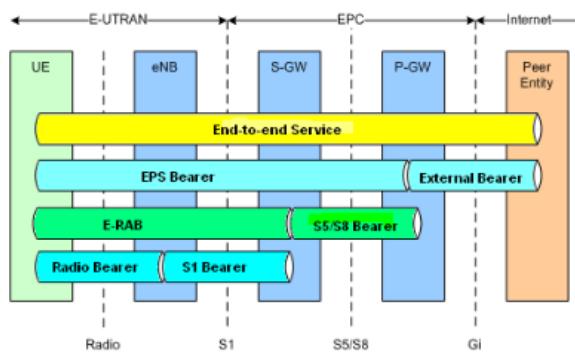
**Figura 3.27:** Componenti di EPC

**3.4.2.1.2 Bearers** Tutte le comunicazioni sono gestite attraverso dei “tunnel” denominati **bearers**, situati tra il *PGW* e *SGW* che a loro volta sono connessi a un ulteriore tunnel che parte dal *SGW* e arriva alla base station, e ancora tra user agent ed eNodeB. All'interno della rete i tunnel possono essere creati per soddisfare dei requisiti in termini di qualità del servizio, creando bearer dedicati a servizi specifici. E' presente un bearer default che stabilisce una connessione con il *PGW* quando un UE è attivato.

**Figura 3.28:** Bearers

Esistono tre differenti tipologie di bearer:

- **S5 bearer**, connette SGW con PGW (*può estendersi da P-GW al Internet*).
- **S1 bearer**, connette eNodeB con SGW. Il meccanismo di handover stabilisce un nuovo S1 bearer per le connessioni end-to-end.
- **Radio bearer**, connette UE e eNodeB. Questa tipologia segue l'utente in movimento in direzione del MME in quanto la radio esegue degli handover quando l'utente si muova da una cella all'altra.

**Figura 3.29:** Tipologie di Bearers

### 3.4.2.2 E-UTRAN

La E-UTRAN consiste principalmente di eNodeB con un interfaccia X2 per connettere gli eNodeB (due tipologie: X2 control e X2 user).

Le funzioni principali sono:

- **Gestione delle risorse radio** come radio bearer control, radio mobility control, scheduling ed allocazione dinamica delle risorse radio per uplink e downlink.

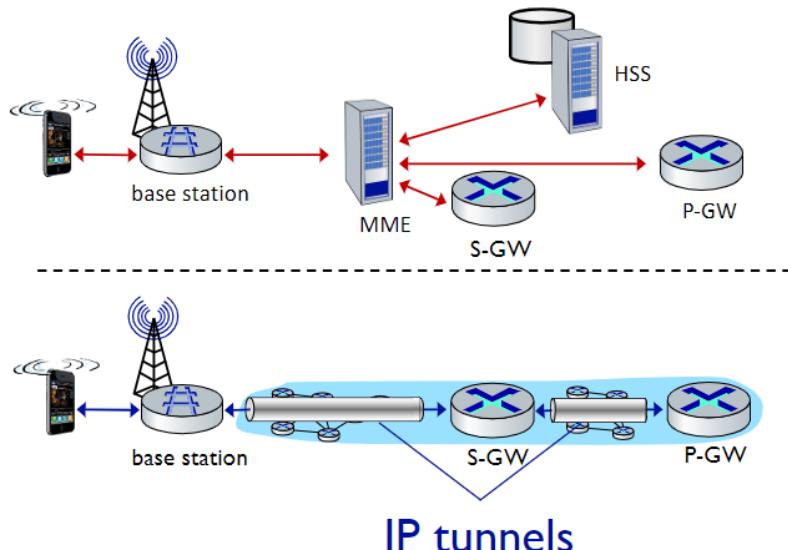
- **Compressione** (senza perdita) **degli header**.
- Sicurezza.
- **connettività** verso EPC.

### 3.4.2.3 Data Plane e Control Plane

In LTE avviene una separazione netta tra il **Control Plane** e il **Data Plane**.

Il **Control Plane** introduce nuovi protocolli per la gestione della mobilità, sicurezza e autenticazione.

Il **Data Plane**, analogamente, vede l'introduzione di nuovi protocolli al livello di data link (2) e fisico (1). Inoltre, è presente un uso esteso di tunnel per facilitare la mobilità.



**Figura 3.30:** Data Plane (basso) e Control Plane (alto)

A livello 3 è utilizzato IP mentre a quello *data link* (2) sono presenti tre sottolivelli:

- **medium access**: equivalente del sottolivello mac, si occupa dell'accesso al canale
- **Radio Link Control (RLC)**: si occupa della frammentazione e assemblaggio dei dati. Offre un *reliable data transfer*, ovvero si assicura che la comunicazione avvenga con successo.
- **Packet data convergence**: compressione dell'header e della cifratura.

Il livello fisico è gestito attraverso **OFDM** (*Orthogonal Frequency Division Multiplexing*), ovvero tante frequenze ortogonali che minimizzano l'interferenza tra i canali. Inoltre, sono definiti degli slot TDM (non diversamente dalla gestione del canale link wireless su GSM).

Il **downstream channel** (canale di scaricamento) utilizza **FDM**, TDM within frequency channel (OFDM - orthogonal frequency division multiplexing).

Per **upstream** si utilizza FDM, TDM similar to OFDM. Ciascun dispositivo attivo alloca uno o più slot di 0,5 ms su 12 frequenze. L'algoritmo di scheduling non è standardizzato, ma è lasciato all'operatore. E' possibile raggiungere fino a 100 Mbps per dispositivo.

Qui abbiamo tanti piccoli slot che devono essere assegnati dalla rete in modo dinamico, in modo da adattarsi a quello che deve essere inviato in modo efficiente.

I bit trasmessi sono inseriti all'interno di un frame che ha una struttura suddivisa in modo predefinito denominata *Physical channels*. Ciascun channel ha informazioni specifiche relative a user data, tx/rx parameters, eNB identity, network control e molto altro, oltre al format del canale stesso. Ciascun canale fisico è mappato in una porzione di un subframe LTE. I canali fisici sono divisi in downlink e uplink channels, ciascun u/d channel è ulteriormente diviso in data e control.

In uplink è possibile utilizzare gruppi di 3 TTIs per aumentare la performance e ridurre l'overhead dei livelli superiori..

La tecnologia tunneling utilizzata per le reti cellulari si chiama **GPRS Tunneling Protocol**, ovvero tunnel realizzati su UDP.

#### 3.4.2.4 Step di associazione

Un nodo per associarsi a una base station deve eseguire vari step:

1. Ogni 5ms la *base station* invia su tutte le frequenze un *broadcast primary sync signal*.
2. Il dispositivo riceve il *primary sync signal* e manda un secondo *sync signal* alla medesima frequenza. In questo modo si trovano le informazioni dalla base station come la bandwith del canale, la configurazione, cellular carrier info etc.
3. Il dispositivo sceglie il BS a cui associarsi.
4. inizia il processo di autenticazione e set up data plane.

### 3.5 Modalità di sleep

I terminali possono andare in *sleep mode* in modo da risparmiare energia. Le modalità di sleep sono:

- **light sleep**: ogni 100ms il dispositivo si sveglia per controllare se ci sono messaggi da inviare o ricevere, se non ci sono messaggi torna a *dormire*.

- **deep sleep:** dopo 5 o 10 secondi di inattività, il dispositivo si mette in deep sleep. In questo modo si risparmia molto più energia. Si da per scontato che l'utente debba ripartire da zero in quanto anche la cella potrebbe essere cambiata.

### 3.5.1 5G - quinta generazione

L'obiettivo del **5G** è superare la differenza tra rete cellulare e wifi, in modo di raggiungere un'alta mobilità sui dispositivi. Per riuscire a fornire i nuovi servizi saranno necessari, oltre al miglioramento della rete, una integrazione di risorse di rete, potenza di calcolo e archiviazione. Per ottenere ciò è necessario dislocare le varie risorse e di effettuare “*networks slices*”, porzioni di risorse riservate a una certa comunicazione che consentano di emulare ciò che faceva il “circuito” (ovvero qualità). Per fare ciò è richiesto l'utilizzo del **SDN** (virtualizzazione della rete).

Per gestire tutte le risorse in modo flessibile e dinamico viene utilizzato un “*orchestratore di rete*” denominato **orchestrator function** (o network) che si occupa di allocare in modo appropriato le risorse di calcolo e di rete. Verrà inoltre supportato il **cross-domain orchestration** (orchestrazione tra domini), ovvero la possibilità di gestire più domini di rete (ad esempio, rete cellulare e wifi) in modo da poter offrire servizi di rete più flessibili. Questo richiederà un interworking tra gli operatori dei network function layer.

Alcuni utilizzi del 5G sono:

- **eMBB:** *enhanced Mobile Broadband*, servizi ad alta qualità per utenti mobili.
- **mMTC:** *massive Machine Type Communication*, comunicazione industriale a bassa latenza.
- **URLLC:** *Ultra-Reliable Low-Latency Communication*, connessioni in grado di garantire latenze fino a 1ms in modo da mettere in comunicazione la rete cellulare con, ad esempio, il robot.

Le tecnologie utilizzate per raggiungere tali prestazioni sono:

- **forme d'onda avanzate**, alternative al puro OFDM come *RBF-OFDM*, *FBMC*, *GFDM* e *UFMC*.
- **MIMO avanzate** (antenne), che superano l'efficienza delle MIMO di LTE.
- **Millimeter Wave**, utilizzo di uno spettro ad altissime frequenze, chunk fino a 2Ghz, banda continua, altissimo throughput e bassa latenza.
- **Software define networking**, SDN è un approccio al networking il controllo del centralizzato e disaccoppiato dall'infrastruttura fisica (data plane), la quale è distribuita.
- **Network Function Virtualization**, sposta i servizi di rete dall'hardware al software, creando una virtual building blocks capace di connettersi semplicemente.
- **SDN/NFV Orchestration**, la gestione di tutte le risorse in modo dinamico e flessibile.

L'architettura di rete si divide in:

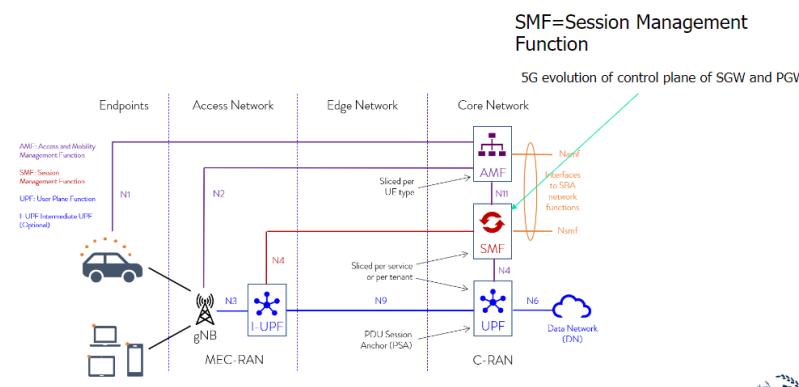
- **Radio access Network (RAN)**, basata sugli gNodeB (evoluzione degli eNodeB).
- **Edge Network (MEC)**, si occupa del hosting di potenza di calcolo e archiviazione per i servizi locali.
- **Core Network (CN)**, include tutti i dispositivi responsabili del trasporto dei dati da e verso internet attraverso i dispositivi utenti.

Le principali differenze tra le reti 4G e 5G si suddividono in:

- Nomenclatura: La radio Interface E-UTRA prende il nome di New Radio (NR), il radio access network E-UTRAN prende il nome di 5G-RAN mentre il Core Network (EPC) prende il nome di 5GC.
- Il 5G introduce una distinzione netta tra il *data plane* e il *control plane* (ancora leggermente mischiato in LTE).
- Nuove funzionalità: il network slicing implementa configurazioni differenti di RAN, è presente un nuovo framework per il QoS mediante flussi invece di end-to-end bearers e infine è previsto un nuovo approccio al 5GC che utilizza il concetto di service based architecture (SBA).

In particolare, la service based architecture vece l'utlizzo di:

- AMF: Access & mobility Function, evoluzione del MME.
- SMF: Session Management Function, evoluzione del control plane con SGW e PGW.
- UPF: User Plane Function, evoluzione del data plane.

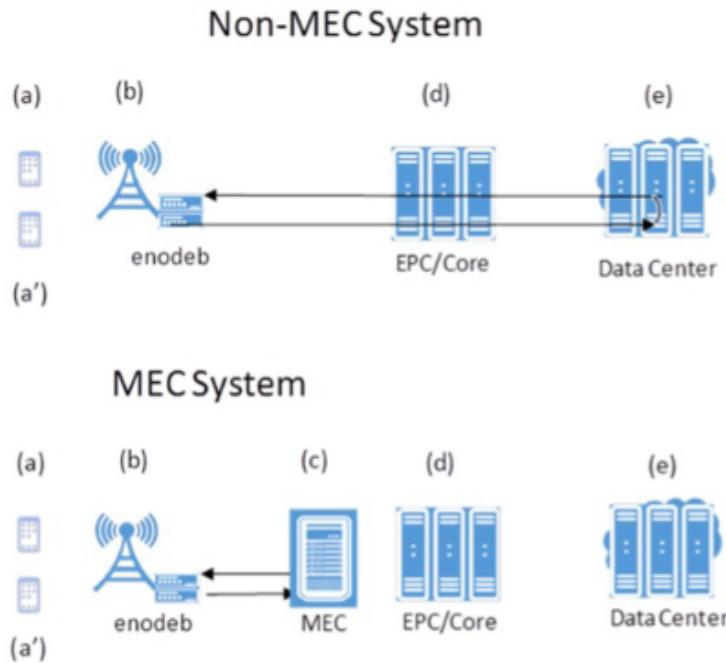


**Figura 3.31:** Service Based Architecture

### 3.5.1.1 Edge Network

L'infrastruttura edge network fornisce servizi di IT e cloud computing ai dispositivi mobili, in prossimità dei mobile subscribers. La standardizzazione è cominciata nel 2014 e pubblicata nel 2017. I Vantaggi sono:

- ultra low latency
- alta bandwidth
- accesso real time alla radio network
- contextual information
- location awareness
- flexible and extendable framework for services



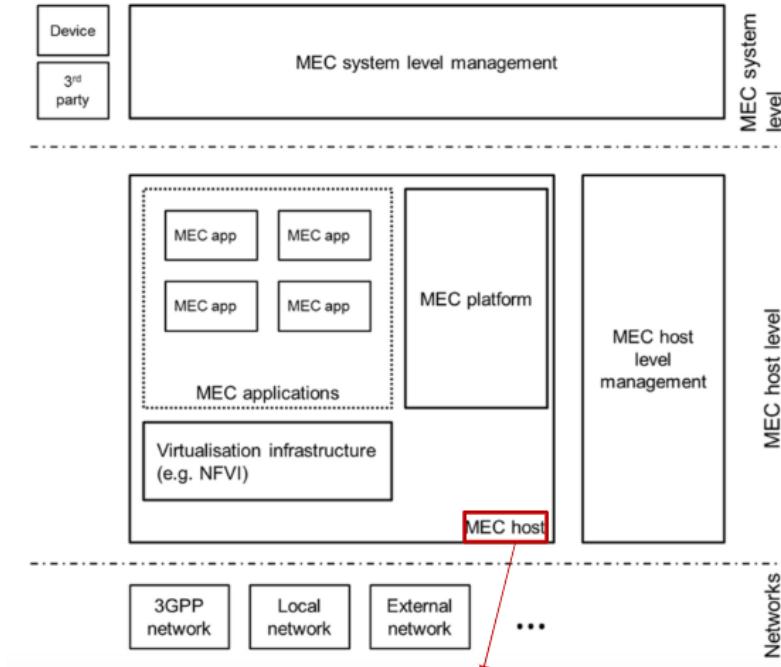
**Figura 3.32:** MEC

### 3.5.1.2 Radio Access Network

Il 5G introduce un framework flessibile basato slot, che consente l'utilizzo di un numero variabile di slot per subframe in cui la trasmissione può iniziare in un punto qualsiasi dello slot. Supporta lo slot aggregation per trasmissioni con dati molto pesanti. Different subcarrier spacing (“numerology”): slot più corti consentono uno spacing più elevato.

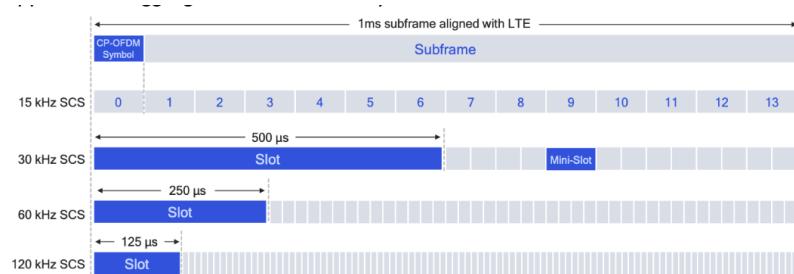
## 3.6 Mobilità nel 4G/5G

Nelle reti cellulari la mobilità è gestita chiedendo alla rete di riferimento dove l'utente si trovi (stesso approccio di trovare una persona di cui non si conosce la persona, come chiamare a casa per chiedere



**MEC host** contains the MEC platform and a virtualization infrastructure which provides compute, storage, and network resources for the MEC applications.

**Figura 3.33:** Architettura MEC



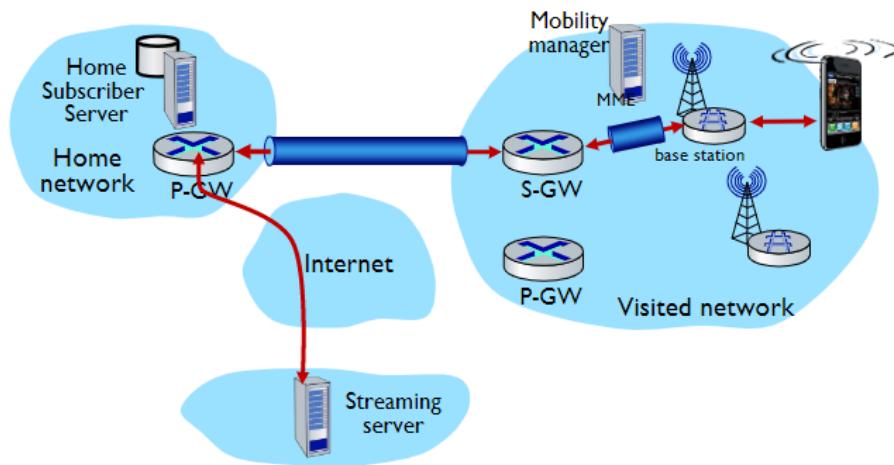
**Figura 3.34:** Slots

ai genitori dove sia). E' presente una home network e una visited network dove faccio roaming. Quando accedo alla visiting network la nuova rete mi assegna un indirizzo (spesso privato). Devo dunque dialogare con mms di quella rete in modo che possa indicare al hss che mi trovo attualmente nella sua rete. Quando un utente si sposta devo gestire *4 fasi*:

- **associazione** alla nuova base station
- **configurare** la **control plane** informando la rete dove si trova il dispositivo
- **configurazione della data plane** per la creazione dei tunnel
- **mobile handover**, se la cella dovesse cambiare (ad esempio durante la chiamata) dovrebbe essere eseguito l'handover

La configurazione della data plane tunnel per i dispositivi avviene:

- **S-GW a BS tunnel**: quando il dispositivo cambia base station, semplicemente cambia l'endpoint ip address del tunnel
- **S-GW a home P-GW tunnel**: implementazione del routing indiretto
- tunneling via GPT (GPRS tunneling protocol): i datagrammi del dispositivo vengono inviati allo streaming server incapsulati utilizzando GTP inside UDP, all'interno del datagramma



**Figura 3.35:** Configuring data plane

L'handover attraverso le base station all'interno della stessa rete cellulare avviene in quattro step:

1. il source BS seleziona il target BS, invia un Handover Request message al target BS
2. Il target BS prealloca un radio time slots, risponde con HR ACK con le informazioni del dispositivo
3. Il source BS informa il dispositivo del nuovo BS (ora il dispositivo può inviare e ricevere attraverso la nuova BS) e l'handover risulta completato agli occhi del dispositivo

4. Il source BS smette di inviare i datagrammi al dispositivo, invece li inoltra alla nuova base station (che li inoltrerà al dispositivo attraverso il radio channel)
5. Il target Bs informa MME che del nuovo BS per il dispositivo (MME istruisce S-GW di cambiare l'endopoint del tunnel al nuovo BS)
6. La base station target inoltra un ack alla base station sorgente informando che l'handover è completato e la bs sorgente può rilasciare le sue risorse.
7. I datagrammi del dispositivo possono ora utilizzare il nuovo tunnel dal target BS al S-GW



# 4 Principi del modern Lan Design

Le **Wide Area Network** (WAN) appaiono negli anni 60, sfruttavano alcuni mainframes per la potenza di calcolo e i dispositivi si connettevano da remoto (per ridurre tra più autorità i costi). Soltanto alla fine degli anni 70 compaiono le **Local Area Networks** in seguito alla comparsa dei primi minicomputer e alla successiva riduzione costi che hanno reso meno utile l'utilizzo di mainframes (*ancora usati per motivi differenti come la ricerca*).

Inizialmente WAN e LAN si sono evolute indipendentemente in quanto dovevano sopperire a esigenze differenti e, per tale motivo, erano utilizzati protocolli diversi. Soltanto successivamente LAN e WAN sono state collegate ed è stato decretato come unico vincitore il protocollo IP.

Sul livello fisico ha vinto lo standard **IEEE 802**, in particolare **802.3** ovvero **ethernet** e **802.11** ovvero **WIFI**. Dal punto di vista cablato invece: EIA/TIA 568, ISO/IEC 11801.

I dispositivi LAN si dividono in:

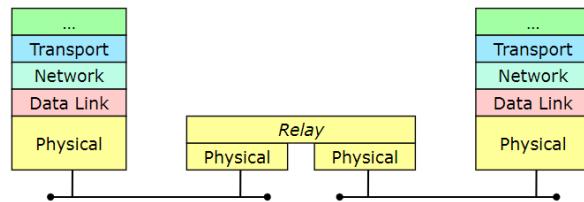
- **ripetitori** (*livello 1*): hub, stesso *dominio di collisione* ma separato *dominio fisico*.
- **bridge** (*livello 2*): switch, *dominio di collisione* separato ma stesso *dominio di broadcast*.
- **router** (*livello 3*): L3 switch, *dominio di broadcast* separato, non specifico per le LAN (*e non trattato in questa dispensa*).

## 4.1 Ripetitori

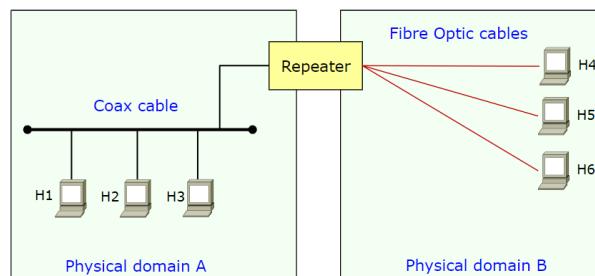
I **ripetitori** sono dispositivi di *livello 1* che consentono di interconnettere il livello fisico ricevendo e propagando una sequenza di bit. E' utilizzato per interconnettere le reti aventi lo stesso MAC (Medium Access Control) address e ripristinare la degradazione del segnale (su lunghi cavi) in modo da raggiungere maggiori distanze.

Con l'avvento del cavo in rame compaiono gli HUB che utilizzano una struttura a stella. Tutti i dispositivi connessi a un hub appartengono allo stesso dominio di collisione.

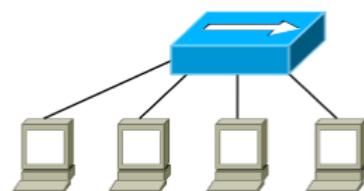
I ripetitore con più di due porte prendono il nome di **hub**, sono necessari per il twisted pairs e il fiber cabling (nella topologia hub-and-spoke).



**Figura 4.1:** Struttura dei ripetitori



**Figura 4.2:** Esempio di utilizzo di un ripetitore

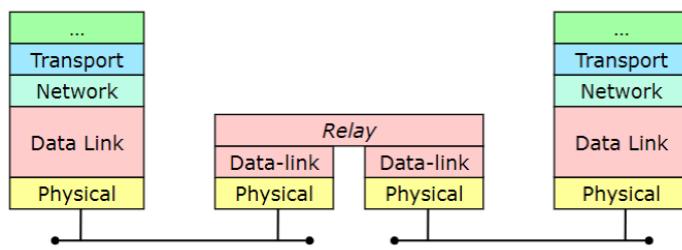


**Figura 4.3:** Hub

## 4.2 Bridge

Il **bridge** è un dispositivo di *livello 2* e pertanto è in grado di comprendere una trama ethernet. Sono implementati completamente in software e composti da due porte (per questioni economiche). Interconnettono al livello di data link (da ethernet a wifi) e hanno differenti MAC (*medium access mechanism, framing*).

**Nota:** Lo switch è un bridge a più porte.



**Figura 4.4:** Bridge

Adotta una modalità **store and forward**, ovvero è in grado di ricevere tutta la trama, “ragionarci” e poi inoltrarla verso la porta corretta che ha individuato grazie al MAC e la tabella di inoltro.

Non necessariamente interconnette link layer uguali (anche se per lo più è così), ma è pensato per supportarne anche di tipi differenti. Inoltre riesce a gestire le collisioni ed evitarle, ottenendo una **divisione del collision domain** ma mantenendo un **unico broadcast domain** (quindi il broadcast continua a funzionare correttamente).

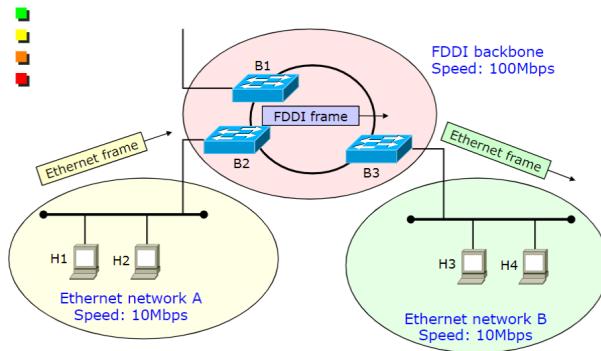
Viene utilizzato per estendere le reti LAN (specialmente per FastEthernet), ma vi sono problemi di collisione.

Il funzionamento consiste nel ricevere e ritrasmettere (dopo) un frame, il quale viene salvato, modificato e rinvia.

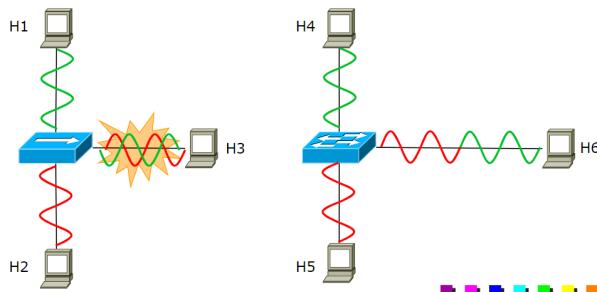
Il meccanismo **store and forward** permette un invio più intelligente dei dati nelle interfacce di output, riuscendo a disaccoppiare le collisioni sul dominio di broadcast (dunque il collision domain non è più un problema).

Bisogna però fare attenzione al fatto che sui singoli segmenti di rete possono ancora esserci collisioni, che vengono risolte attraverso la modalità **full duplex** (funzionante tra host e switch, switch e switch e host e host).

La trasmissione **half-duplex** avviene in un solo verso, è lo standard nelle interfacce di rete. Non è possibile ottenere contemporaneamente la trasmissione e la ricezione di dati. Per sopperire a tale



**Figura 4.5:** Esempio di bridge con interconnessioni



**Figura 4.6:** Bridge e collisioni

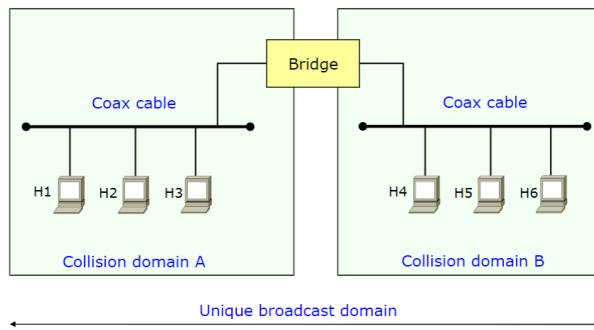
mancanza è stato successivamente introdotto il **full-duplex** con il Fast Ethernet. In questo modo la bandwidth aumenta in favore di un throughput raddoppiato (almeno teoricamente) che è in realtà un vantaggio limitato in quanto i client tendono a saturare i downlink mentre i server gli uplink. Risulta invece più utile nei bridge delle backbone dove il bandwidth è più simmetrico. CSMA/CD non è più necessario in quanto con la modalità *full duplex* non sono più presenti collisioni.

Il **dominio di collisione** si può riassumere come l'area in cui un singolo *access control algorithm*, ad esempio quella coperta da un singolo cavo fisico.

Il **dominio di broadcast** si può riassumere come l'area in cui un frame può essere propagato (ad esempio quella in cui opera una LAN). Può includere molti collision domains.

### 4.3 LAN Moderne

Le reti LAN moderne sono basate su **full-duplex**, switch e ethernet. Oggi le porte ethernet possono raggiungere i gigabit e anche se quando ci riferiamo a switch facciamo in realtà riferimento a switch



**Figura 4.7:** Broadcast collision domain unico ma separazione del collision domain

ethernet. Non è più necessario utilizzare CSMA/CD (non definito per portate sopra 1GE).

Viene utilizzata la topologia Hub-and-spoke, ovvero connessioni punto punto tra gli host e il bridge in modo da non avere domini di collisione.

**Attenzione:** Le wireless LAN funzionano in modo completamente diverso (utilizzo di CSMA/CA) e sono ancora presenti gli hub.

### 4.3.1 Transparent bridges

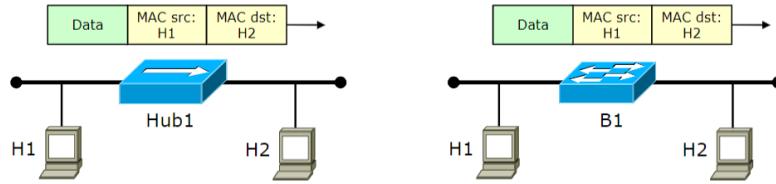
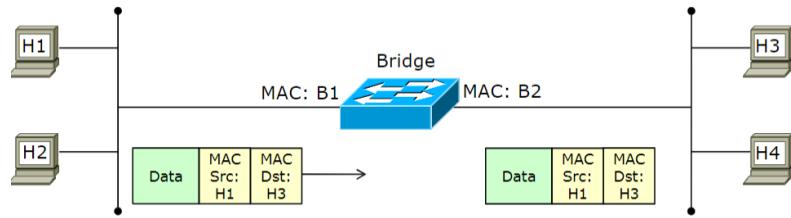
I *bridge* e gli *switch* in ethernet prendono il nome di **transparent bridge**, che si contrappongono a quelli non trasparenti ma che non vengono più utilizzati. Il nome sottolinea il comportamento *plug & play*, senza richiedere una configurazione manuale.

Per l'utente finale non devono essere percepite differenze rispetto agli hub e devono avere il medesimo funzionamento con o senza *bridges*, con al massimo una leggera differenza di performance rispetto alla rete originale. In particolare non devono essere presenti cambiamenti sui frame inviati dagli end systems (stesso frame, stesso MAC address, ecc), potrebbero invece esserci variazioni nel come questi vengono ricevuti ma non a livello di formato.

**Nota:** per l'utente gli switch non hanno indirizzi MAC, ma non è così.

Ciascuna porta di un bridge ha un indirizzo MAC (e dunque un MAC level) che non viene utilizzato per eseguire il forwarding dei data frames ma per consentire l'indirizzamento del traffico attraverso i management frames.

E' possibile eseguire il forwarding attraverso delle strategie più convenienti come quella unicast in cui l'inoltro avviene solamente verso la destinazione mediante il MAC-based forwarding, oppure in

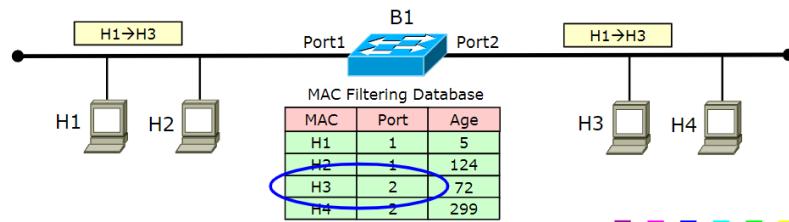
**Figura 4.8:** Stesso frame tra hub e bridge**Figura 4.9:** Transparent bridges e port addresses

multicast/broadcast attraverso il flooding inviando il pacchetto su tutte le porte eccetto quella da cui il frame è stato ricevuto.

Localmente deve essere disponibile una sorta di “routing table” che prende il nome di **filtering database** contenente gli indirizzi MAC delle destinazioni. Per far ciò è necessaria una *forwarding table locale* (filtering database), stazioni *auto-learning* (backward learning) e *loop detection* (spanning tree algorithm).

### 4.3.2 Filtering database

Un **filtering database** è una tabella contenente la “posizione” di ciascun *MAC address* trovato nella rete, ovvero la *destination port* ed *ageing time* (default 300s). Lo scopo della tabella è quello di filtrare “fuori” il traffico non voluto da un link.

**Figura 4.10:** Filtering database

La *filter table* può essere popolata manualmente (poco comodo) oppure mediante appositi algoritmi come il **backward learning**: quando uno switch riceve una trama riceve anche il mac sorgente e,

grazie a questo, capisce quale porta utilizzare per raggiungere il dispositivo. Per tale motivo la tabella ha due tipi di entry:

- **statiche**: non aggiornate dal processo di *learning*, solitamente minori di 1000.
- **dinamiche**: popolate e aggiornate dal *backward learning process*. Il massimo numero di entry è pari a 2/64000. Vengono eliminate quando le stazioni non esistono più o dopo un lasso di tempo (*default 300 secondi*).

Un esempio reale è il seguente:

```

1 Cisco-switch-1> show cam dynamic
2
3 * = Static Entry. + = Permanent Entry.
4 # = System Entry X = Port Security Entry
5
6 Dest MAC Address Ports Age
7 -----
8 00-00-86-1a-a6-44 1/1 1
9 00-00-c9-10-b3-0f 1/1 0
10 00-00-f8-31-1c-3b 1/2 4
11 00-00-f8-31-f7-a0 1/1 2
12 00-01-e7-00-e3-80 2/2 0
13 00-02-a5-84-a7-a6 2/1 1
14 00-02-b3-1e-b4-aa 2/1 5
15 00-02-b3-1e-da-da 2/5 1
16 00-02-b3-1e-dc-fd 2/4 2

```

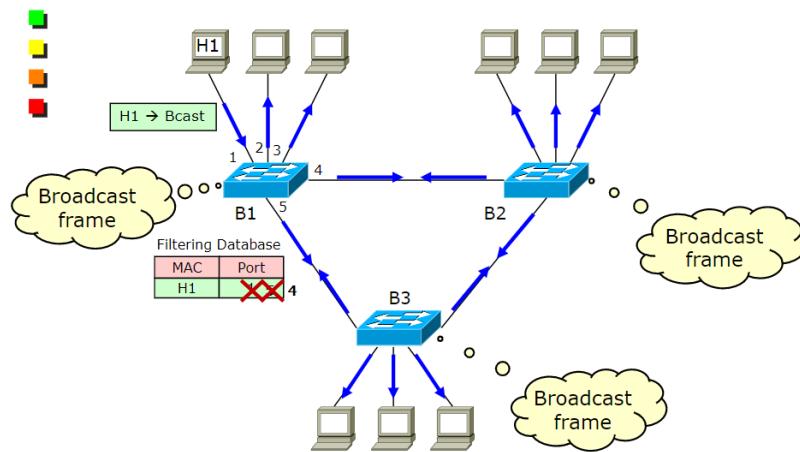
Quando uno switch non sa dove si trova un nodo (aging terminato) viene operato il *flooding*, ma non è una strategia molto efficiente in quanto comporta l'invio del messaggio su tutte le porte. In realtà è un falso problema in quanto i nodi informano di loro semplicemente col traffico, per cui tutti i nodi riceveranno il pacchetto e immediatamente tutti gli switch riescono ad aggiornare i propri database. Quando l'utente si muove non smette di trasmettere il traffico e per tale motivo al prossimo pacchetto le informazioni verranno aggiornate.

Un end-system il cui *MAC address* non è presente nel database è sempre raggiungibile, in quanto un frame inviato su un host non esistente viene sempre inoltrato verso tutte le porte. Al contrario, un end-system il cui *MAC address* nel db potrebbe non essere raggiungibile (perlopiù per motivi di aging)

Una filtering table può subire degli attacchi di tipo **MAC Flooding Attack**: vengono generati dei frame con sorgenti MAC casuali in modo da saturare la filtering table e far sì che i bridge eseguano per lo più operazioni di flooding. L'obiettivo di questo attacco è dunque costringere i bridge ad operare come fossero degli hub, in modo da poter intercettare il traffico generato da altre stazioni e rallentare la connessione. Per ridurre tale fenomeno, alcuni vendor danno la possibilità di limitare il numero di

MAC address che possono essere memorizzati su ciascuna porta.

Nel caso si utilizzi una topologia a maglia si possono generare dei cicli che rendono impossibile operare il backward learning. Solitamente ciò è dovuto a pacchetti multicast/broadcast, oltre a frame inviati a stazioni non esistenti (non presenti nel DB). In particolare se viene inviato un pacchetto broadcast si può verificare il **broadcast storm**: il pacchetto viene mandato a tutti e reinoltrato generando un loop che non termina fino a quando non vengono riavviati gli switch. In questi casi non vi è soluzione se non disabilitando fisicamente il cavo, ciò è dovuto alla mancanza di un campo *time to live* nei frame di livello 2.



**Figura 4.11:** Si possono verificare dei cicli

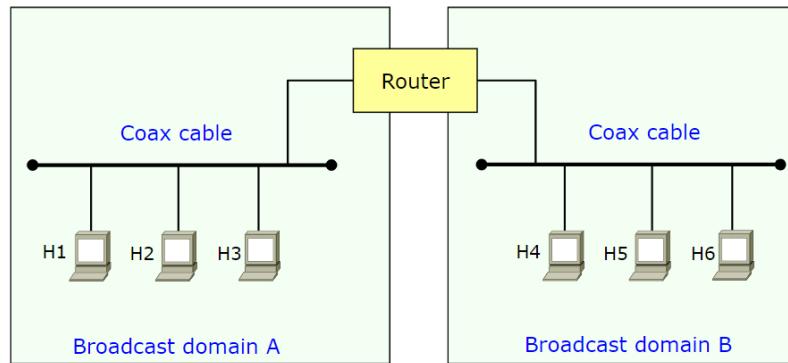
Per risolvere tali problemi è possibile utilizzare lo *Spanning Tree*, in modo da eliminare i cicli nella rete fisica mediante algoritmi che disabilitano temporaneamente i cicli (quando vengono individuate delle maglie vengono disabilitate, in modo da far diventare la rete un albero con un unico percorso da sorgente a destinatario). Altre soluzioni potrebbero essere creare delle reti *looping free*, ma non sono soluzioni sconsigliate in quanto non robuste.

## 4.4 Routers

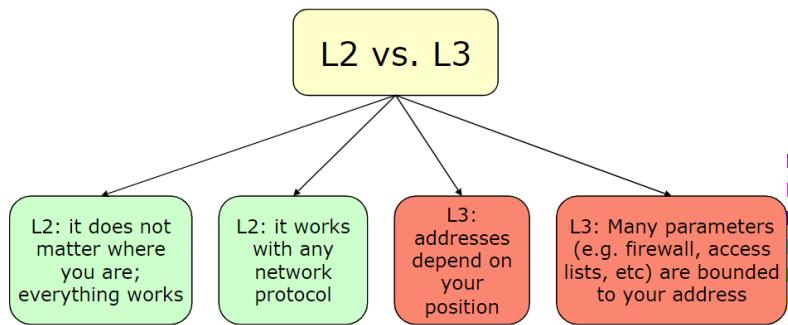
I **router** sono dei dispositivi di livello 3, non trasparenti, in grado di separare il *dominio di broadcast*.

## 4.5 VLAN

Sarebbe legittimo chiedersi se sia più conveniente utilizzare una singola grande LAN oppure multiple LAN più piccole:



**Figura 4.12:** Dominio di broadcast separato nei router

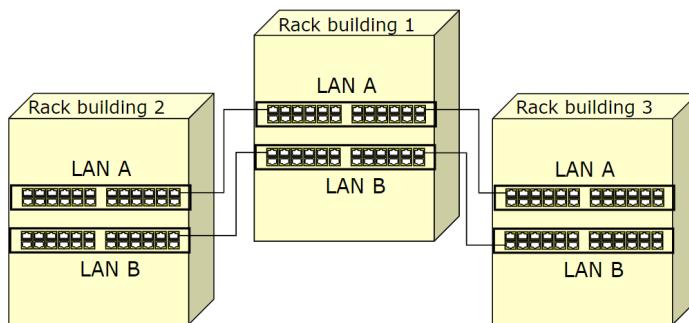


**Figura 4.13:** L2 o L3?

- *Performance*: una singola LAN ha molto più traffico broadcast oppure *flooded* (ad esempio STP reconfiguration).
- *Privacy e sicurezza*: una singola LAN è più vulnerabile a intrusioni e attacchi.
- *Gestione*: reti più piccole sono più facili da gestire.

Risulta evidente che la scelta migliore sia utilizzare LAN differenti di dimensione minore.

Per ragioni di sicurezza o semplice preferenza, è possibile dividere una rete in più parti generando reti distinte. Se dovessimo fare ciò manualmente cablando separatamente le reti, comporterebbe il dover gestire ciascun edificio con una propria rete che poi, attraverso dei cavi, connettono gli switch dei vari edifici.



**Figura 4.14:** Esempio di edifici per lan multipla

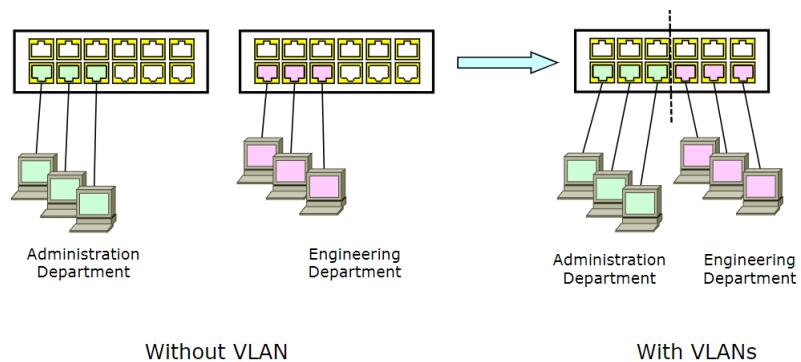
Questo è però indubbiamente molto costoso, perciò sono state realizzate le **Virtual LANs (VLAN)** che consentono a un set di porte specifiche di uno switch di simulare che facciano parte di un **dominio di broadcast differente**, utilizzando un'unica infrastruttura di rete. Per far parlare le *VLANs* è necessario un router con tutte le sottoreti connesse, permettendo la comunicazione in modo tradizionale anche se la rete di origine è in realtà la medesima.

**Attenzione:** il traffico di livello 2 non può attraversare le VLANs.

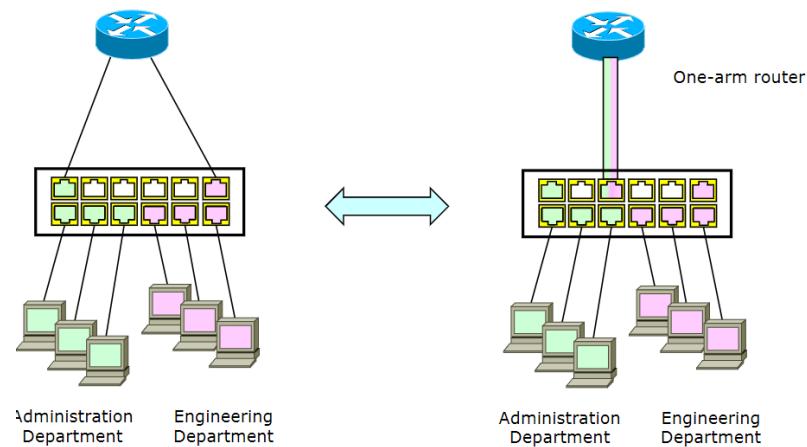
Un altro modo è connettere il router a un'unica interfaccia che lavora per entrambe le sottoreti, ottenendo il **one arm router**.

Per connettere più VLAN è necessario un router (*livello 3*) per eseguire il lookup, l'header di livello 2 viene scartato (non può viaggiare tra VLAN) in favore di uno nuovo creato con un **differente indirizzo MAC**.

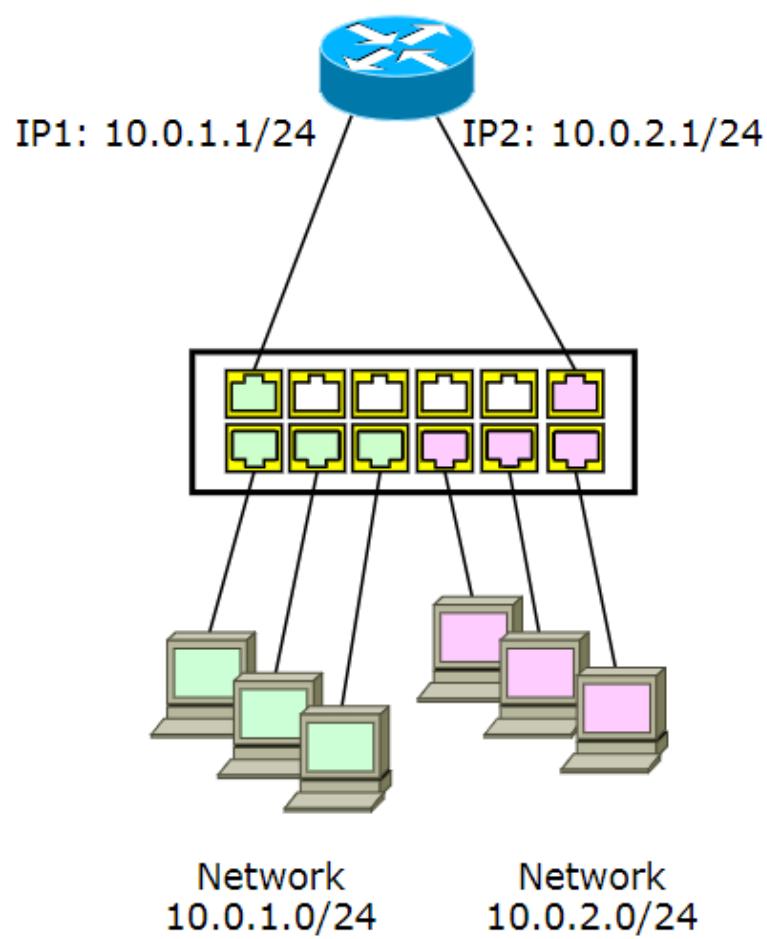
Il *broadcast* non può attraversare VLAN differenti, per questo non è possibile utilizzare ARP per individuare gli indirizzi MAC di un'altra VLAN. Gli host di Virtual LANs differenti devono fare riferimento a reti IP differenti.



**Figura 4.15:** Non servono differenti reti fisiche

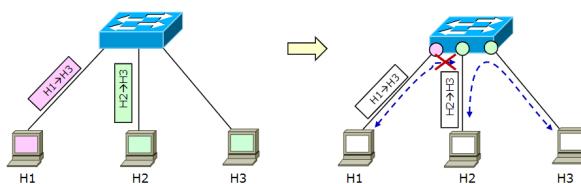


**Figura 4.16:** One Arm

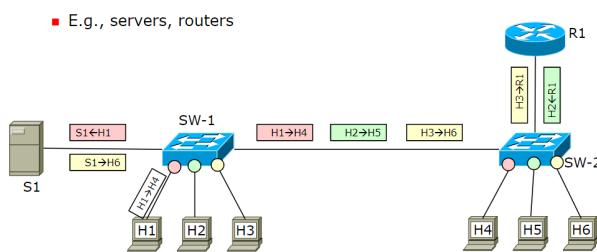


**Figura 4.17:** VLAN e indirizzi IP

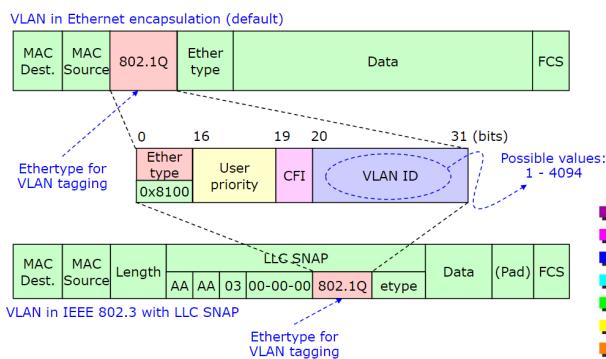
Il modo più semplice per associare un frame a una *VLAN* è marcarlo all'arrivo in base alla porta attraversata. Se però non si altera la trama, l'informazione sarà evidenziata solo all'interno dello switch locale che ha attraversato ma non agli altri switch. Per superare questo problema è stato introdotto il **tagging**, ovvero un campo aggiuntivo di 4 byte nella trama ethernet contenente il *vlanID*, in modo da identificarlo anche negli switch rimanenti.



**Figura 4.18:** VLAN su singolo switch



**Figura 4.19:** VLAN su più switch



**Figura 4.20:** Tag Encoding

Per fare ciò è necessario apportare delle piccole modifiche al protocollo MAC già esistente, in particolare un nuovo framing (per il tagging) indipendente dall'indirizzo MAC e la lunghezza massima dei frame deve essere estesa di 4 byte.

Le porte si dividono in:

- **access:** inviano e ricevono trame **non taggate** (default sugli end systems come host, switches,

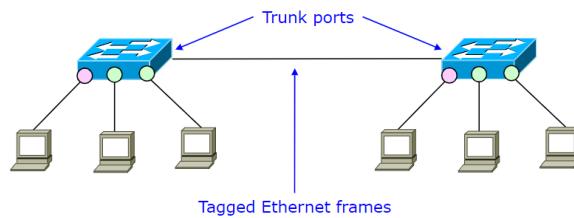
servers, routers ecc), vengono solitamente utilizzati per connettere end-stations alla rete. Il formato del frame non deve essere cambiato dagli host (che non sa dell'esistenza della VLAN).

- **trunk:** inviano e ricevono trame taggata, devono essere configurate esplicitamente. Spesso viene utilizzato nelle connessioni switch-to-switch e per connettere server/router.

Nell'immagine che segue, in cui tutte le porte sono di tipo access, l'host H1 è in grado di comunicare con l'host H4 in quanto i valori configurati su porte access non sono propagati al di fuori dallo switch.



**Figura 4.21:** Access Ports



**Figura 4.22:** Trunk ports

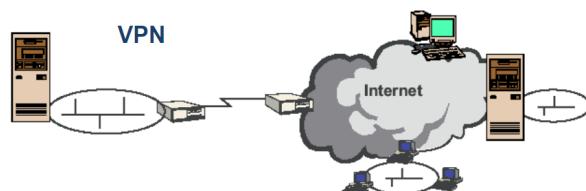
## 5 VPN

Una **Virtual Private Network** (VPN) è un insieme di tecnologie che consente di realizzare una connettività tra più sottoreti distinte in modo che possano comunicare come se fossero un'unica rete privata.

Quando un utente si connette su internet non attraversa necessariamente un unico ISP, e questo rende lo scenario molto variegato.

L'obiettivo è far sì che due sottoreti (anche in organizzazioni diverse) riescano a comunicare mantenendo le stesse politiche di sicurezza, quality of service, affidabilità.

L'utilizzo di una VPN non rende automaticamente una connessione sicura.



**Figura 5.1:** Esempio di VPN

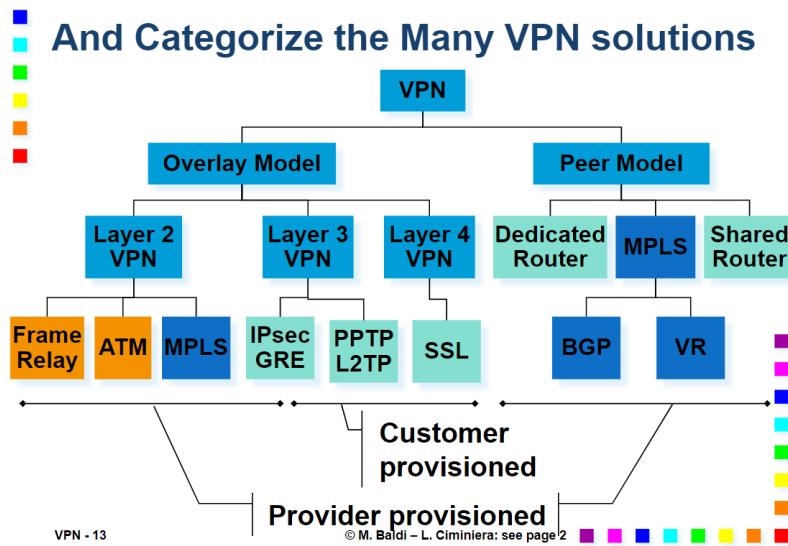
Le VPN sono contraddistinte dalla presenza di:

- **Tunnel:** Consentono di incapsulare in modo sicuro il traffico in transito sulla rete condivisa (non presente in alcune soluzioni).
- **VPN gateway:** Apre e termina i tunnel, dovranno supportare uno tra i vari protocolli specifici per fare tunneling.

Il motivo per cui le connessioni VPN sono utilizzate è quello di non dover utilizzare cavi per la realizzazione di reti private.

Definiremo alcune soluzioni:

- **site to site:** VPN a livello di sottorete (gateway), connette due reti remote (virtualizzazione del cavo di connessione).



**Figura 5.2:** Gerarchia dei protocolli

- **end to end**: sottorete a livello di host (terminali), connette due terminali remoti (virtualizzazione del cavo di connessione).
- **Access VPN / Remote VPN / Dial In**: canale sicuro tra un terminale verso un'intera sottorete (es smart working per collegarsi alla rete aziendale).

Dal punto di vista della distribuzione, si dividono in:

- **Intranet VPN**: interconnette uffici remoti della stessa azienda.
- **Extranet VPN**: interconnette aziende diverse, partner, clienti in modo da fornire un accesso controllato.

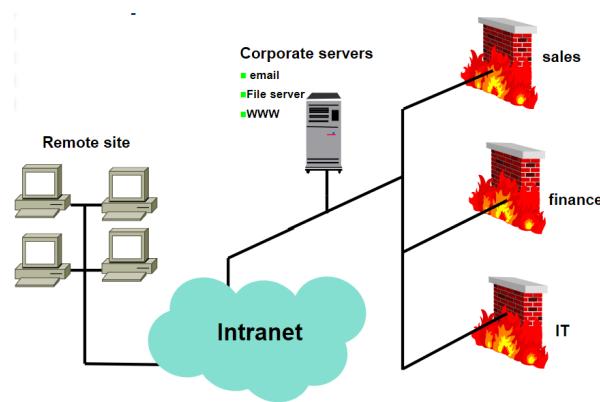
Nelle extranet sono presenti alcune limitazioni, in quanto è si vuole ridurre l'accesso alle risorse di rete mediante **firewall**, ottenere **Overlapping Address Spaces** mediante *Network Address Translation* e **controllare il traffico** in modo che quello dei partner non possa compromettere il funzionamento della rete aziendale.

Quello che contraddistingue i due tipi di rete sono perlopiù motivi di sicurezza.

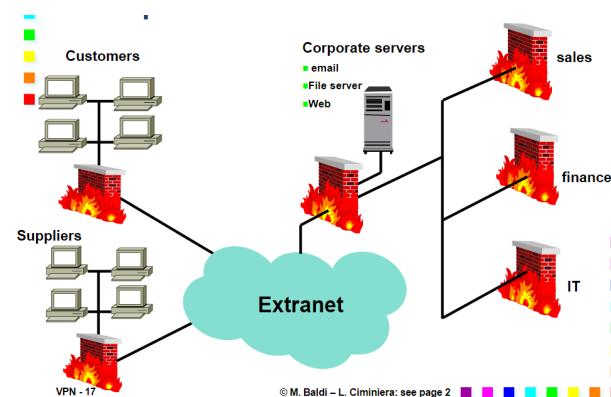
**Quindi tutto il traffico sulle reti VPN è sicuro?** No, sono necessari degli ulteriori protocolli appositi (e nelle connessioni s2s bisogna "fidarsi" che la rete locale sia sicura).

L'accesso a internet può essere:

- **Centralizzato**: gli utenti remoti utilizzano una rete IP pubblica per connettersi, disponibile solo



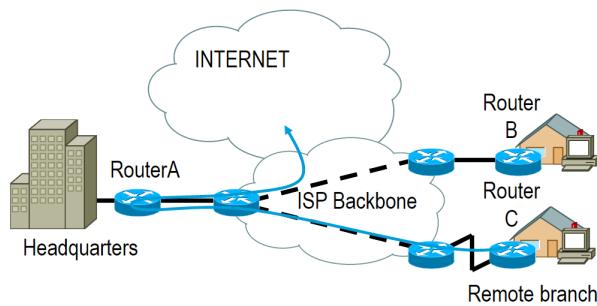
**Figura 5.3:** Esempio di intranet



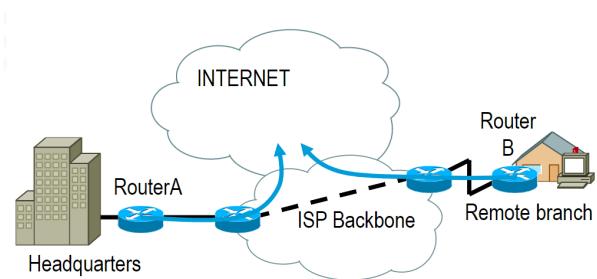
**Figura 5.4:** Esempio di extranet

negli headquarters e trasmette il traffico nella sua **interezza** da e verso internet. L'accesso è centralizzato e controllato da firewall. Il vantaggio di tale modalità è un maggior controllo.

- **Distribuito** (voluntary connection): gli utenti remoti si connettono attraverso la propria rete IP e la VPN è utilizzata solo per il traffico aziendale. Il vantaggio lo si ha nei costi che risultano essere ridotti.



**Figura 5.5:** Accesso centralizzato



**Figura 5.6:** Accesso distribuito

Riassumendo, le features che una VPN mette a disposizione sono:

- Separazione dei dati (tramite tunneling).
- Aumento della sicurezza (tramite cifratura).
- Prevent tempering (integrità).
- Identificazione delle sorgenti (tramite autenticazione).

Dal punto di vista della sicurezza gli obiettivi sono:

- **End point authentication**, verificando che un dispositivo sia chi dice di essere.
- **Integrità dei dati**, assicurando che non vengano cambiati.
- **Confidenzialità dei dati**, assicurando che non possano essere letti da altri al di fuori del destinatario.

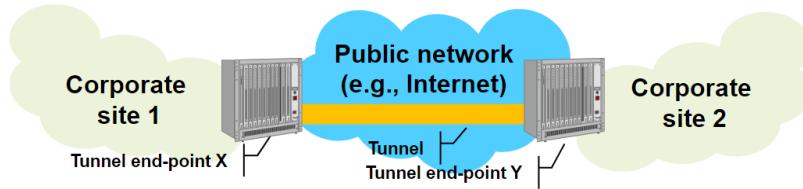
Riassumendo, le VPN consentono di ottenere:

- Separazione dei dati, mediante tunneling.
- Aumentare la sicurezza, mediante crittografia per *end point authentication*, *integrità dei dati* (dati non modificati) e *confidenzialità dei dati* (dati acceduti solo dal destinatario).
- Prevenire il *tempering*, ovvero la modifica dei dati.
- Identificare le sorgenti, mediante autenticazione.

## 5.1 Modalità di distribuzione

### 5.1.1 Site to Site VPN Tunneling (s2s)

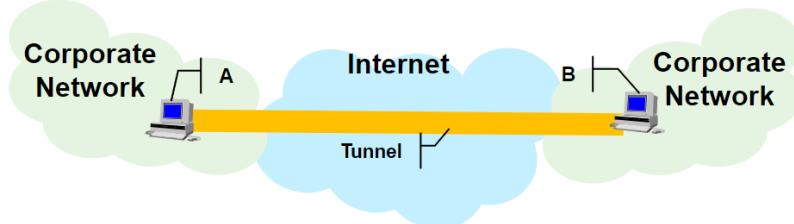
I tunnel **site to site** forniscono la garanzia che le politiche di rete avvengono a livello di infrastrutture pubblica. All'interno delle due reti aziendali la comunicazione è ritenuta sicura di default, ma se l'attaccante è interno alla rete questa risulta comunque vulnerabile.



**Figura 5.7:** S2S tunneling

### 5.1.2 End to End VPN Tunneling (e2e)

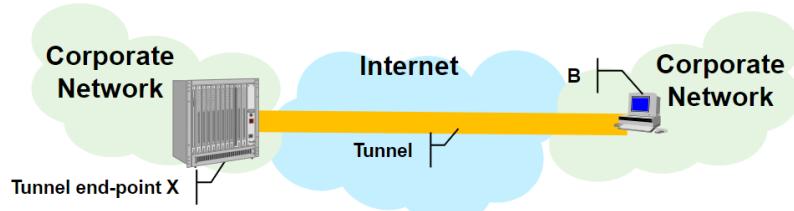
I tunnel **End to End** forniscono maggiore sicurezza in quanto il tunnel è realizzato direttamente tra i due host. Fin dall'inizio della comunicazione il traffico mantiene le stesse politiche di rete, in quanto a complessità è molto più oneroso sia in termini di costo che di gestione.



**Figura 5.8:** E2E tunneling

### 5.1.3 Remote VPN Tunneling

Il **Remote VPN Tunneling** connette un endpoint con un vpn gateway. E' possibile aggregare un'intera sottorete, ma ogni dispositivo deve essere sufficientemente robusto per connettersi.



**Figura 5.9:** Remote tunneling

### 5.1.4 Overlay Model

Nel **Overlay Model** la rete pubblica non partecipa alla realizzazione della VPN, non sa quale siano le destinazioni e la connessione avviene attraverso VPN gateways. Ciascuno di questi deve essere in contatto con tutti gli altri generando molti tunnel mesh. Il routing è ottenuto attraverso i gateway.

La creazione dei tunnel va a influenzare anche gli aspetti di routing: perdiamo il vantaggio del routing ma costa meno ed è del tutto trasparente (anche se il pacchetto potrebbe metterci un po' di più).

Si può pensare come una rete *parallela* che si sovrappone a quella fornita dal *ISP*.

### 5.1.5 Peer Model

Nel **Peer Model** ciascun VPN gateway interagisce con i router pubblici, scambiando informazioni di routing che si aggiungono a quelle fornite dal service provider. Il traffico che subisce rilouting sulla rete pubblica si muove all'interno della stessa rete VPN.

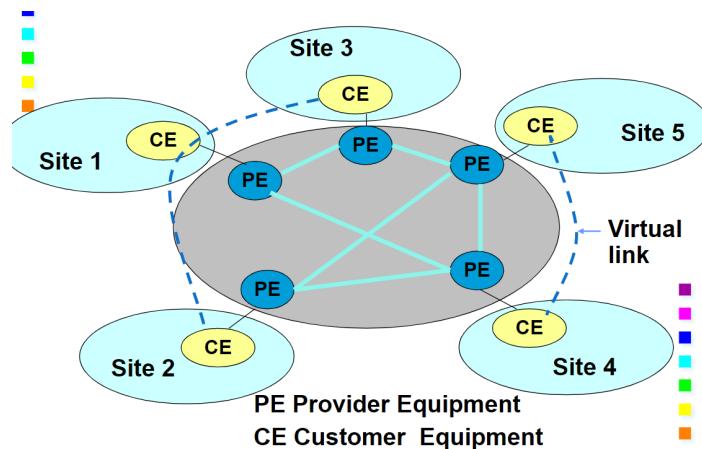
In questo approccio il routing è migliorato, ma chi realizza la VPN è fortemente coinvolto alla comunicazione di rete (non più trasparente). Inoltre, i tunnel sono tra i router compromettendo in parte la sicurezza (a livello di router posso sniffare il traffico).

### 5.1.6 Customer Provisioned VPN

Nel **Customer Provisioned VPN** il cliente implementa la soluzione VPN e possiede, configura e gestisce i dispositivi connessi adoperando del *Customer Equipment* (CE). Il Network Provider non è a

conoscenza del fatto che il traffico generato dal cliente sia VPN. Tutte le features sono implementate sui device e i CE sono i terminatori dei tunnel.

L'host deve necessariamente avere 2 indirizzi, il remote host deve terminare il tunnel e deve averlo attivo, in caso contrario può operare ugualmente ma senza VPN.



**Figura 5.10:** Customer Provisioned VPN

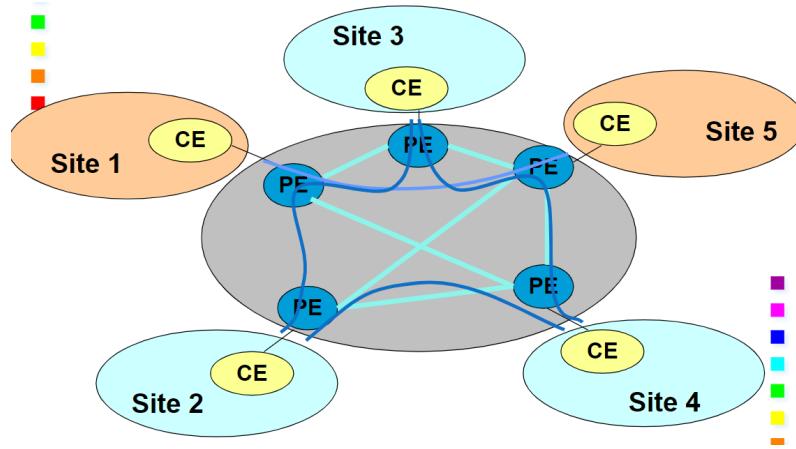
### 5.1.7 Provider Provisioned VPN

Nel **Provider Provisioned VPN** il provider implementa la soluzione VPN (quindi sotto il controllo dell'azienda), e la VPN stessa è mantenuta dal provider che si occupa di gestire i dispositivi. Il *Customer Equipment* si potrebbe comportare come se si trovasse all'interno di una rete privata, i terminatori dei tunnel sono dei *Provider Equipment*. E' meno costosa ma richiede la "fiducia" del provider.

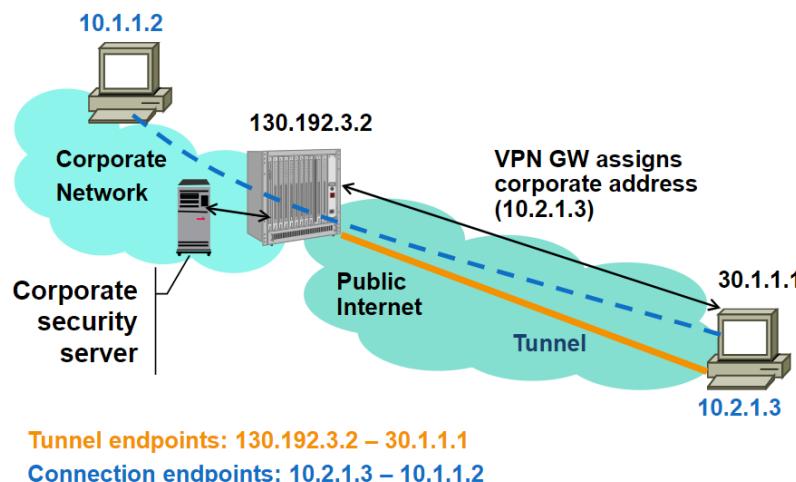
Il remote host deve essere sempre nella VPN, obbligando l'utente ad installare determinati dispositivi. In questo modo si ha un solo indirizzo in quanto si è sempre all'interno della VPN, necessitando di un accesso a uno specifico *Internet Service Provider*.

### 5.1.8 Access VPN Customer Provisioned

E' necessario considerare anche gli aspetti inerenti al piano di indirizzamento. Sui terminatori della VPN è necessario avere un indirizzo pubblico, costringendo ad avere due indirizzi. Tipicamente le remote access sono più semplici a livello di Customer Provisioner.



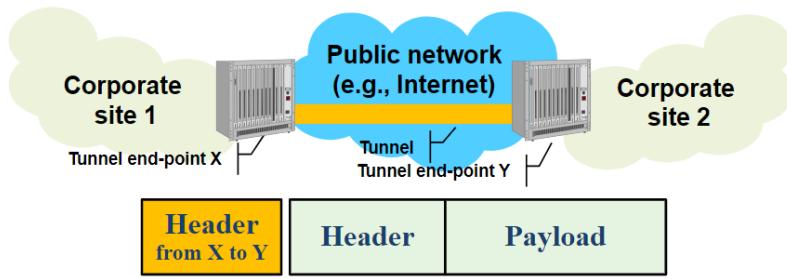
**Figura 5.11:** Provider Provisioned VPN



**Figura 5.12:** Access Customer Provisioned

### 5.1.9 Tunneling

Un pacchetto (o frame) viene inviato attraverso una rete pubblica tra due siti privati mediante nodi pubblici.

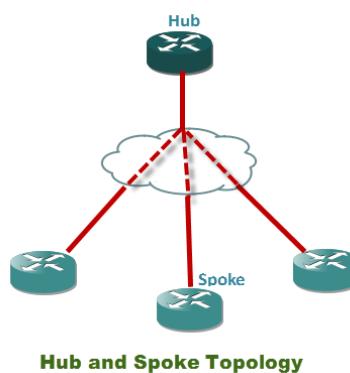


**Figura 5.13:** Tunneling

## 5.2 Topologie

Le VPN si differenziano in due topologie (virtuali):

- **Hub and spoke:** Ciascun branch comunica direttamente con il quartier generale, il quale raggruppa il flusso di dati di molte aziende (centralizzate in mainframe o data center). Il routing è sub-ottimo e sono richiesti pochi tunnel, è però presente il rischio che l'hub possa diventare un *bottleneck* rallentando le prestazioni.
- **Mesh:** Utilizza un gran numero di tunnel, più difficile da configurare manualmente ma il routing è ottimizzato.



**Figura 5.14:** Hub and spoke

## 5.3 Livelli

Una VPN può essere implementata in più livelli dello stack ISO/OSI, in ciascuno dei quali può comportarsi come *Layer N Service* oppure mediante un *Layer N Protocol*.

### 5.3.1 Livello 2

Il *livello 2* si suddivide in:

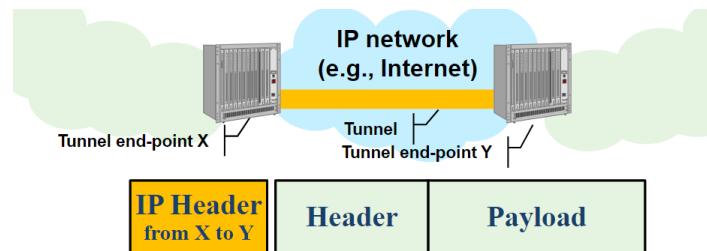
- **Virtual Private LAN service:** emula le funzionalità della *LAN* e può essere utilizzato per connettere alcuni segmenti LAN (funziona come una lan singola attraverso la rete pubblica). La soluzione emula anche i *learning bridges*, con routing basato sul *MAC address*.
- **Virtual Private Wire Service:** emula una connessione cablata, può trasportare qualsiasi protocollo.
- **IP-only Lan-like Service:** i *CE* sono IP routers o IP hosts (non ethernet switches), viene utilizzato solo IP (insieme a ICMP e ARP) per far viaggiare i dati nella VPN.

### 5.3.2 Livello 3

Le soluzioni *VPN* di livello 3 sono **standard**: i pacchetti sono inviati attraverso la rete pubblica con routing basato su indirizzi di livello 3, che possono essere **peer** (vpn/corporate/indirizzi cliente) oppure **overlay** (backbone addresses), mentre i *Customer Equipment* possono essere sia *IP routers* che *IP hosts* (in generale non ethernet switches).

I pacchetti (o frame) sono trasportati attraverso la rete come pacchetti IP nelle seguenti modalità:

- un **pacchetto IP in un pacchetto IP** (IP in IP), come *GRE* o *IPsec*.
- Un **frame layer 2 in un pacchetto IP** (IP in frame), come *L2TP*, *PPTP* (basato su GRE).



**Figura 5.15:** Layer 3

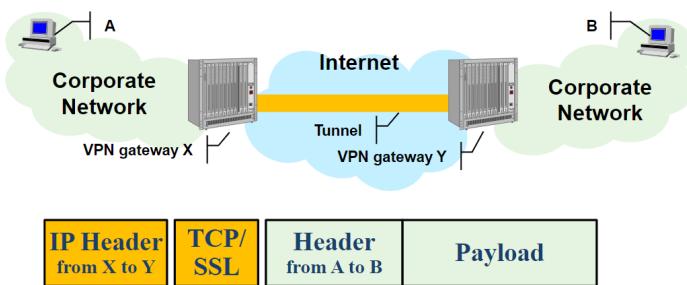
In particolare nel tunneling basato su **IP in IP**, dati due nodi A e B, dotati di indirizzo aziendale (non necessariamente pubblico), il tunneling abilita la comunicazione ma non assicura la sicurezza.

### 5.3.3 Livello 4

Le soluzioni VPN di livello 4 provvedono solo alla sicurezza. Hanno come grande svantaggio l'utilizzo di soluzioni non standard.

#### 5.3.3.1 Site to Site (s2s)

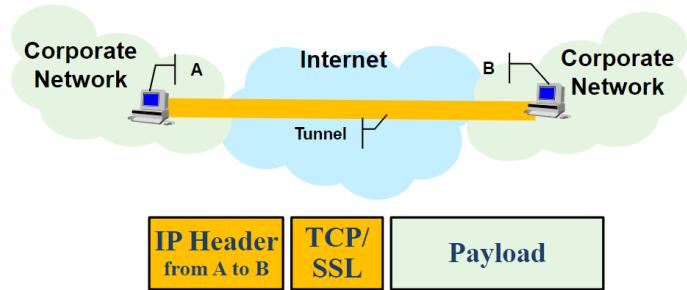
Nel **Site to Site** la VPN è costruita utilizzando connessioni TCP, sfruttato anche dai tunnel, mentre la sicurezza è garantita attraverso SSL/TSL. E' possibile avere header di livello 3 o di livello 4.



**Figura 5.16:** s2s

#### 5.3.3.2 End to End (e2e)

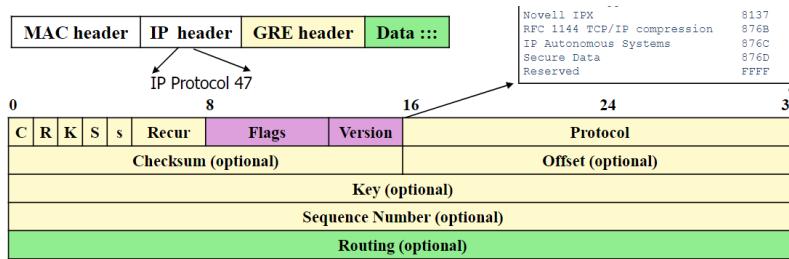
Nelle connessioni **End to End** il tunnel è terminato da un end system.



**Figura 5.17:** e2e

## 5.4 Generic Routing Encapsulation (GRE)

Il **Generic Routing Encapsulation** è un protocollo di livello 3 che si basa sul concetto di incapsulamento (IP in frame), il formato utilizzato è il seguente:

**Figura 5.18:** Formato del pacchetto

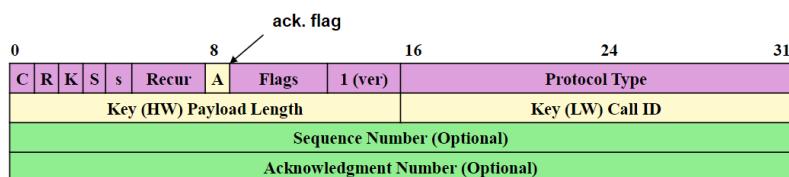
Possiamo notare alcuni campi dell'header:

- **C, R, K, S:** sono dei flag che indicano la presenza o l'assenza di alcuni campi opzionali.
- **s:** *strict source routing flag*, se il destinatario non è raggiunto quando la source route list termina, il pacchetto viene eliminato.
- **Recur:** massimo numero di volte che il pacchetto può essere incapsulato (deve essere 0).
- **protocol:** id del protocollo per il payload (*non è vietato metterci ulteriori protocolli*).
- **routing:** Sequenza di indirizzi dei router IP per ASs o per *source routing*.

**Nota:** anche se GRE è di livello 3, può incapsulare qualsiasi protocollo.

#### 5.4.1 Enhanced GRE (version 1)

Esiste una **versione estesa** di GRE denominata **version 1** che utilizza PPTP e aggiunge un *acknowledgment number* in modo da avere la garanzia di invio dei pacchetti al end-point remoto.

**Figura 5.19:** Formato di Enhanced GRE

Alcune funzionalità avanzate:

- **Payload Length** (key, 16 bit alti): numero di bytes a esclusione dell'header GRE.
- **Call ID** (key, 16 bit bassi): session ID per il pacchetto.
- **Sequence number:** per ordinare i pacchetti ricevuti, error detection e correction.
- **Acknowledgment number:** massimo numero di pacchetti GRE ricevuti in sequenza in questa sessione (ACK cumulativo).

GRE consente la **gestione del flusso dati** (flow control) attraverso una *sliding window*. I pacchetti **non possono essere ricevuti fuori ordine** e pertanto vengono scartati, ciò è causato da PPP che consente pacchetti persi ma non fuori ordine.

Ogni volta che un pacchetto di acknowledge (ack) viene inviato, i timeout values vengono ricalcolati ogni volta. Viene effettuato un **controllo della congestione** (congestion control) in quanto il timeout non causa la ritrasmissione ma è utilizzato solo per muovere la sliding window. I pacchetti verranno persi (il loro valore dovrebbe essere incrementato rapidamente).

## 5.5 Protocolli di livello 2

**Nota:** Questi protocolli di livello 2 non sono domande da esame. Cosa differente nel caso GRE e IPsec.

Per le **Access VPN** sono disponibili due protocolli:

- **L2TP** (Layer 2 Tunneling Protocol): inizialmente sono provider provisioner e non molto implementato sui terminali. E' indipendente dal protocollo di livello 2 sul host e la sicurezza è garantita da IPsec.
- **PPTP** (Point to Point Tunneling Protocol): customer provisioner, originariamente proposto da Microsoft, Apple... Ha una bassa encryption e autenticazione e utilizza un key management proprietario.

### 5.5.1 L2TP

Le due componenti principali sono:

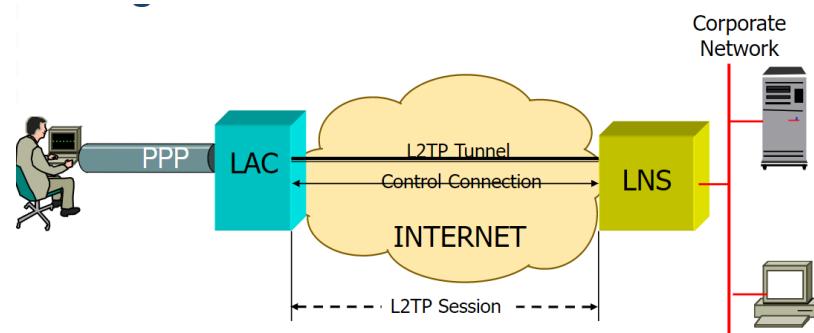
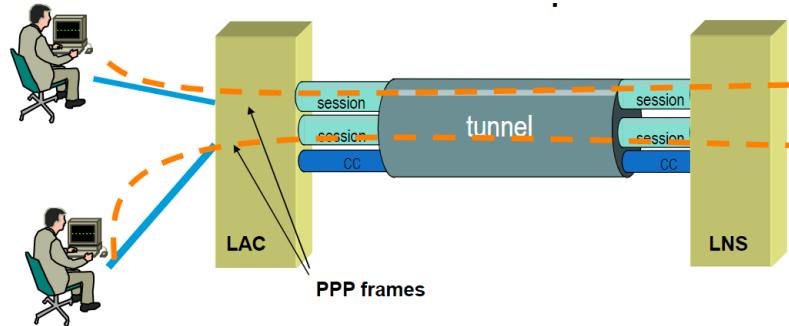
- **L2TP Access Concentrator (LAC)**: accesso alla rete, NAS (Network access server).
- **L2TP Network Server (LNS)**: corporate VPN gateway

Customer provisioned deployment mode by including LAC functionality in host

Più connessioni potrebbero esistere nello stesso tunnel e più tunnel potrebbero essere stabiliti per lo stesso LAC e LNS o multipli LNS.

Le operazioni l2TP compiute sono:

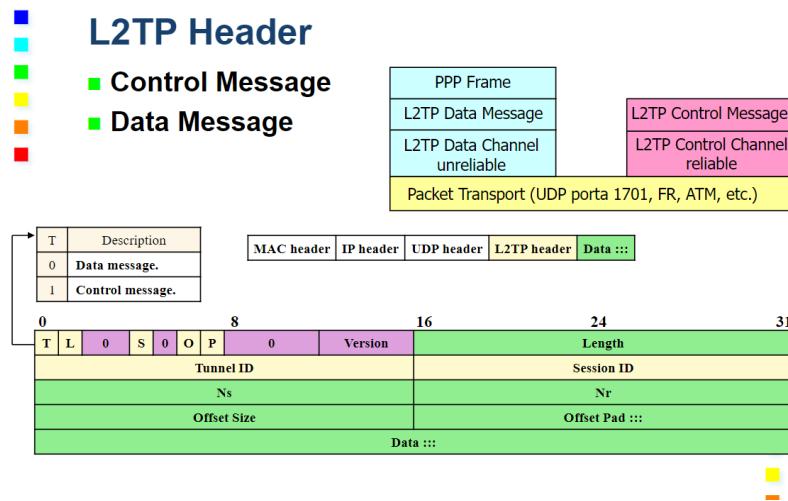
1. Stabilire una control connection per un tunnel tra lac e lns
2. stabilire una o più sessioni triggered da una call request

**Figura 5.20:** L2TP**Figura 5.21:** L2TP

La control connection deve essere stabilita prima che la connection request sia generata, e una sessione deve essere stabilita prima di inviare nel tunnel i frame PPP.

Quando il tunnel viene stabilito, il peer può essere autenticato. Per fare ciò si condivide uno shared secret tra LAC ed LNS. L2TP utilizza un CHAP-like mechanism: ovvero si utilizza un challenge-response protocol per autenticare il peer. Il challenge viene generato dal peer che lo invia al peer remoto, il quale risponde con la risposta. Il peer remoto può verificare la risposta e quindi autenticare il peer. Il tunnel endpoint scambia infine il local ID attribuito al tunnel.

L'header del protocollo utilizza un meccanismo particolare:



**Figura 5.22:** L2TP

i campi presenti sono:

- L, S, O
- P
- Ver
- Tunnel ID
- Session ID
- Ns
- Nr
- Offset

Le connessioni dati utilizzano un sequence number per individuare i pacchetti ricevuti fuori ordine. Non è presente la ritrasmissione di un flusso di dati e non vi è nessun ack per i data streams in quanto altri protocolli di livello 2 possono preoccuparsi di 2. I control packets invece utilizzano ack e ritrasmissione mediante selective repeat, la windows tra Tx e Rx è settata a 32k.

Dal punto di vista della sicurezza, l'autenticazione avviene solo in fase di creazione del tunnel. Un utente potrebbe fare snoop del traffico, e iniettare pacchetti nella sessione. Il tunnel e session ID dovrebbero essere selezionati in un modo non prevedibile (non sequenzialmente).

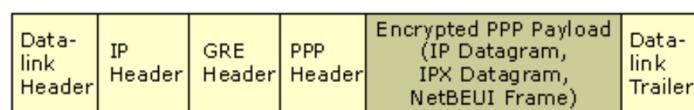
Crittografia, autenticazione e integrità devono essere assicurati da un meccanismo di trasporto (es IPsec).

### 5.5.2 Point to Point Tunneling Protocol (PPTP)

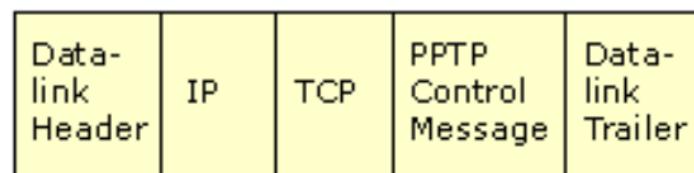
Alcuni features:

- Adopted by IETF (RFC 2637)
- Tunneling of PPP frames over packetswitched networks
- Microsoft Encryption: MPPE
- Microsoft Authentication: MS CHAP
- PPTP Network Server (PNS)
- Corporate (VPN) gateway
- PPTP Access Concentrator (PAC)
- For provider provisioned deployment mode

Sono presenti due pacchetti, uno per la parte di controllo e una per il data tunneling.



**Figura 5.23:** PPTP Data



**Figura 5.24:** PPTP Control

## 5.6 IPsec

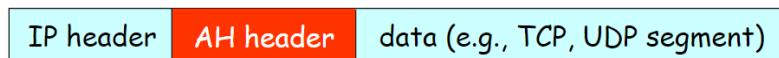
**Nota:** Questo è un argomento molto importante, spesso chiesto all'esame. È importante sapere cosa garantisce, a cosa serve ESP ed AH, le 3 proprietà ecc mentre è meno importante sapere dettagliatamente Transport mode, tunnel mode, come funziona.

Il protocollo **IPsec** si basa sull'utilizzo di due ulteriori protocolli che possono essere utilizzati insieme o meno, ovvero **AH** e **ESP**. **AH** è un protocollo che garantisce l'integrità dell'header originale e del payload, mentre **ESP** garantisce integrità ed autenticazione.

**AH**, acronimo di *authentication header*, garantisce l'integrità dei dati, l'autenticazione del sorgente ma non la confidenzialità. L'header è inserito tra l'header IP e il payload, con protocol field pari a **51**. I router processano datagrammi come di consueto, ma non è possibile adoperare il *NAT*.

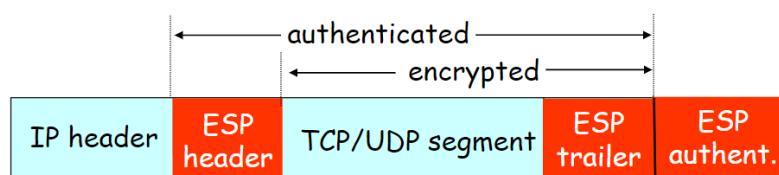
Alcuni campi di AH sono i seguenti:

- **SPI:** *Security Parameter Index*, contiene il *Session ID* e viene utilizzato per verificare la signature mediante algoritmi di cifratura e un riferimento alla chiave.
- **Authentication data:** contiene la signature generata dal router di destinazione.
- **Next header:** specifica il protocollo utilizzato nel payload (ad esempio TCP, UDP, ICMP, etc).



**Figura 5.25:** AH header

**ESP**, acronimo di *Encapsulation Security Payload*, garantisce la confidenzialità dei dati, i quali sono criptati insieme al next header nel *ESP trailer*. Inoltre, consente l'autenticazione dell'host e l'integrità dei dati, mediante una autenticazione simile a quella di AH. Il protocol field è **50**.



**Figura 5.26:** ESP header

La differenza tra l'integrità garantita da AH ed ESP risiede nel tipo:

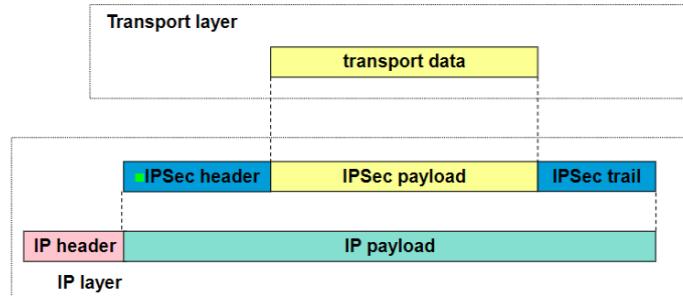
- **AH:** garantisce l'integrità dell'header originario, del payload originario e del nuovo header.
- **ESP:** garantisce solo l'integrità dell'header originario e del payload originario, **non** riuscendo per il nuovo header.

Un tunnel IPsec è perciò capace di garantire **incapsulazione, autenticazione e cifratura** tra due VPN gateways.

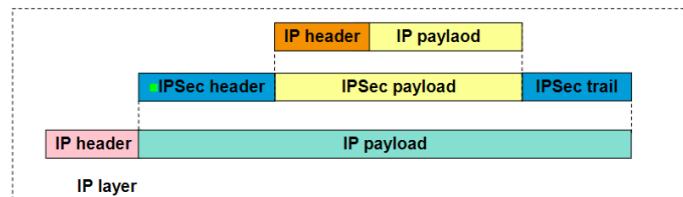
IPsec supporta due modalità di funzionamento:

- **transport mode:** l'header IP non è completamente protetto ma solo autenticato se si utilizza AH.
- **tunnel mode:** connessione tramite tunnel, in questo caso l'*header IP* è completamente protetto sia nel header che nel payload mediante l'incapsulazione attraverso un ulteriore header di livello 3.

Nel **tunnel mode** è dunque previsto che venga criptata l'intestazione IP, l'intestazione TCP/UDP e il payload del pacchetto interno.



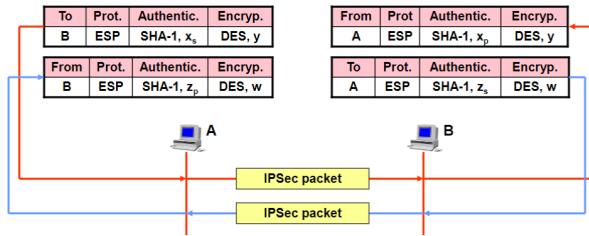
**Figura 5.27:** Header non completamente protetto



**Figura 5.28:** Tunnel Mode

Le **Security Association** (SA) sono canali logici unidirezionali che rappresentano un “*contratto*” tra due entità coinvolte nella comunicazione. Questi negoziano alcune informazioni prima di cominciare lo scambio di pacchetti IPsec. Sono identificate mediante dei *Security Parameter Index* (SPI) nel header/trailer IPsec (in base alle proprietà di sicurezza richieste).

Il protocollo **Internet Key Exchange** (IKE) viene utilizzato per stabilire e mantenere le SA in IPsec in modo da ottenere una comunicazione sicura per lo scambio dei messaggi IKE. Per lo scambio sicuro dei dati vengono utilizzati uno o più SA “figlie”, le quali utilizzano la negoziazione di chiavi tramite

**Figura 5.29:** Security Association

IKE SA (potrebbero tutti partire da uno shared secret), con la possibilità di utilizzare certificati. In particolare si parla di **Internet Security Association Key Management Protocol** (ISAKMP), utilizzato per la negoziazione di parametri IKE e dello shared secret, oltre a chiavi pubbliche, certificati e dati firmati ed autenticati (e verifica della Certificate Revocation List, CRL).

**Vi sono problemi tra l'utilizzo di IPsec e il NAT?** Si, in quanto IPsec deve garantire l'autenticazione, che non è possibile se il NAT modifica l'indirizzo IP dei pacchetti.

IPsec lavora a livello **kernel**.

## 5.7 SSL VPN

Il protocollo SSL è il meccanismo centrale su cui si basa l'accesso sicuro. Le modalità di utilizzo nelle VPN vedono:

- *site to site VPN*
- *remote access VPN*
- *Secure service access* (sarebbe e2e)

Spesso si perde il termine “VPN” o viene aggiunto “pseudo VPN”, in quanto il meccanismo cambia rispetto al modello classico. Il modello di trasporto utilizzato è sempre *TCP* o *UDP*.

Il problema di questa tipologie di VPN risiede nell'utilizzo di soluzioni **non standard**, che a causa dell'utilizzo di protocolli spesso proprietari rende la gestione più complicata.

Il motivo per non utilizzare IPsec VPN risiede nei costi troppo elevati e/o nelle troppe opzioni che necessitano una configurazione per garantire sicurezza. Un ulteriore motivo potrebbe essere il fatto che opera a livello **kernel**, per cui installazioni sbagliate possono avere conseguenze catastrofiche (oltre a installazioni difficili e rischiose).

Utilizzare **SSLVPN** ha come vantaggio:

- **Minore complessità** (installazione, configurazione, gestione).
- **Non interferisce con il kernel**, in quanto le soluzioni non sono al livello kernel.
- **Molto utilizzato**.
- **Maggiore e più robusta sicurezza** (SSL).
- **Non ci sono problemi di attraversamento del NAT** o di mascheramento (non è presente l'autenticazione del header IP e non è presente la cifratura delle porte come con ESP).

Il grosso svantaggio è però che i pacchetti vengono scartati a un livello più alto, rendendolo vulnerabile ad attacchi *DDOS*.

Alcune problematiche relative alle prestazioni possono essere:

- **IP su TCP**: Nessuna consegna di pacchetti dopo uno smarrito, inoltre la perdita comporta la strozzatura del tunnel (a causa del controllo della congestione TCP)
- **TCP su TCP**: imprevedibile
- **Ampi buffer** di trasmissione nei gateway

Le principali problematiche sono:

- **interoperabilità**: client e server devono installare lo stesso software.
- **features specifiche** del produttore.
- Ogni implementazione potrebbe avere **bug** (perché soluzioni proprietarie).
- **Disponibilità** del client sulle specifiche piattaforme.

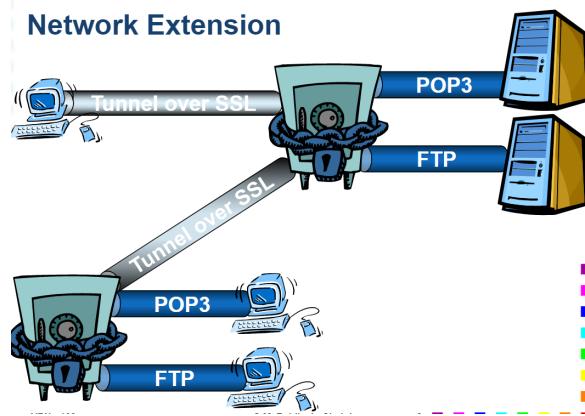
Per questo motivo sono definite “pseudo VPN”: mentre le VPN con *IPsec* connettono reti, host a reti, o host a host, le *SSLVPN* connettono **utenti a servizi** o client application a server application.

**Riassumendo:** Le *SSLVPN* utilizzano tunneling TCP o UDP, forniscono NAT traversal, packet filter traversal, router traversal e utilizzano client universali (web browser).

Alcune soluzioni utilizzano schemi di protezione simili a protezioni vpn di livello 3.

Nelle soluzioni Pseudo VPN rientrano:

- Protocolli con SSL
- Application translation
- Port Forwarding
- Web proxying
- Application proxying



**Figura 5.30:** Network Extension

### 5.7.1 Protocolli con SSL

I protocolli che utilizzano **SSL** sono definiti **secure application protocol**, richiedono i supporto del client e del server e hanno un funzionamento del tipo Protocol-over-SSL (POP-over-SSL, IMAP-over-SSL, SMTP-over-SSL).

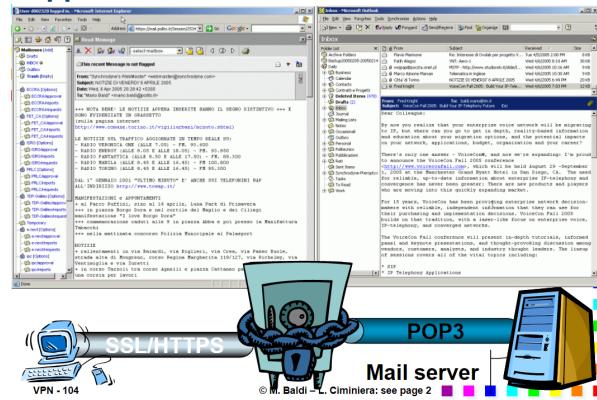


**Figura 5.31:** Protocolli con SSL

### 5.7.2 Application Translation

La **Application Translation** sfrutta protocolli nativi tra il VPN server e l'application server (FTP, SMTP, POP), sfruttando un'applicazione come user interface (ad esempio web page). Il gateway spezza in comunicazione sicura e non sicura. Inoltre, è presente HTTPS tra VPN Server e Client. Non è una soluzione adatta per tutte le applicazioni.

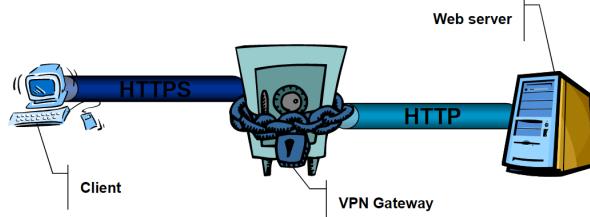
**Nota:** la filosofia che vi è dietro è di utilizzare in modo “standard” e meno protetto il protocollo nativo, che verrà richiamato dall'esterno attraverso un'interfaccia sicura SSL.



**Figura 5.32:** App translation

### 5.7.3 Application Proxying

L'**Application proxying** utilizza VPN gateway per scaricare le webpage attraverso *http* e le invia tramite *https*. Consente la compatibilità con server vecchi, in quanto la richiesta viene effettuata in modo sicuro dal client al gateway e in modo classico (ma meno protetto) verso il server. I client puntano a un SSL-VPN gateway.



**Figura 5.33:** Application Proxying

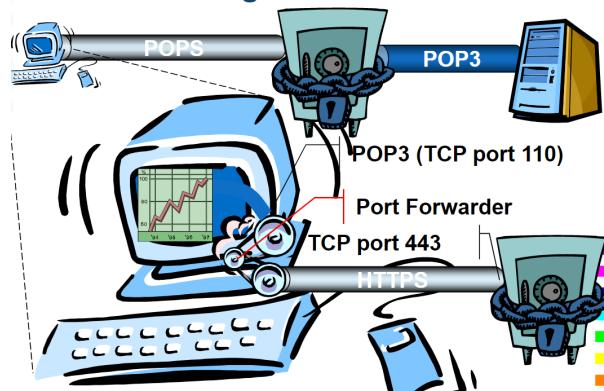
### 5.7.4 Port Forwarding

## 5.8 VPN Gateway Positioning & anomalies

### 5.9 Anomalie e posizionamento

La posizione del VPN comporta delle problematiche differenti a seconda di dove viene posizionato (in riferimento al firewall):

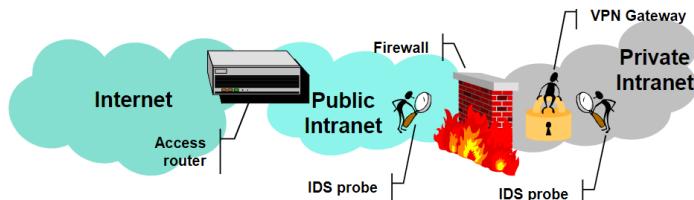
- **Internamente:** nessuna ispezione del traffico VPN oppure il VPN gateway protetto da firewall.



**Figura 5.34:** Port Forwarding

- **Parallelamente:** potenziale accesso non controllato.
- **Esteriormente:** il VPN gateway potrebbe essere protetto da un access router, Consistent policy.
- **Integrato:** Massima flessibilità.

Soltanamente vengono posti degli *Intrusion Detection System* (IDS) all'esterno del firewall senza controllo del traffico VPN e dopo il VPN gateway.



**Figura 5.35:** IDS

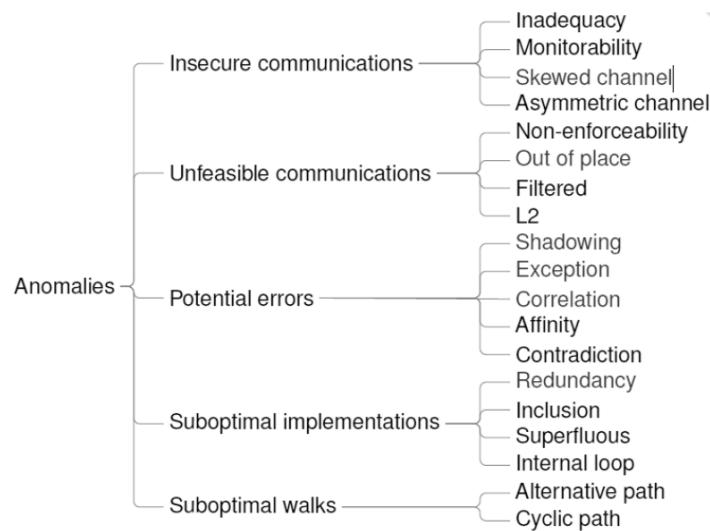
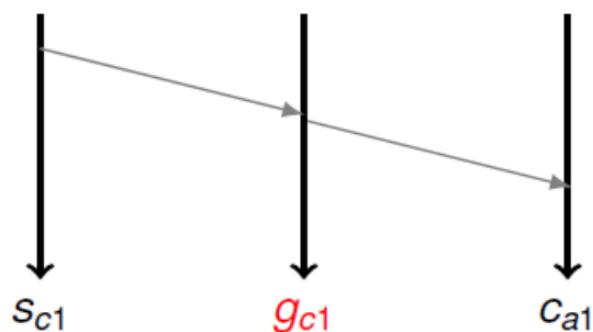
Alcune problematiche sono relative all'utilizzo del NAT all'interno della rete: nel Authentication Header, gli indirizzi IP sono parte del checksum e per tale motivo, se venisse modificato, il pacchetto verrebbe scartato. Analogamente, se indirizzo IP del tunnel IPsec non è lo stesso di quello atteso, il pacchetto viene scartato. Non è dunque possibile nemmeno utilizzare PAT/NAPT.

In transport mode le porte non sono visibili, mentre in tunnel mode l'indirizzo IP del pacchetto protetto può essere cambiato prima che questo entri nel gateway.

Altre anomalie possibili sono:

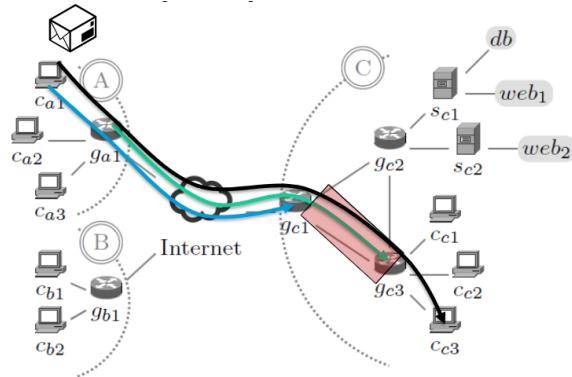
### 5.9.1 Monitorability anomaly

Si ha un **Monitorability Anomaly** quando un nodo del canale “congiunto” può vedere lo scambio dei dati.

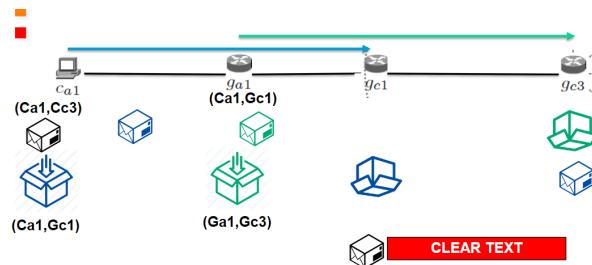
**Figura 5.36:** Anomalie**Figura 5.37:** Monitorability Anomaly

### 5.9.2 Skewed Channel anomaly

Si ha uno **Skewed Channel Anomaly** quando si ha una sovrapposizione errata dei tunnel che rimuove la confidenzialità nella comunicazione. Dunque anche avendo più livelli di sicurezza, se configurati male si può avere un problema di confidenzialità e non avere nessuna sicurezza.



**Figura 5.38:** Skewed Channel



**Figura 5.39:** Skewed Channel



# 6 Routing

## 6.1 Introduzione

Con il termine **routing** si fa riferimento al percorso che i pacchetti devono compiere nella rete, mentre **forwarding** il processo di inviare pacchetti nella rete, includendo decisioni di routing.

Distinguiamo il concetto di:

- **routing (proactive)**
- **forwarding (on the fly routing)**

### 6.1.1 Proactive routing

Il **proactive routing** è indipendente dal traffico che passa istantaneamente su un nodo, definisce quale percorso è migliore rispetto ad altri in base a una metrica scelta, determinando quali siano le destinazioni raggiungibili.

Soltanente è chiamato semplicemente *routing*.

### 6.1.2 On the fly routing

Comunemente definito **forwarding**, l'**on the fly routing** si occupa di gestire i pacchetti mediante informazioni locali come *routing/forwarding table*. E' il risultato del proactive routing o signaling.

La scelta dipende dal tipo di indirizzamento che si vuole stabilire:

- **routing by network address**: routing in base alla destinazione
- **label swapping** (es. MPLS)
- **source routing**

Quello che avviene è un'operazione di switching, ovvero il trasferimento verso una porta di output, oltre che di trasmissione. Ogni protocollo può adoperare una o più di queste strategie.

Soltanente è chiamato semplicemente *forwarding*.

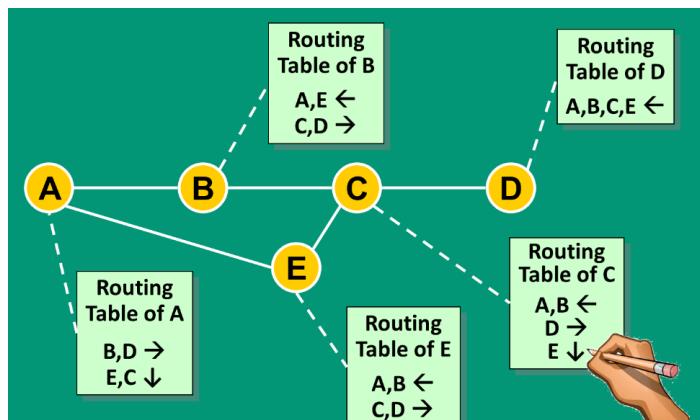
## 6.2 Algoritmi per il proactive routing

Gli algoritmi di *proactive routing* si dividono in:

- **non-adaptive algorithms**: statici
- **adaptive algorithms**: dinamici

### 6.2.1 Non adaptive algorithms

I **non adaptive algorithms** si dividono a loro volta in: **Fixed Directory routing**, il quale utilizza *static routing* è richiede una configurazione manuale, e il **flooding e derivati** (selective), anche questo con approccio statico che non cambia in base alla rete.



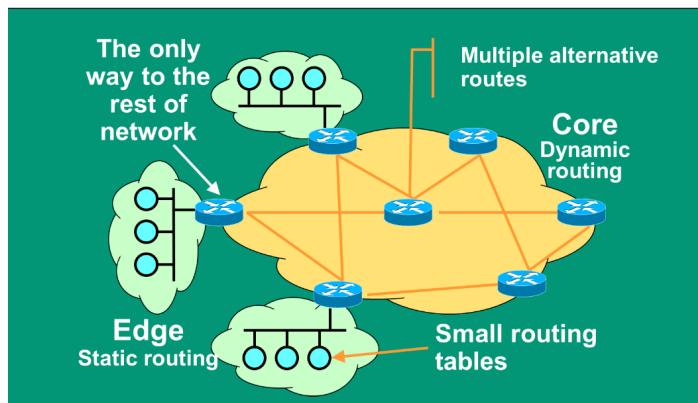
**Figura 6.1:** Fixed Directory Routing

Il vantaggio principale è il pieno controllo della rete da parte dell'amministratore, ma al costo di essere più soggetti ad eventuali errori e di un'architettura che non si adatta al cambio di topologia.

### 6.2.2 Adaptive algorithms

Gli algoritmi dinamici si dividono in:

- **centralized routing**
- **isolated routing**
- **distributed routing** (distance vector e link state)



**Figura 6.2:** Statico vs Dinamico

In riferimento al **centralized routing**, un unico nodo si occupa di gestire la rete denominata **Routing Control Center** (RCC). Ha bisogno di sapere le informazioni di tutti i nodi per prendere le strategie di routing migliori e ottimizzare le performance. Inoltre, effettua il calcolo e la distribuzione delle routing table. Il vantaggio è che semplifica il troubleshooting anche se è presente un carico di rete significativo in prossimità del RCC. Lo svantaggio è però il rischio che RCC diventi un bottleneck o un single point of failure, per tale motivo non è adatto per reti dinamiche di grandi dimensioni.

Nella **isolated routing** ogni nodo si comporta in modo indipendente senza alcun scambio di informazione. Non si ha dunque garanzia che il pacchetto venga effettivamente trasmesso. Uno scenario plausibile è in una rete lineare.

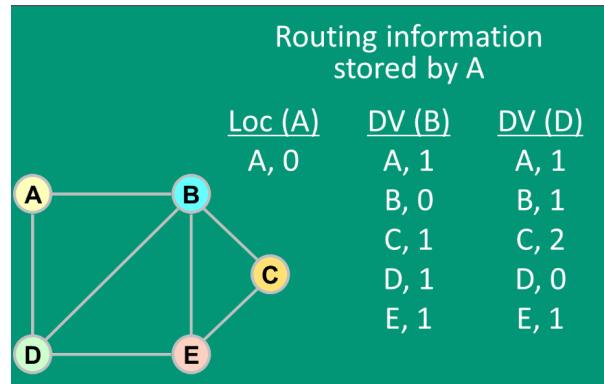
Nell'approccio **distributed routing** i router collaborano nello scambiare le informazioni sulla connettività. Ciascun router decide indipendentemente, ma in modo coerente. Combina i vantaggi e svantaggi rispetto ai due approcci precedenti.

**Attenzione:** al contrario da quanto potrebbe essere suggerito dal nome, l'isolated routing è a tutti gli effetti un algoritmo dinamico perché anche se non vi è scambio di informazioni, la configurazione varia in base al traffico.

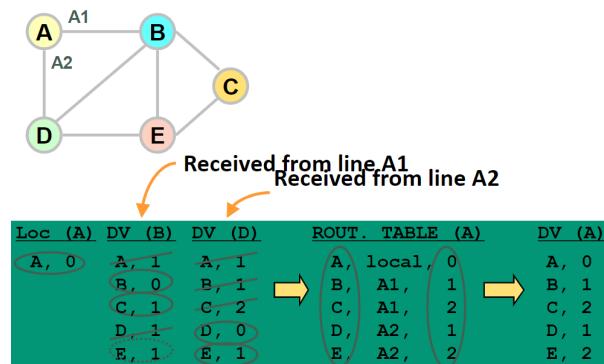
### 6.3 Distance vector (Bellman-Ford)

Nel algoritmo **Distance Vector** (DV), facente parte dei *distributed routing (adaptive algorithms)*, ogni nodo invia e riceve informazioni inerenti alla distanza con gli altri router ai nodi vicini. E' un algoritmo distribuito in cui ogni nodo ha la lista completa dei destinatari. Sono inoltre necessari dei router transitori, ovvero che non sono destinatari ma che sono necessari per raggiungere la destinazione.

Visto che ogni nodo comunica con i vicini, è importante tenere conto della distanza dal announcing routing.



**Figura 6.3:** Scenario d'esempio (1)



**Figura 6.4:** Scenario d'esempio (2)

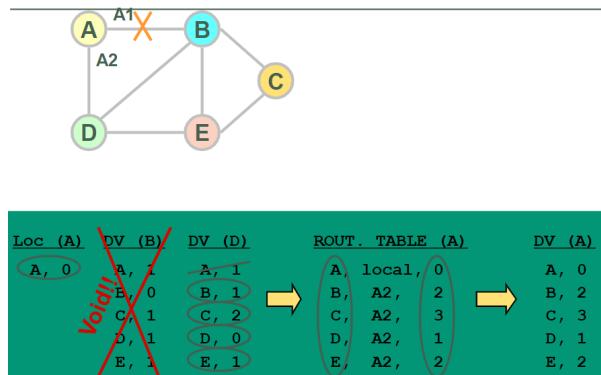
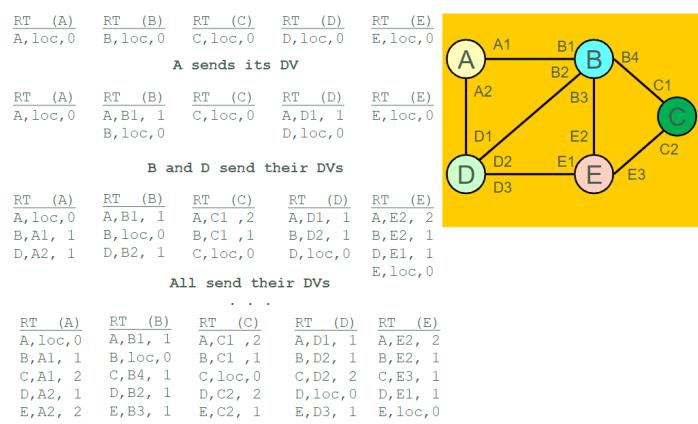
L'algoritmo cerca ogni volta la distanza minore per raggiungere un determinato nodo, tenendo conto dei percorsi alternativi in caso di guasto.

All'inizio ogni router ha solo le informazioni in locale, deve dunque mandare le proprie informazioni ai vicini in modo che si possa propagare nella rete la possibilità di poter raggiungere il nuovo nodo, ad esempio a. Il routing avviene a livello 3.

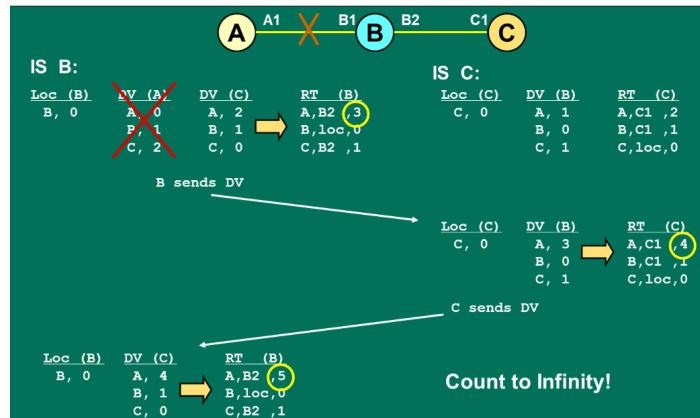
Le informazioni non vengono mandate direttamente a tutti, bensì solo ai nodi vicini (ma in riferimento a tutti i nodi).

Questo algoritmo soffre però di alcuni problemi:

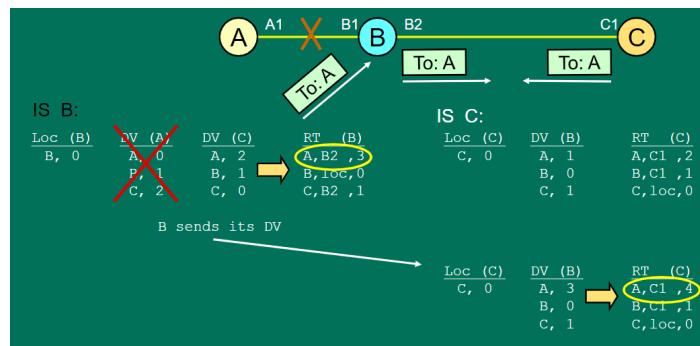
- **Black hole:** un nodo non risponde ai messaggi di routing, quindi non si ha più informazioni sulla rete.

**Figura 6.5:** Scenario d'esempio (3) con cambio di topologia**Figura 6.6:** Cold Start

- **Count to infinity:** scenario di loop, le informazioni sono propagate all'infinito.
- **Bouncing effect:** se un nodo è più vicino ad un altro, ma il percorso è più lungo, allora il nodo più vicino non sarà scelto.



**Figura 6.7:** Esempio count to infinity



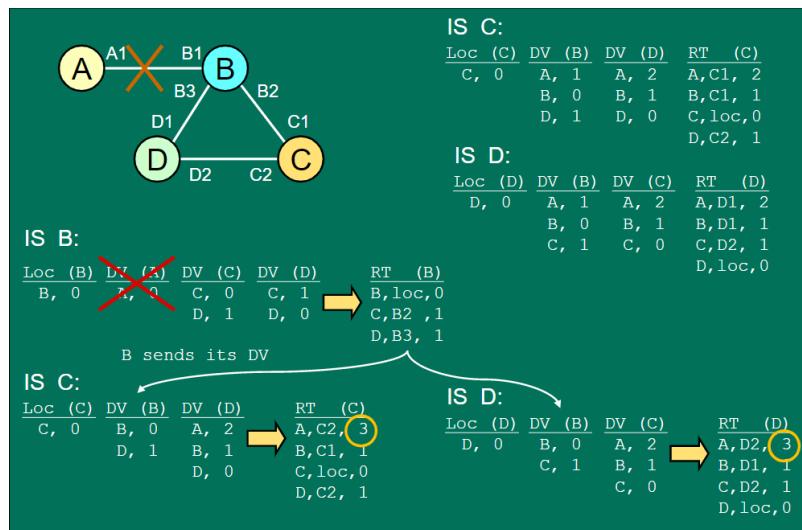
**Figura 6.8:** Esempio bouncing effect

Alcune soluzioni a tali problematiche sono:

- **Split horizon:** se C raggiunge A mediante B, è inutile per B provare a raggiungere A tramite C. Previene cicli tra due nodi, velocizza la convergenza e consente di “personalizzare” le DV per i vicini. Non risolve tutti i problemi quando abbiamo delle maglie chiuse (*mesh*). Nelle attuali implementazioni la route deve “scadere” dopo un po’ di tempo.
- **Path hold down:** se un link L fallisce, le destinazioni raggiungibili da L vengono considerate non raggiungibili per un certo periodo di tempo (*in quarantena*). Ha un tempo di copertura elevato e i router che hanno notato l’errore potrebbero non partecipare a un loop fino a quando non è scaduto un *Hold Down timer*.
- **Route poisoning:** le route scorrette sono inoltrate, al posto di essere omesse, ad un costo e distanza infinito in modo da scoprire prima cosa succede nella rete e individuare i guasti. Quando

il link fallisce il costo è incrementato, fino a quanto non si raggiunge il costo massimo (denominato infinito) e si ricerca un altro percorso. Il tempo di convergenza è più rapido e può sostituire o essere complementare al *path hold down* e *split horizon*.

Più varianti sono possibili contemporaneamente, in base al protocollo utilizzato.



**Figura 6.9:** Split Horizon su mesh

I vantaggi complessivi sono dunque la semplicità di implementazione è la semplicità di deploy per i protocolli, senza necessitare particolare configurazione.

I *routing loops* si verificano quando le *routes* hanno un incremento di costo, per questo motivo non vengono utilizzate (sono identificate da due advertisements successivi). E' possibile che succeda con il path hold down, potrebbero essere bloccate route con un incremento legittimo dei costi.

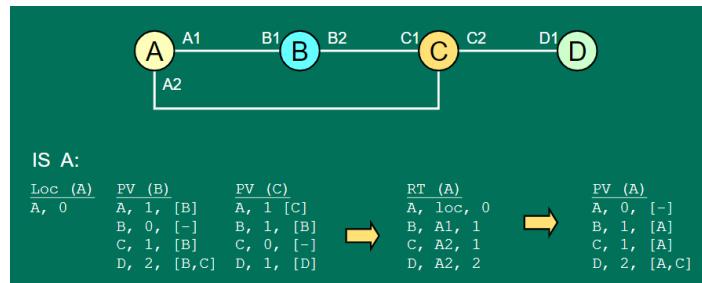
Un esempio di utilizzo può essere *Split Horizon with Poisonous Reverse*, che risulta essere più aggressivo e consente di non aspettare per la *expiration* di una *route*.

Il caso più estremo ci consente di fare in modo che i router non siano a conoscenza della topologia della quale fanno parte.

La complessità nel caso peggiore relativo al tempo di convergenza va da  $O(n^2)$  a  $O(n^3)$ , risulta inoltre limitata dai router più lenti e il set space dei router. Anche il numero di link presenti risulta essere un fattore limitante in termini di prestazioni.

## 6.4 Path Vector

L'algoritmo **Path Vector** elimina i cicli inviando, in aggiunta alle informazioni sulla distanza, le informazioni sui nodi traversati per raggiungere una determinata destinazione. In questo modo si evitano i loop all'interno dei transitori, ma nonostante ciò è molto utilizzato in quanto è un compromesso con gli svantaggi di entrambi.



**Figura 6.10:** Esempio di Path Vector

## 6.5 Link State Routing Algorithm

Nel **Link State Routing Algorithm** vengono inoltrate le informazioni relative a tutta la rete, contenente lo stato di ogni nodo (e non solo di quelli traversati). In questo modo ciascuno è in grado di realizzare una mappa locale, inviando le informazioni attraverso un *selective flooding*.

La convergenza è rapida e i *link state* sono piccoli. Il traffico di rete e lo storage sono limitati, in quanto il *neighbor greeting* è veloce ed efficiente. Raramente genera loop ed è semplice da comprendere e “riparare”, ma è più complesso da implementare e configurare.

I lati negativi vedo una implementazione complessa che consente una difficile configurazione dei protocolli utilizzati.

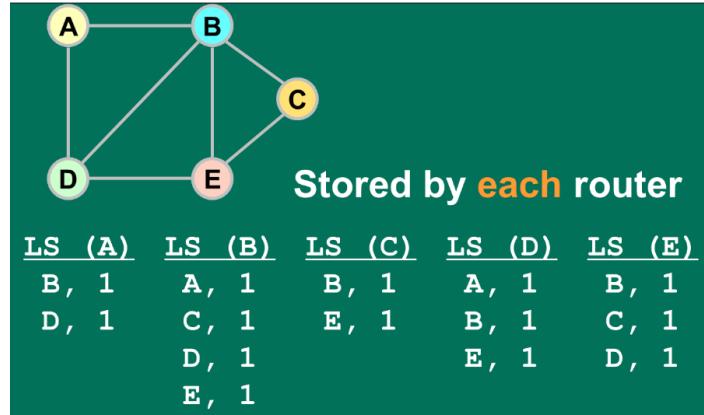
I *link state* vengono generati, in linea di principio, quando avvengono cambiamenti topologici ma nei protocolli attuali si preferisce rigenerarli periodicamente in modo da aumentare l'affidabilità.

Per calcolare le informazioni della rete è possibile utilizzare un qualsiasi algoritmo per l'albero di instradamento, come ad esempio l'*algoritmo di Dijkstra*.

Per reti piccole potrebbe non essere conveniente utilizzare link state.

Nell'algoritmo link state le informazioni vengono propagate verso tutti gli altri nodi attraverso un *selective flooding*.

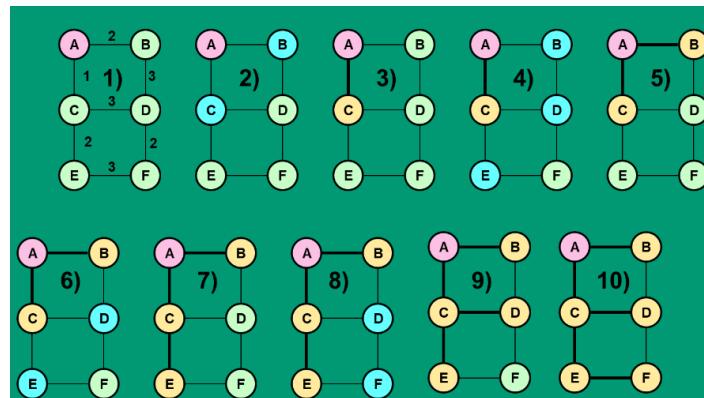
**Raramente genera loop**, non significa che non lo genera! Potrebbero generarsi in casi di guasti multipli e scenari molto grandi.



**Figura 6.11:** Link state database

### 6.5.1 Algoritmo di Dijkstra

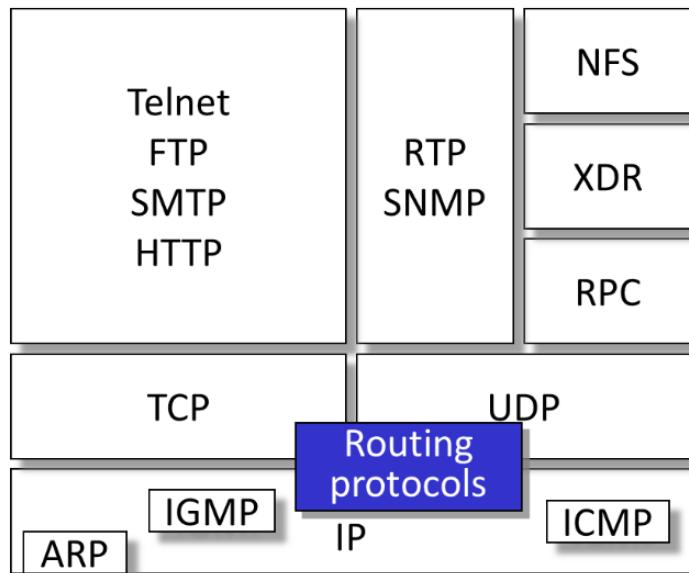
L'**algoritmo di Dijkstra** viene utilizzato per calcolare l'albero di copertura minimo di un grafo (albero di instradamento). Ha una bassa complessità pari ad  $O(L \log n)$ , con  $L$  numero di link ed  $n$  numero di nodi. Utilizza un meccanismo di **shortest path first** (ricerca del percorso più breve), dove il prossimo nodo è il più vicino alla sorgente e il *next hop* è inserito all'interno della *routing table*.



**Figura 6.12:** Esempio con Dijkstra

## 6.6 Internet Routing Architecture

I protocolli di routing viaggiano tra il livello IP e il livello TCP. Un protocollo di routing è il modo con cui si determina le rotte per lo scambio di informazioni attraverso una rete, basandosi su un algoritmo di routing di partenza.



**Figura 6.13:** Protocol Architecture

Per i routing protocol è necessario definire delle metriche, il meccanismo di encoding per il pacchetto, i parametri configurabili e lo specifico timing.

Il **dominio di routing** è un insieme di router che utilizzano lo stesso protocollo di routing, che sono connessi a una porzione della rete. Un router potrebbe far parte di più routing domains (utilizzando più protocolli di routing) e può **ridistribuire** le informazioni imparate con un protocollo mediante uno differente. Questo processo è possibile attraverso una conversione delle metriche, utilizzo di filtri di advertisement e *information source priority* tramite una configurazione dell'amministratore.

### 6.6.1 Autonomous System

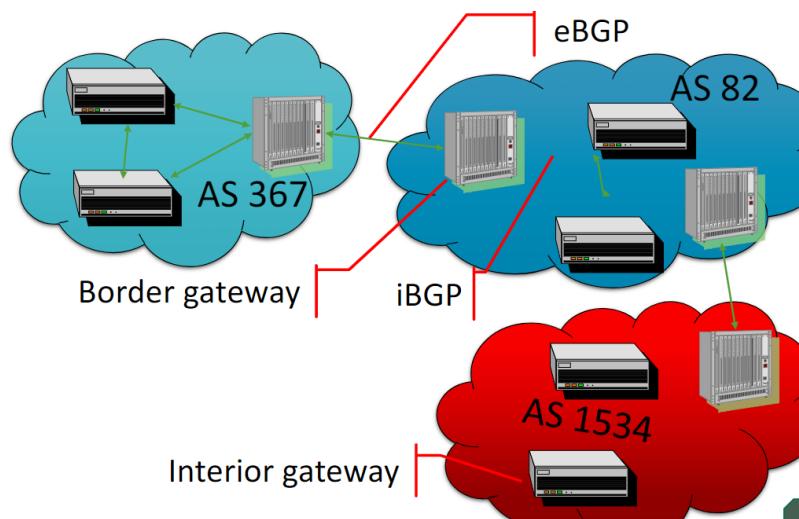
Un **Autonomous System (AS)** è un gruppo di router e sottoreti raggruppate in base alla topologia o un criterio organizzativo (ad esempio una subnet di un grande *ISP*) sotto il controllo di una singola e ben definita autorità.

L'indirizzamento e l'instradamento sono strettamente coordinati e l'interfaccia AS è controllata (data, informazioni di routing). Dal punto di vista amministrativo è possibile indicare delle scelte di routing

interno autonome e negoziare scelte di routing esterno. E' scalabile, in quanto nessuna delle informazioni è propagata "ovunque" ma è il singolo AS a decidere dove far passare i propri dati.

E' identificato da **due byte** numerici assegnati dalla *IANA (Internet Assigned Numbers Authority)*. Il range di numeri privati va da 64512 a 65534, lo scambio di informazioni di routing è controllato.

All'interno di un AS sono presenti dei dispositivi detti **border gateway**, posizionati al confine tra differenti AS. Questi dovranno comunicare tra loro e con i dispositivi interni alla rete, per tale motivo distinguiamo i protocolli in **iBGP (intra Border Gateway Protocol)** per la comunicazione all'interno della rete e **eBGP (Exterior Border Gateway Protocol)** per le comunicazioni tra gli AS.



**Figura 6.14:** iBGP e eBGP

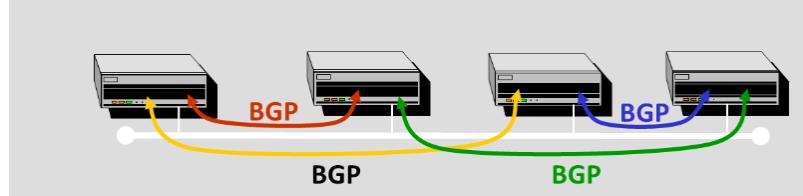
Si definisce **exterior routing** il routing tra AS. Questo non deve necessariamente seguire il percorso più corto in quanto la scelta viene effettuata in funzioni di alcune *policy*, per tale motivo non si parla più di percorso più *breve* ma bensì di percorso *migliore*.

Le destinazioni possono essere aggregate (195.1.2.0/24 e 195.1.3.0/24 in 192.1.2.0/23) secondo un routing *gerarchico*.

**Neutral Access Point (NAP)** è un punto di accesso neutrale, che permette di collegare più AS tra loro, mentre un Internet eXchange Point (IXP) è un punto di scambio di traffico tra più AS. Sono realizzabili mediante BGP.

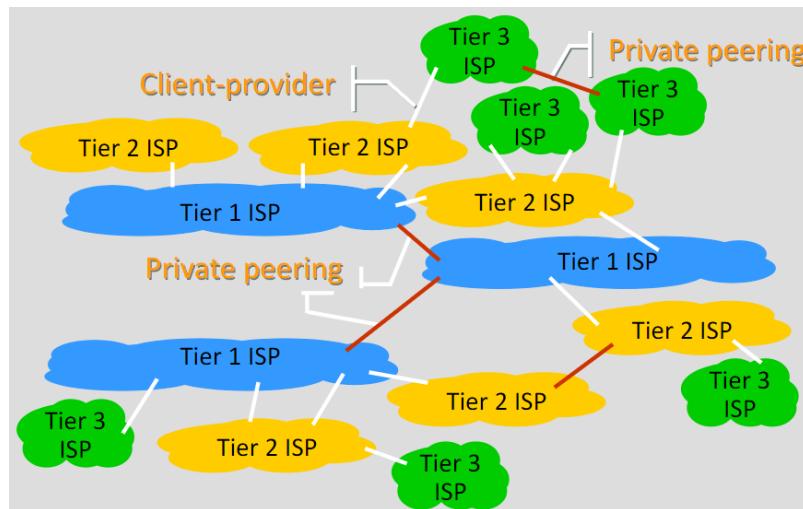
In queste reti viene inserito un elemento detto **NAP/IXP (Neutral Access Point/Internet eXchange Point)**, ovvero entità terze che mettono a disposizione delle infrastrutture che permettono a diversi *ISP* di comunicare tra loro. Tramite questi elementi è possibile connettere porzioni di rete senza dover creare nuovi canali, ma semplicemente portando tutte le connessioni verso un unico punto.

Si può affermare che un NAP/IXP consista in una LAN dove router di differenti Autonomous Systems sono connessi.



**Figura 6.15:** Implementazione con BGP

Si definisce *private peering* la comunicazione che avviene tra due ISP dello stesso tier.



**Figura 6.16:** Internet Routing Architecture

## 6.7 Protocolli di routing

I protocolli di routing si distinguono in **IGP** (*Interior Gateway Protocol*) ed **EGP** (*Exterior Gateway Protocol*).

Le *features* di **IGP** sono:

- Informazioni distribuite nella topologia
- Le route sono scelte in base alle informazioni della topologia in modo da individuare la route migliore

Le *features* di **EGP**:

- informazioni degli *Autonomous System* distribuite
- Costi amministrativi distribuiti
- Decisioni prese in base alle *policies* (trova la route “*preferita*”, non necessariamente la migliore)

### 6.7.1 Algoritmi IGP

Gli algoritmi di tipo *Interior Gateway Protocol* si distinguono in:

- **Distance Vector**: comprende **RIP** (Routing Information Protocol) e **IGRP** (Interior Gateway Routing Protocol).
- **Link State**: comprende **OSPF** e Integrated **IS-IS**.

Tali algoritmi consentono di utilizzare differenti metriche rispetto all’hop count, come: delay, bandwidth, affidabilità, carico, massima lunghezza del pacchetto. Inoltre, consentono il **multipath routing**, ovvero la possibilità di utilizzare più percorsi per raggiungere una destinazione.

Il **multipath routing** può essere causa di loop se si ammette che i percorsi utilizzati possano avere costi differenti.

#### 6.7.1.1 RIP (Distance Vector)

**RIP** è stato il primo protocollo di routing proposto, di tipo *distance vector*, nel 1988. Veniva supportato da macchine Unix e Linux. Come metrica utilizza l’*hop count*, con un tempo di convergenza di 3 minuti e un massimo di distanza di 15 hop. I messaggi di aggiornamento erano periodici ogni 30 secondi, con operazioni basate sul timeout.

Il protocollo è soggetto a frequenti instabilità e facilità nel creare percorsi di inoltro circolari.

RIP non è utilizzato in reti di grandi dimensioni.

RIP è imbustato direttamente in UDP, per ragioni legate principalmente ha una maggiore semplicità di sviluppo software.

#### 6.7.1.2 IGRP (Distance Vector)

E’ un sistema proprietario di Cisco, che supera alcuni dei problemi di RIP, diventandone l’unica alternativa nel primo periodo.

### 6.7.1.3 OSPF (Link State)

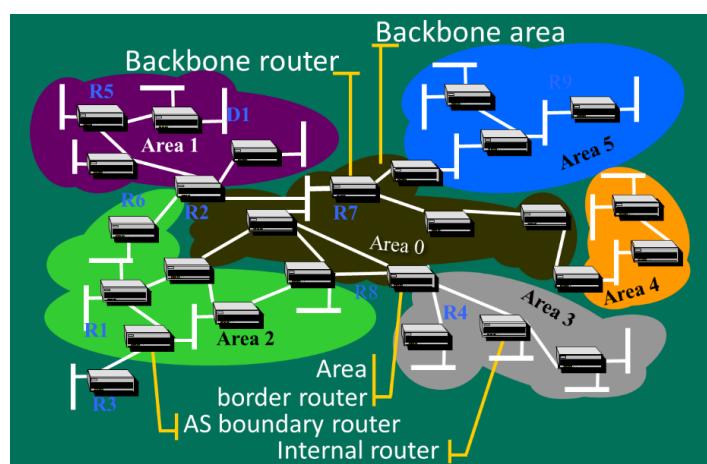
**OSPF** fa parte degli algoritmi di *link state*, utilizza un routing di tipo gerarchico. Il routing domain è diviso in aree, in ciascuna delle quali avviene una aggregazione delle informazioni. I router sanno tutti i dettagli delle zone/domain/area, ma non sanno nulla o hanno informazioni limitate relative all'esterno. Può essere iterato.

Nello *strictly hierarchical routing* i router non hanno informazioni sull'esterno della rete, motivo per cui quando il destinatario del pacchetto non è nella stessa area viene eseguito il forwarding verso un edge router, ovvero un router di confine con l'esterno della rete. Il routing è limitato in termini di efficacia, ma è altamente scalabile. I percorsi sono *sub-ottimali*, ma si ha perdita di connettività in caso di errori.

Nel *loosely hierarchical routing* si ha minore scalabilità in quanto i router devono mantenere e scambiare più informazioni, ma non è richiesto *strictly hierarchical addressing*. Tutti gli host nel *dominio B* non hanno bisogno di un identificatore comune, bensì vengono utilizzati dei prefissi. È possibile implementarlo in IPv4.

Ogni area avrà una visione completa della propria topologia interna, ma verso l'esterno soltanto i collegamenti per parlare con le altre aree, avendone una visione aggregata conoscendone i router di *frontiera*.

Per  $N$  router si hanno  $N^2$  adiacenze (dunque link). La complessità di *Dijkstra* è lineare nel numero di link.



**Figura 6.17:** OSPF

### 6.7.1.4 IS-IS (Link State)

L'algoritmo **IS-IS** è una estensione del protocollo *OSI*. Utilizza routing di tipo gerarchico con diversi livelli. Viene ancora utilizzato, ma non è più diffuso nelle nuove strutture in quanto è soppiantato da OSPF. Ha avuto in passato un largo utilizzo in grandi reti e *ISP*.

## 6.7.2 Algoritmi EGP

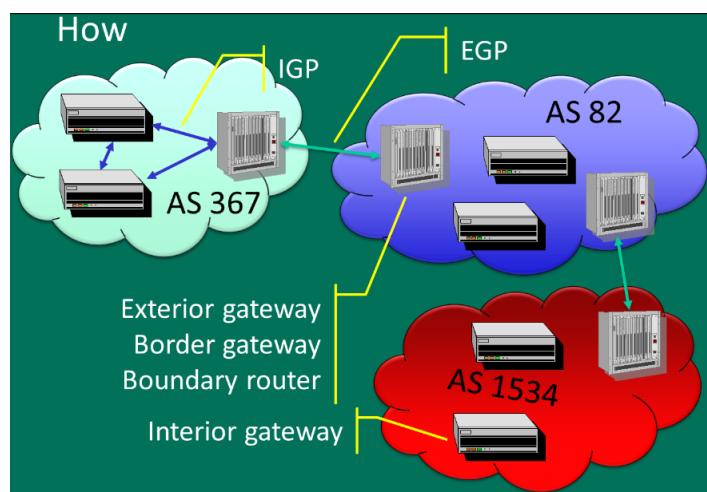
Gli algoritmi di tipo *Exterior Gateway Protocol* applicano soluzioni che non sono ne completamente *distance vector* ne *link state*, le principali soluzioni sono:

- **BGP** (Border Gateway Protocol)
- **IDRP** (Inter Domain Routing Protocol)
- **routing statico**

### 6.7.2.1 BGP

**BGP** è attualmente alla versione 4, utilizza algoritmi di tipo *path vector* inviando alla destinazione la sequenza di *Autonomous System* attraversati. È ricco di attributi ed è possibile configurare la *route computation policy*.

Il *vector exchange* avviene su *TCP* (per maggiore affidabilità) solo a seguito di un cambiamento. Vengono create delle sessioni tra *vicini* per lo scambio di informazioni mediante una configurazione specifica, senza la necessità per la connettività diretta.



**Figura 6.18:** BGP

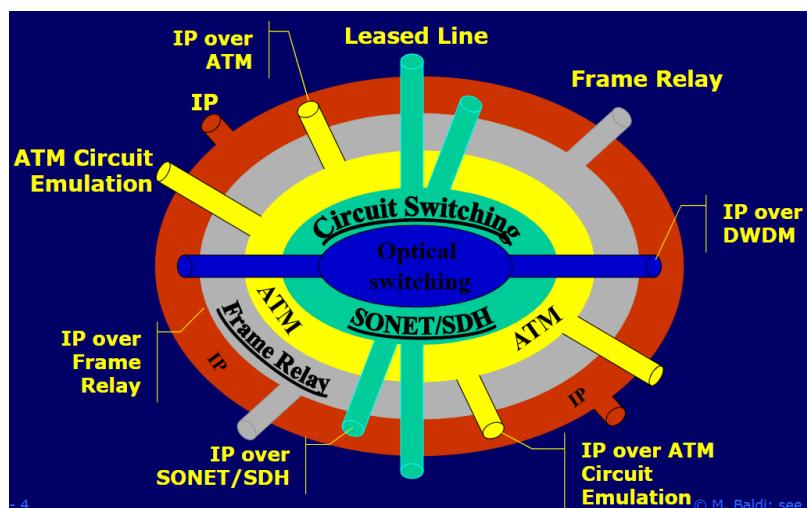
### 6.7.2.2 Inter Domain Routing Protocol (IDRP)

L'algoritmo **IDRP**, *Inter Domain Routing Protocol*, utilizza TCP/IP e rappresenta un'evoluzione di *BGP* per OSI. Doveva essere "la" soluzione per IPv6, ma nel concreto non è molto utilizzato.

## 7 MPLS

**MPLS** è una tecnologia importante in quanto permette la realizzazione di un nuovo tipo di rete pubblica basata su IP, dove con rete pubblica si intende una rete con traffico di diversi utenti e aziende su cui è possibile vendere dei servizi.

Una struttura utilizzata molto in passato era a “cipolla”, con vari strati di livelli protocollari che comunicano tra di loro per implementare varie funzionalità. Ciò comportava però una conoscenza orizzontale da parte dei tecnici su più tecnologie che dovevano comunicare tra di loro.

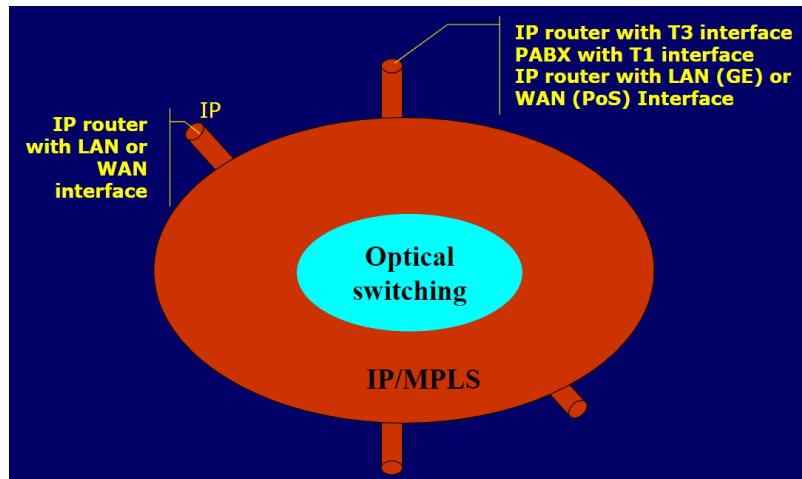


**Figura 7.1:** Struttura a cipolla

MPLS consente di eliminare questa struttura utilizzando un solo livello protocollare, abbattendo i costi degli operatori.

L'inoltro dei pacchetti avviene attraverso l'aggiunta di una **etichetta**, in base alla quale il routing effettua il forwarding invece di guardare l'indirizzo IP di destinazione e consentendo l'invio verso multiple destinazioni usando le etichette come indice. Il motivo di questo approccio risiede nella **maggior rapidità**: se utilizzassimo l'indirizzo di destinazione bisognerebbe eseguire il *longest prefix matching* cercando il prefisso più lungo nel quale l'indirizzo IP di destinazione è contenuto (su tabelle molto grandi).

Oggi è ancora molto utilizzata per il *traffic engineering*, ovvero la distribuzione del traffico nella rete.



**Figura 7.2:** La promessa di MPLS



**Figura 7.3:** Etichetta

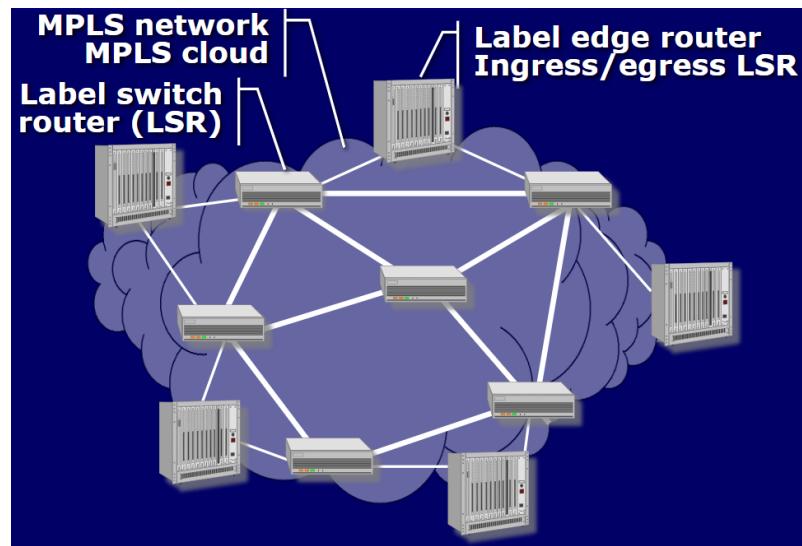
Quello che fa MPLS è dunque far diventare IP *connection oriented*. Lo svantaggio di tale approccio è la necessità di creare una connessione per la comunicazione per poi eliminarla, ma aver implementato IP in modo connection-less ha però generato dei problemi più grandi. La rete continua ad essere a *commutazione di pacchetto*, ma semplicemente esiste un concetto di circuito virtuale che non occupa risorse quando non utilizzato.

## 7.1 Architettura di rete

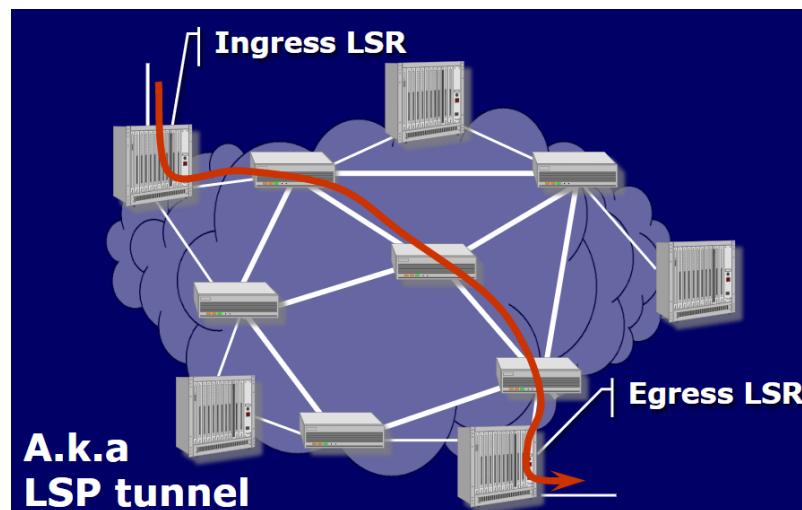
MPLS non utilizza gli *end system* e può essere utilizzato in una porzione di una rete, denominata **MPLS Cloud** (non ha correlazione con il cloud computing).

Osservando l'immagine si può vedere:

- **LSR:** *Label Switching Router*, eseguono la *commutazione a pacchetto* basata su etichetta.
- **Label Edge Router:** router che non ha altri router MPLS collegati, posti al bordo della rete. Compiono l'aggiunta dell'etichetta quando un nodo entra nella rete e la rimuovono prima che esca.
- **LSP:** *Label Switch Path*, percorso di comunicazione attraverso cui viaggiano i pacchetti, in modo che più stazioni possano comunicare tra di loro.

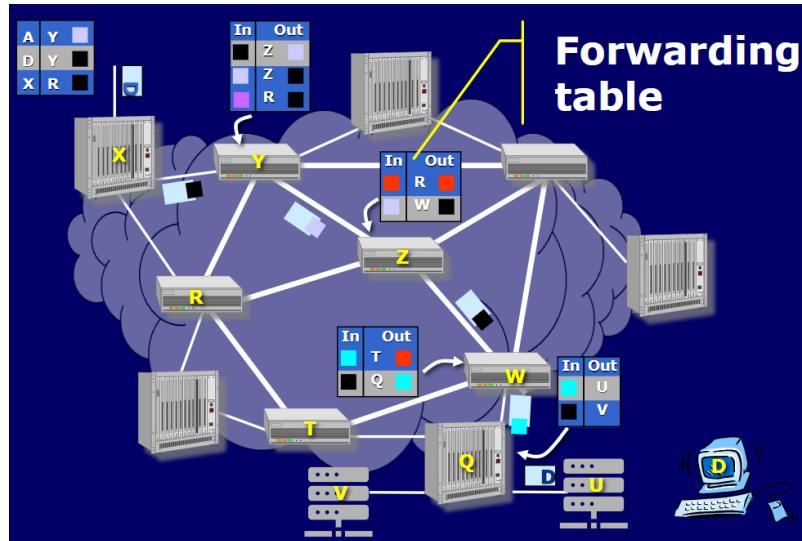


**Figura 7.4:** Architettura di rete



**Figura 7.5:** LSP tunnel

L'etichetta viene cambiata a ogni nodo, in modo da mantenere una etichetta più corta e poterla riutilizzare senza dover mettere d'accordo i nodi (in quanto in caso contrario dovrebbero essere uniche nella rete, in questo modo uniche nel nodo). Questa tecnica prende il nome di **label switching** e consente di ottenere scalabilità.



**Figura 7.6:** Label Switching

Gli elementi chiave di MPLS sono:

- **header MPLS**, contiene l'etichetta.
- **protocolli per la distribuzione** delle etichette
- **protocolli di routing** migliorati e modificati

**Riassumendo:** Il vantaggio è quello di utilizzare un solo protocollo per la gestione delle comunicazioni tra i nodi e anche verso l'esterno.

## 7.2 Storia di MPLS

A differenza di **IPv6**, **MPLS** è stato utilizzato da subito in produzione riuscendo a risolvere problemi di attori molto differenti.

Inizialmente venne implementato il *tag switching* da parte di Cisco nei propri sistemi per sostituire il *longest prefix matching*.

Fino a qualche anno fa si ipotizzava che lo standard di trasmissione **ATM** (*Asynchronous Transfer Mode*) avrebbe soppiantato internet in quanto molto superiore, ma ha come problematica il costo troppo ele-

vato per la struttura (nessun problema di risoluzione indirizzi, signaling semplificato e un solo piano di controllo). Una prima soluzione fu quello di utilizzare ATM con IP, riutilizzando l'hardware del *ATM switching*. Venne successivamente introdotto **MPLS** (*lambda!*), acronimo di *Multi-Protocol Lambda Switching*.

Ha seguito **G-MPLS** (*Generalized MPLS*), il quale ha introdotto:

- Packet switching
- Cell switching
- Circuit switching (SONET/SDH)
- Lambda switching
- Anything switching
- Unifying control plane

### 7.3 Header MPLS

L'header **MPLS** è di livello 2 ed è composta da più moduli uniti che per tale motivo viene chiamato anche *shim header*. Ogni modulo è composto da:

- **label**: 20 bit, l'etichetta da trasmettere. La lunghezza consente il passaggio fino a un milione di flussi sullo stesso link.
- **exp**: experimental bits, 3 bit. Utilizzati per il *Class Of Service*, ovvero per dividere in classi di servizio.
- **S**: *bottom of stack*, 1 bit, è posto a 1 se è l'ultimo modulo dello stack, 0 altrimenti. I moduli vengono gestiti come uno stack e l'aggiunta/rimozione avviene sempre dalla testa.
- **TTL**: *time to live*, 8 bit.

Nel caso di *ATM* e *frame relay* veniva utilizzato il *label switch*, dunque negli header di livello 2 erano già presenti dei campi per contenere l'etichetta. Per tale motivo si è scelto di riutilizzare tali campi invece di aggiungerne di nuovi:

- *VIC/VPI* in *ATM*
- *DLCI* in *frame relay*

In questi casi non si guarderà il modulo MPLS ma i suddetti campi dei moduli già presenti. In questo modo il costruttore di apparati *ATM* non deve cambiare l'hardware ma bensì solamente il software, contribuendo a migliorare lo standard. Questo fa in modo che degli switch *ATM/Frame Relay* diventino degli switch *MPLS*.

## 7.4 LSP setup - selezione del path e delle etichette

Una **FEC**, *Forwarding Equivalence Class*, è un insieme di pacchetti che hanno lo **stesso destinatario**, dunque un *LSP* è un percorso di comunicazione che viene utilizzato per trasportare un *FEC*. Tali pacchetti sono trattati nello stesso modo da ciascun *LSR* e seguono la stessa strada in una rete *MPLS*, ricevendo la medesima label.

Quando viene creato un *LSP*, sono necessarie tre operazioni da parte degli *LSR*:

- **label binding**: associazione di una etichetta a dei pacchetti di una *FEC*.
- **label mapping**: creazione della riga nella tabella di forwarding, tra ingresso e uscita.
- **label distribution**: l'etichetta scelta deve essere comunicata a uno o più nodi vicini.

### 7.4.1 Label Binding

Nel **Label Binding** un *LSR* determina l'etichetta che deve essere utilizzata per i pacchetti di una determinata *FEC*. Il *binding* viene effettuato in modalità **downstream**, ovvero tra due nodi che si trovano ai capi di un link il *binding* è eseguito da quello a valle.

Per sapere di dover usare tale etichetta, l'*LSR* (e in particolare l'*upstream node*, quello a monte) deve essere notificato, e può farlo in due modi:

- **unsolicited**: senza una richiesta diretta.
- **on-demand**: in seguito a una richiesta.

### 7.4.2 Label Mapping

Il **label mapping** esegue l'associazione tra una etichetta di ingresso, scelta dal *LSR* considerato, e una etichetta di uscita, scelta dal *downstream LSR*, per riuscire a raggiungere il next hop in base al routing.

### 7.4.3 Label Distribution

Quando un router ha effettuato il binding di una etichetta, deve comunicare tale etichetta ai nodi vicini, in modo che questi possano fare il *mapping* (almeno al nodo di upstream). Tale operazione è detta **label distribution** e serve a notificare l'etichetta scelta per una data *FEC*, in seguito al *label binding*.

#### 7.4.4 Label Binding statico (e mapping)

Il **label binding statico** avviene attraverso un gestore di rete (in modo equivalente al PVC in ATM) che stabilisce e impone l'etichetta ai nodi della rete.

Non è scalabile e non è c'è interoperabilità tra i sistemi di controllo. Inoltre, è impossibile avere un LSP che attraversa più operatori.

#### 7.4.5 Label Binding dinamico

Il **label binding dinamico** può essere scatenato in due modi:

- **data/traffic driven**: innescato dall'arrivo di un pacchetto.
- **control driven**: ovvero innescato dai messaggi di controllo che può essere di segnalazione o di routing.

##### 7.4.5.1 Control Driven Label Binding

La creazione degli **LSP control driven** da origine a due tipi di LSP diversi:

- **topology based**: il router scopre che esiste una destinazione, in base alla topologia della rete, dei percorsi e delle destinazioni dunque gli *LSR* (*Label Switching Router*) creano degli *LSP* (*Label Switch Path*) per le destinazioni.
- **creazione esplicita degli LSP**: avviene una segnalazione esplicita, inizializzata dai *LER* (*Label Edge router*). Avviene *on-demand*.

#### 7.4.6 Protocolli per la Label Distribution

La distribuzione delle etichette avviene attraverso dei protocolli, in particolare ne esistono 3 (non compatibili tra di loro):

- **BGP**: utilizzo di un protocollo di routing, solo *topology based* (quando vengono segnalate le destinazioni vengono mandate anche le etichette).
- **LDP**: *Label Distribution Protocol*, è un'evoluzione del *Tag Labelling* di Cisco, attualmente deprecato. Poco utilizzato perché, essendo un sistema proprietario, si aveva paura di avvantaggiare Cisco.
- **RSVP**: *Resource reSerVation Protocol*, utilizzato per l'allocazione di servizi integrati all'interno delle reti.

## 7.5 Protocolli di routing

I **protocolli di routing** servono per determinare il percorso che sarà compiuto da un *LSP*, attraverso i quali verranno realizzate le tabelle per il *label mapping* e verrà deciso il next hop mediante il *packet routing*.

I protocolli di routing utilizzati sono in realtà quelli già esistenti:

- *OSPF*
- *IS-IS*
- *BGP-4*

RIP non viene utilizzato perché non funziona bene su reti di grandi dimensioni.

Tali protocolli **vengono modificati** per trasportare informazioni riguardo alle scelte di routing, oltre a quelle topologiche, e porre dei **vincoli** come:

- capacità dei link
- utilizzo dei link
- dipendenze tra i link (utilizzato per il recupero dei guasti)

Tali vincoli verranno utilizzati per eseguire il **Constraint based routing**, ovvero un protocollo di routing di controllo che permette di scegliere il percorso più adatto in base ai vincoli imposti. Per fare ciò vengono utilizzate delle versioni modificate per il *Traffic Engineering* che prendono il nome di *OSPF-TE* ed *IS-IS-TE*, dove *TE* è un acronimo per *Traffic Engineering*. *BGP* non viene usato per fare *constraint based routing*.

Le migliorie apportate a *OSPF-TE* e *IS-IS-TE* **non** sono dunque state introdotte per separare il piano di controllo da quello dati, in quanto non vi è alcuna correlazione tra i due fatti. Le migliori riguardano esclusivamente la possibilità di fare **constraint based routing**.

### 7.5.1 Modalità di Routing

Le **modalità di routing** sono due:

- *hop by hop routing*: soluzione classica.
- *explicit routing*: utilizzo di informazioni che variano in modo dinamico.

### 7.5.1.1 Hop by hop routing

Nel **Hop by hop routing** ciascun *LSR* decide il prossimo *hop*, dunque un *LSR*, del percorso *LSP*. Il principio è lo stesso del *IP routing* tradizionale.

La procedura avviene nei seguenti passi:

- viene scelta una label per l'upstream link (label binding)
- la label viene mappata all'indirizzo della interfaccia del prossimo LSR (next hop)
- la label viene annunciata dal LSR che segue

### 7.5.1.2 Explicit constraint based routing

Nel **Explicit constraint based routing** un singolo switch sceglie il percorso per l'intera *LSP*, oltre al *FEC*. Il percorso potrebbe non essere ottimale, ma almeno si evita il rischio di fare percorsi circolari (il nodo deve avere le informazioni su tutta la rete). Viene poi condiviso il percorso esplicito da compiere.

La scelta del percorso avviene mediante **Constraint Based Routing** (mediante vincoli), ma la distribuzione delle operazioni tra nodi è impossibile in quanto non c'è un unico criterio per scegliere il percorso e possono esserci vincoli in conflitto. Inoltre potrebbe essere difficoltoso mantenere i vincoli e le informazioni sincronizzate, in quanto variano più velocemente delle informazioni relative alla topologia.

Per tale motivo i protocolli per la distribuzione delle etichette sono modificati in modo da supportare informazioni su quale è il percorso scelto.

In particolare sono utilizzati:

- **CR-LDP**: *Constraint based Routing Label Distribution Protocol*
- **RSVP-TE**: *Resource Reservation Protocol Traffic Engineering*

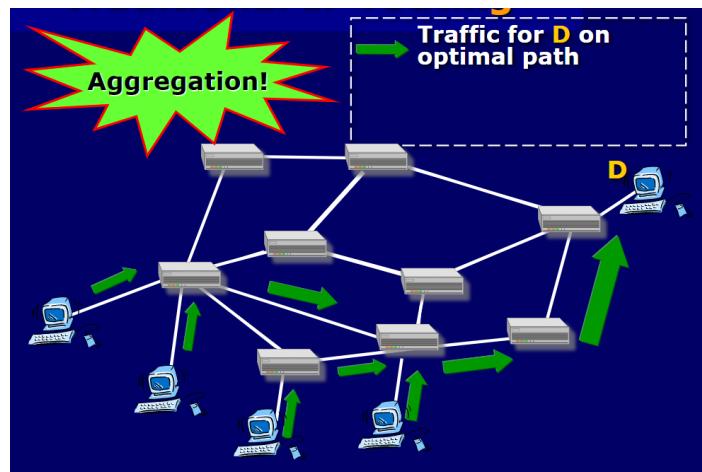
Questi vengono utilizzati con *OSPF-TE* e *IS-IS-TE*.

Le modifiche adoperate permettono:

- **Traffic Engineering**
- garantire la **qualità del servizio (QoS)**
- *per-class traffic engineering* (in sinergia con DiffServ)
- recupero dai guasti rapido, meno di 50 ms

## 7.6 Traffic Engineering

I pacchetti, quando spediti mediante IP, vengono inviati verso le destinazioni realizzando quello che è un fenomeno a “imbuto”, comportando aggregazione che sfocia in una riduzione delle prestazioni. Una soluzione potrebbe essere comprare nuovi router, ma questi diverrebbero inutili al termine dell’intasamento.



**Figura 7.7:** Aggregazione delle informazioni

Il traffic engineering si pone come soluzione per consentire la ridistribuzione del traffico **non in base alla destinazione**, ma in modo **omogeneo** evitando la congestione.

Se si scegliesse di inviare pacchetti in modo tradizionale mediante IP in accordo al carico di ogni link, ogni volta che il router ricalcola i percorsi e i next-hop sono cambiati viene aggiornata la tabella con i nuovi percorsi di rete. Questo causa un inversione di tendenza tra i carichi che iniziano a cambiare molto velocemente causando **instabilità**.

In MPLS la situazione è diversa perché l’inoltro basato sulle etichette permette di separare il *piano di controllo* dal *piano dati*. Nel *piano di controllo* i router raccolgono le informazioni di routing e calcolano le *routing table*. I pacchetti, però, non vengono inoltrati in base al contenuto delle *routing table*, ma in base alle *forwarding table* che si sono create facendo il mapping (realizzate durante la distribuzione delle etichette nel setup del *LSP*). Anche se la *routing table* viene aggiornata, la *forwarding table* non cambia. Perciò il traffico inviato inizialmente, anche se la *routing table* viene aggiornata, continua ad andare sul percorso creato in precedenza. Il nuovo traffico, invece, seguirà il percorso aggiornato. Ciò è possibile perché il piano di controllo lavora sugli indirizzi IP, mentre il piano dati lavora sulle etichette.

In MPLS non c’è un aggiornamento costante tra piano di controllo e piano dati, a differenza di IP, consentendo il *traffic engineering*. Senza MPLS l’alternativa era ATM con due *control plans* (i router sono

ATM-unaware), comportando però una ridotta scalabilità e un alto numero di adiacenze.

MPLS è IP-aware, è presente solo un *control plan* operativo su una topologia fisica, rendendo il tutto più scalabile e semplice.

MPLS vede alcune estensioni come:

- **MPλS**, ovvero MPLS *control plans* su rete ottica.
- **GMPLS**, ovvero *Generalized MPLS*, estensione di MPLS per supportare più tipi di rete (pacchetti, circuito, optics, etc).

## 7.7 CoS e QoS

Le risorse e le modalità di servizio potrebbero essere associate a un *FEC* alla creazione di un *LSP*. E' richiesto un supporto esplicito nel data plan e control plan del LSR.

### 7.7.1 Class of Service (CoS)

La **CoS** (*Class of Service*) è un insieme di parametri che descrivono il servizio richiesto. Consente una priorità relativa tra *FEC* differenti ed è in grado di fornire un garanzia assoluta.

Supporta il modello *DiffServ* con un comportamento *per-hop*, *EF* (expedit forwarding) e *AF* (assured forwarding), oltre al *class traffic engineering* (ds-aware traffic engineering).

### 7.7.2 Quality of Service (QoS)

La **QoS** (*Quality of Service*) garantisce specificatamente:

- bandwidth
- Delay
- burst size

I vantaggi di *QoS* in *MPLS* sono vari, tra questi vi è la possibilità di avere una rete unificata in grado di supportare tutti i tipi di servizi (messaggio di marketing).

Il supporto per *QoS* e i servizi real time su IP non è ancora pronto.

Molte reti multi servizio utilizzano ora un paradigma "ships in the night", dove i protocolli ATM sono per servizi tipici di ATM ed MPLS control plan è utilizzato per i servizi IP.

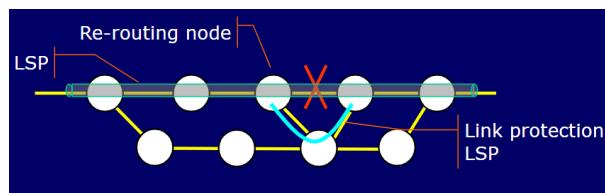
## 7.8 Fast fault recovery

In *MPLS* è garantito il recupero rapido dai guasti mediante link re-routing e edge-to-edge re-routing.

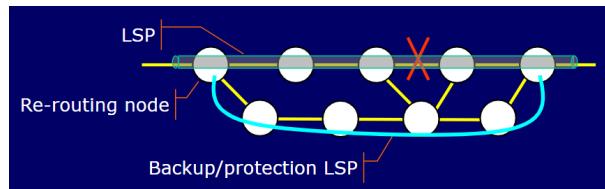
Una volta creato un *LSP*, se un link si rompe il protocollo di routing se ne accorge e ricalcola la *route*, ma il piano dati continua a inviare pacchetti in base alla tabella di forwarding che non cambia. Per risolvere a tale problema, si può allora creare un ulteriore *LSP* che funge da **backup** del link, fatto non quando il link si rompe ma a priori (e utilizzato al momento della rottura).

Quando due nodi si scoprano, si crea un *LSP* lungo la rete che permettano a questi due nodi di inviarsi pacchetti. Se il link diretto si rompe, sarà presente un *LSP* che permetterà di andare verso Y.

Quando il link si rompe, verrà aggiunta un'ulteriore etichetta a quella già presente. Y saprà che quando arrivano con una certa etichetta (cioè sono per Y) dovrà rimuovere l'etichetta più esterna e procedere normalmente. Tale processo è molto veloce ed è detto *link re-routing*.



**Figura 7.8:** Link re-routing



**Figura 7.9:** Edge to edge re-routing

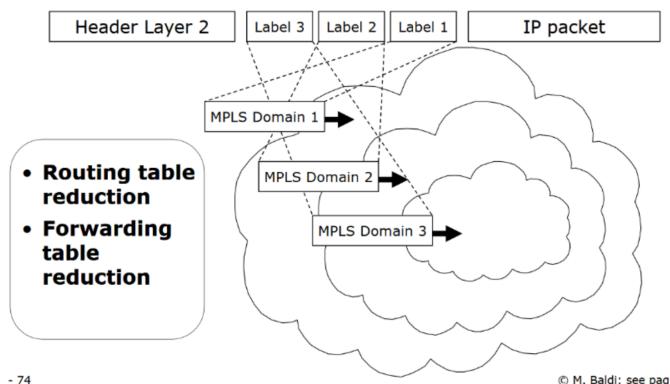
<

## 7.9 Gerarchia e scalabilità

Utilizzare più etichette aiuta a generare una gerarchia tra reti *MPLS* diverse che contribuisce alla scalabilità della rete. Utile soprattutto per instradare pacchetti in punti geografici molto lontani, quando un ISP non ha connettività diretta verso tale destinazione. Ciò comporta una riduzione delle routing table e delle forwarding table.

Le etichette *MPLS* introducono gerarchie su più livelli, a seconda di quanto richiesto per la scalabilità. Le tabelle di routing dei router di transito non devono essere necessariamente complete, in quanto LSP è gestito tra gli *edge router*.

In questo modo è più semplice e veloce gestire il match delle etichette piuttosto che il *longest prefix matching*.



**Figura 7.10:** Utilizzo di più etichette

## 7.10 Penultimate Hop Popping (PHP)

Nel **Penultimate Hop Popping (PHP)**, il penultimo nodo esegue il pop della label dal *LSP*, in modo da non doverlo fare il nodo di destinazione. Il **Label Edge Router (LER)** indirizza il pacchetto in base all'indirizzo IP (o la prossima label nello stack).

La distribuzione di label 3 indica un implicito PHP, in quanto l'*edge router* vede che il *next hop* è all'esterno.

Per qualsiasi router sull'ultimo hop avviene lo swap sull'etichetta 0.

Il PHP si divide in:

- **PHP隐式 (PHP implicito):** l'etichetta più esterna viene rimossa
- **PHP显式 (PHP esplicito):** l'etichetta non viene rimossa, ma ci si scrive “0” all'interno

## 7.11 MPLS VPN

Uno dei problemi fondamentali che le *VPN* devono risolvere nella connessione di due reti private è quello di utilizzare **reti private**. Più aziende diverse possono utilizzare lo stesso range di indirizzi privati, perciò i pacchetti che viaggiano nell'infrastruttura di rete (backbone) non possono contenere

tal IP destinazione. Un'alternativa sono i NAT oppure i tunnel. Gli *LSP* in realtà sono dei tunnel, dato che gli *LSR* non guardano l'indirizzo IP destinazione ma l'etichetta, perciò basta utilizzare le etichette per indirizzare i pacchetti a destinazioni diverse.

### 7.11.1 PWE3

**PWE3**, ovvero e **Pseudo Wire Emulation End-to-End**, consente l'utilizzo di VPN basate su *MPLS* al **livello 2** andando a creare *LSP* tra tutti i *router di confine* (BR) della rete *MPLS* che devono fornire connettività aziendale. Il nome deriva dall'idea di unire due aziende attraverso un cavo *virtuale* per l'inoltro dei pacchetti.

Esistono molti servizi nella stessa rete come IP, leased lines, frame relay, ATM ed ethernet.

Per il funzionamento vengono utilizzate due etichette:

- **esterne**: per il routing nella rete, identificazione dell'access point alla rete.
- **interne**: per il multiplexing di molti utenti/servizi sullo stesso access point.

### 7.11.2 MPLS-based Layer 3 VPNs

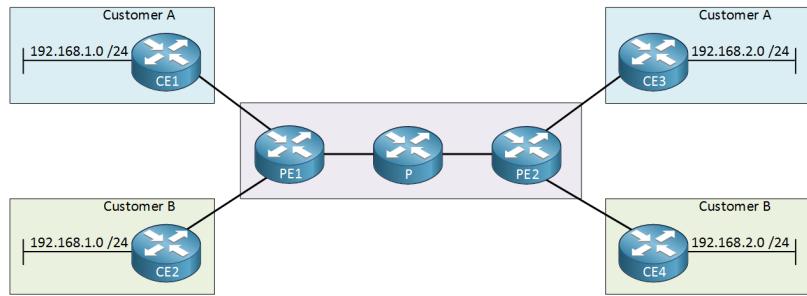
Esistono alternative di **livello 3**, ovvero le **MPLS-based Layer 3 VPNs**, che consentono di creare gli *LSP* automaticamente tra i vari *PE* e poi crearne ulteriori all'interno della rete per il collegamento delle reti private (connesse a tali *PE*). Questa soluzione funziona solo per il trasporto di pacchetti *IP*.

Per rendere tutto automatico è necessario scambiare informazioni di routing. I *CE* scambiano informazioni con i router *PE*: il *CE* comunica le destinazioni che si ritrova, successivamente i *PE*, che sono a conoscenza dell'esistenza di altri *PE* (per via di un protocollo di routing), creano degli *LSP* tra di loro (ad esempio nella *topology-based label binding*). A questo punto, quando i *PE* hanno degli *LSP* tra loro li utilizzano per scambiare informazioni di routing tra loro.

Queste soluzioni sono di tipo Provider Provisioned, in quanto le policies sono implementate dal Service Provider, ma con il vantaggio di non richiedere alcun tipo di esperienza da parte dell'utente. Sono molto scalabili e distribuibili.

Sono disponibili due alternative:

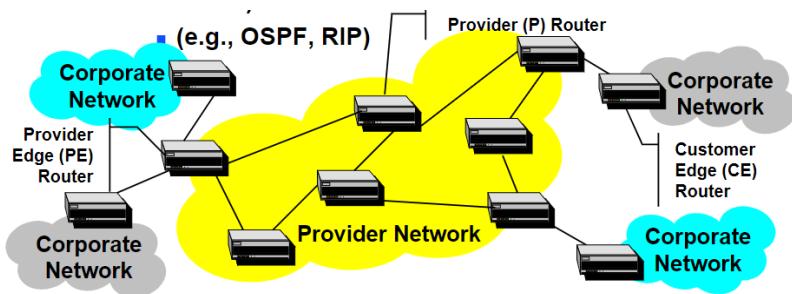
- **BGP**: inizialmente supportato da Cisco, attualmente il sistema più utilizzato.
- **Virtual Router**: inizialmente supportato da Nortel e Lucent.

**Figura 7.11:** Rappresentazione di CE e PE

### 7.11.3 Componenti

Le componenti principali di un *MPLS VPN* sono:

- **CE router:** crea collegamenti con *PE router*. Fa *advertise* verso le destinazioni e riceve gli *advertisements* da altre destinazioni VPN. Utilizza *static routing* o *IGP* (Interior Gateway Protocol).
- **P routers:** hanno *route* solo verso *PE router*. Si occupano di eseguire il setup degli *LSP*.
- **PE routers:** sono utilizzati per scambiare le informazioni di routing. Viene utilizzato *I-BGP* (Interior-Border Gateway Protocol) in soluzioni basate su BGP, oppure IGP in soluzioni VR. Vengono memorizzate solo le routes per le VPN connesse a questo. I PE hanno diverse tabelle denominate *VRF* (*VPN Routing and Forwarding Table*) che consentono di creare delle tabelle di routing separate. Viene effettuato il forwarding delle informazioni utilizzate per il traffico ricevuto mediante la porta.

**Figura 7.12:** Componenti

### 7.11.4 Benefici

Un beneficio è la non necessità della cifratura, in quanto il traffico va sulla rete del service provider (di cui ti fidi) e dunque non passa da router di terze parti non affidabili.

- **Non ci sono vincoli sugli indirizzi** (si possono usare gli indirizzi privati) poiché non viene visto l'indirizzo sul backbone. Oltre all'indirizzo si specifica la VPN a cui appartengono (ed è il motivo per cui si usa BGP), vedi prossimo paragrafo.
- I **CE** non scambiano direttamente informazioni tra loro.
- I customer non gestiscono l'infrastruttura (backbone).
- Il provider non ha un backbone virtuale per cliente.
- La VPN può passare tra provider diversi.
- La sicurezza si basa sulla fiducia verso il service provider.

### 7.11.5 MPLS VPN basate su BGP

Le implementazioni di *MPLS VPN* che utilizzano **BGP** hanno lo scambio di informazioni di routing tra gli edge. Compiono *Router Filtering* ovvero ciascun PE determina quali route installare nella VRF. Supporta la sovrapposizione di spazi di indirizzi differenti.

Oltre all'indirizzo si specifica la VPN a cui appartengono (ed è il motivo per cui si usa BGP). Il BGP è un protocollo che può facilmente essere esteso e in questo caso si chiama MP-BGP (Multi-Protocol GBP) poiché permette di trasportare informazioni su indirizzi che non sono indirizzi IP.

Il BGP si definisce una famiglia di indirizzi VPN particolare definita VPN-IPv4. Oltre ad avere gli indirizzi IP da 32 bit vengono definiti anche i Route Distinguisher.



**Figura 7.13:** Route Distinguisher

### 7.11.6 MPLS Virtual Router VPNS

Esiste un'altra soluzione senza l'uso di BGP per scambiare le informazioni di routing. I PE funzionano non come singolo router ma come tanti router diversi. Ogni router virtuale eseguirà indipendentemente le sue funzioni, come se fosse fisicamente un unico router. I router virtuali della stessa rete privata creano degli LSP tra loro e poi useranno gli LSP per scambiarsi le informazioni.

Le informazioni verranno comunicate solo ai router della stessa rete privata e non più a tutti.

## 7.12 6PE

Utilizzo di protocolli IPv4. I PE eseguono un istanza virtuale del router per ciascuna VPN. Ciascun VR instance ha una struttura dati separati. I VR di uno stesso VPN comunicano attraverso gli LSP. le destinazioni sono IPv4.



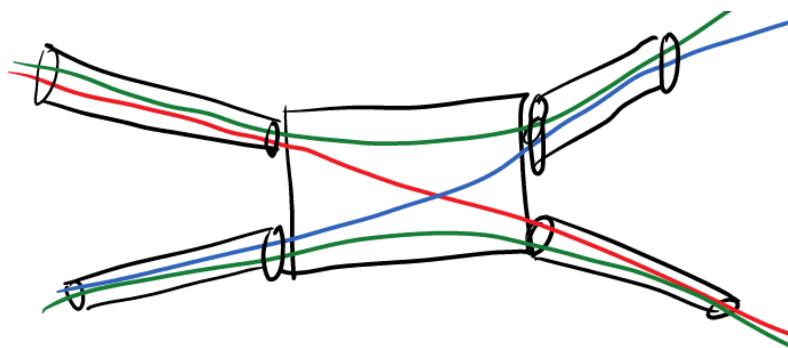
## 8 Rete Ottica

Con rete ottica non si fa riferimento al trasferimento su un cavo di fibra ottica, ma bensì alla **comunicazione ottica**.

Il concetto principale dietro alle reti ottiche è il **Wavelength Division Multiplexing (WDM)**, ovvero la possibilità di trasmissione di frequenze diverse sullo stesso mezzo. Si divide in due tipi:

- **DWDM**: *Dense WDM*, consente la trasmissione di decine o centinaia di segnali sulla stessa fibra, consentendo l'aumento della capacità sui cavi esistenti.
- **CWDM**: *Coarse WDM*, usa più finestre (frequenza diverse) per la trasmissione. Ha un minor numero di lunghezze d'onda, ma è più economico.

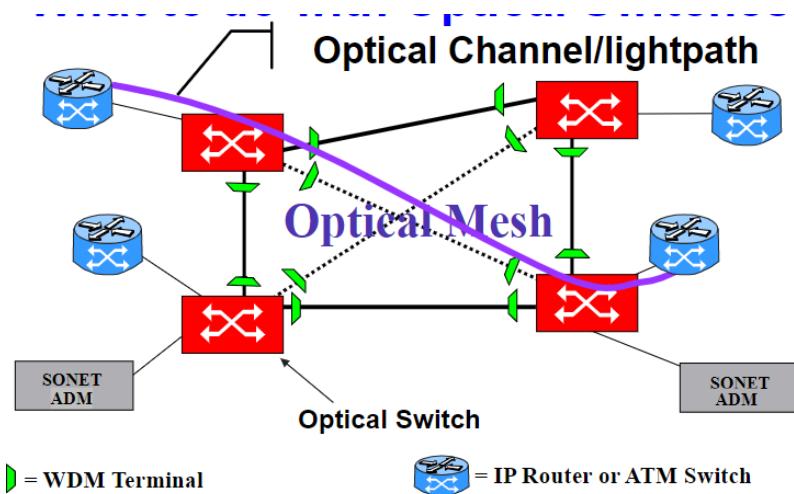
Inizialmente la ragione per cui è stato realizzato *WDM* era per poter riutilizzare la stessa fibra. Solo successivamente si è notato che avendo tanti segnali ottici è possibile costruire un commutatore che riconosce i vari canali e li inoltra su fibre differenti, ovvero un **wavelength switch** (commutatore ottico). L'idea alla base delle reti ottiche è di fare reti in cui i nodi sono collegati da fibre e sono in grado di **commutare canali ottici da una fibra di ingresso a una fibra di uscita**.



**Figura 8.1:** Wavelength Switching

Le reti ottiche si utilizzano al centro delle reti in quanto consentono di commutare un canale ottico. Inoltre, il commutatore ottico ha le potenzialità di essere **molto semplice**, infatti la tecnologia del DWDM ha permesso di abbassare i costi e di aumentare tanto le prestazioni.

E' possibile aggiungere o rimuovere multiplexing, in particolare nelle topologie ad anello. In questo caso prendono il nome di *Optical Add-Drop Multiplexer (OADM)*, consentendo di inerire wavelengths



**Figura 8.2:** Switch Ottici

nell'anello e di estrarre. Per lo più sono configurazioni statiche o semipermanenti che interconnettono le configurazioni (*Reconfigurable OADM*).

## 8.1 Switching Core

Esistono differenti tipologie di *Optical Switch*, che consentono differenti livelli di complessità e di flessibilità, che si differenziano in:

- ottici o elettronici
- cross connect o switch
- wavelength conversion

### 8.1.1 Optical Core

Un **Optical Core** utilizza le proprietà fisiche di alcuni materiali per deflettere la luce da una fibra di ingresso verso una fibra di uscita.

Esistono più tipologie:

- **Tilting mirrors:** *Micro-Electro-Mechanical System* (MEMS), sono apparati meccanici sensibili agli eventi esterni (ad esempio un treno in corsa che genera vibrazioni).
- **Superfici riflettenti olografiche:** funzionano attraverso il voltaggio.
- **Materiali che cambiano le proprietà riflettenti** in base a calore, pressione e voltaggio/corrente.

Le proprietà dei componenti ottici sono le seguenti:

- **potenzialmente poco costosi**, ma non ancora economici.
- **Bit rate e segnale indipendenti**, consentendo alta scalabilità e multi standard.
- **Basso consumo di energia**.

Gli aspetti invece negativi sono **alti prezzi** di produzione (attualmente) e un **alta attenuazione** in quanto non vi è rigenerazione del segnale.

I **MEMS** sono quelli che hanno ottenuto più successo.

### 8.1.2 Electronic Core

Gli **Electronic Core** sono dispositivi elettronici che convertono un segnale ottico in uno elettrico verso un circuito elettronico effettuando lo switch dei bit ricevuti, a prezzi minori rispetto ai **MEMS**.

Purtroppo, si perdono tutte le proprietà dei componenti ottici, in quanto **non si ha indipendenza** del bit rate rispetto al segnale, il consumo non è basso ed i costi sono più sostenuti (anche se, attualmente, più economici). Un grande vantaggio è un migliore rapporto complessità/costi rispetto al packet switching.

## 8.2 Switching Dynamics

### 8.2.1 Cross Connect

Un **cross connect** è un elemento di rete dei sistemi di trasmissione telefonica e dati con la funzione di smistare alte capacità di traffico tra le varie parti di una rete e, per tale motivo, trova impiego essenzialmente nelle dorsali geografiche a livello nazionale o internazionale.

Soltamente vengono adoperati core ottici. Le configurazioni sono *statiche* (fixed), mediante un sistema/interfaccia di configurazione.

### 8.2.2 Fiber Cross Connect

Nel **Fiber Cross Connect** tutti i segnali da una fibra di ingresso vengono trasmessi su una fibra di uscita. Vengono adoperati dei *micro-electro-mechanical system* (MEMS) ma richiedono molto tempo per essere riconfigurati.

A volte viene utilizzata un amplificatore ottico prima e dopo la commutazione (switching).

### 8.2.3 Wavelength Cross Connect

Nel **Wavelength Cross Connect** uno o più wavelength vengono mandate da una fibra di ingresso verso una fibra di uscita. E' presente un WDM de-multiplexer+MEMS a separare le differenti wavelength nello spazio (prism).

La rigenerazione del segnale potrebbe essere utilizzata prima e/o dopo la commutazione (switching), ad esempio la conversione **OEO** (*optical-electrical-optical*) consente di rigenerare elettronicamente, anche se dipende dal bit rate.

## 8.3 Wavelength Conversion

La **Wavelength conversion** è complessa, spesso realizzata mediante **EOE**, e richiede un **costo elevato**. Non è trasparente per ai dati ciò la rende difficilmente scalabile.

Dal punto di vista fisico si comporta come una *cassa di risonanza* (?) mentre dal punto di vista tecnologico non è molto matura, oltre a essere molto costosa.

Non richiede la stessa *wavelength end-to-end* e non è presente il *wavelength assignment problem* ( $N^2$ ).

## 8.4 Combinazioni comuni

### 8.4.1 Wavelength cross-connect con Wavelength Conversion

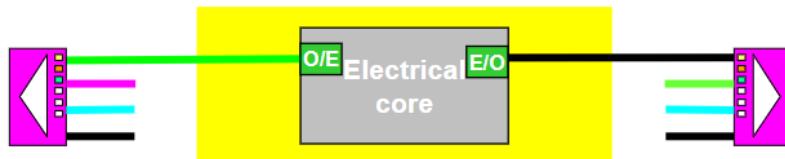
Nella **Wavelength cross-connect con Wavelength Conversion** si ha una o più lunghezze d'onda da una fibra di ingresso verso una o più uscite di fibra.

Potrebbe essere utilizzato un *electrical core* in quanto il monitoraggio del segnale è più semplice. Il *forward error correction* (FEC) del segnale è più semplice e consente di ridurre il rapporto di errore per bit (*BER, da bit error ratio*).

Altrimenti è possibile utilizzare un *optical core* con **OEO** (*optical-electrical-optical*) per la conversione, consentendo anche la rigenerazione del segnale.

### 8.4.2 Dynamic Optical Switching

Nel **Dynamic Optical Switching** può essere presente conversione di lunghezza d'onda, ed è possibile utilizzare un *optical core* con OEO per la rigenerazione e wavelength conversion oppure un *electrical core* con SONET/SDH con la possibilità di inserire multipli OEO per la rigenerazione.



**Figura 8.3:** Struttura

La configurazione dello switch cambia dinamicamente, in base ad alcuni criteri:

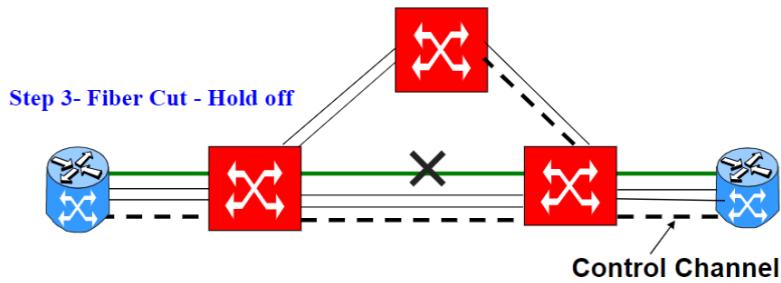
- configurazione
- ora del giorno
- segnalazione dell'end system
- per ogni pacchetto, nel optical packet switching e optical burst switching

Se viene utilizzato un core ottico, nel caso di wavelength conversion è presente anche rigenerazione mediante OEO. Invece nel caso in cui venga adoperato un core elettronico, si adopera SONET/SDH.

## 8.5 Distribuzione

Le reti ottiche consentono di ottenere *provisioning* e *protezione* dai guasti per i lightpaths end-to-end, permettendo di trovare un percorso alternativo. Il client equipment (come i routers) eseguono il controllo del *provisioning* sul piano ottico per i lightpath (signaling).

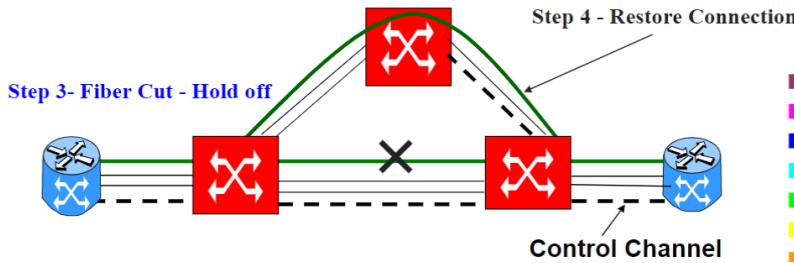
E' possibile ottenere una Distribuzione conveniente e flessibile delle reti.



**Figura 8.4:** Provisioning

## 8.6 Control Plane

Quello di cui hanno bisogno gli switch ottici sono:



**Figura 8.5:** Protezione

- **Resource discovery:** sapere la topologia, quali sono gli access point e poter identificare i nodi, utilizzo delle risorse.
- **Gestione della connessione / signaling:** creazione dei lightpath, *lightpath take down, lightpath modification*.
- **Routing distribuito.**
- **Protezione e recupero** per reti magliate o ad anello.
- Stabilire un **servizio di protezione per le classi**.

Ciò che invece è necessario garantire agli utenti è:

- **resource discovery:** gli indirizzi degli utenti devono essere raggiungibili mediante una rete ottica.
- **gestire lightpath:** setup, take down, modification.
- **negoziare protection service classes:** protected, unprotected, best effort lightpath.

In definitiva, l'unica soluzione per l'Optical Network Control Plane è **MPλS**, Multi-Protocol Lambda Switching, mentre:

- OSPF, IS-IS, BG per il resource discovery
- RSVP/LDP per il signaling

tutti gli aspetti introdotti erano già stati presentati attraverso il protocollo ATM, ma il costo per la tecnologia era troppo elevato.

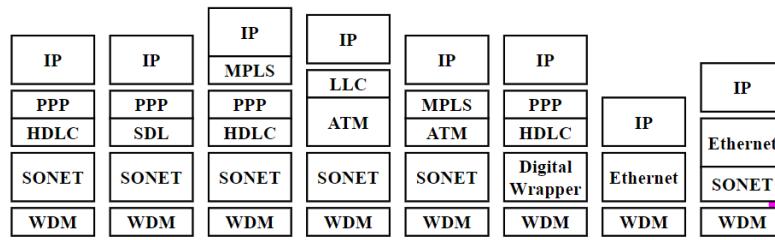
## 8.7 Routing

Nelle reti ottiche gli utenti sono i router, mediante modelli:

- **Overlay:** la rete ottica fornisce connettività attraverso i router, che vedono la rete come una *scatola nera*, a volte forniti con una informazione di reperibilità.

- Peer: i router e gli switch partecipano con gli stessi protocolli di routing. I router conoscono la topologia della rete ottica e possono scegliere dei path preferiti rispetto ad altri per raggiungere destinazioni specifiche.

## 8.8 Data Transport and Protocol Stack



**Figura 8.6:** Incapsulazione



# 9 Quality of Service

Le *applicazioni multimediali* sono molto diverse da quelle tradizionali in quanto potrebbero necessitare di alcuni requisiti di *qualità del servizio*, come ad esempio la latenza, la banda o la perdita di pacchetti. In tali applicazioni il flusso di dati è continuo e il profilo generato deve essere lo stesso di quello ricevuto, in modo che il ricevitore possa riprodurre le informazioni che riceve e mostrarlo all'utente (senza temporizzazione o memorizzazione).

Per questo tipo di applicativi è dunque necessario parlare di **Quality of Service**, ovvero di un meccanismo in grado di garantire una certa qualità del servizio.

## 9.1 Requisiti

I requisiti di rete per garantire il *Quality of Service* sono i seguenti:

- **Streaming**, il flusso di dati deve essere continuo. E' concessa una tolleranza nella perdita del pacchetto, ma è necessario un ritardo che sia al più costante.
- **Interattività**, con una persona o un computer mediante un tempo di risposta basso (sotto i 100 / 150 ms per direzione).
- **Trasmissione su larga banda**: deve essere disponibile una quantità elevata di risorse, capacità elevata e molta memoria nei nodi.
- Comunicazioni di gruppo, ovvero la possibilità di gestire comunicazioni *molti a molti*.

Il vero problema che però deve essere risolto è il **ritardo**. Oltre ai tempi di elaborazione, ai nodi potrebbe esserci congestione, perciò se vi sono molti pacchetti che vogliono uscire tutti dallo stesso link questi non possono uscire tutti insieme, ma vengono inseriti in un buffer. Ciò comporta che i pacchetti attendono e questo si trasforma in un ritardo variabile a seconda del carico del nodo.

**Importante:** Il ritardo di attraversamento dei nodi dipende dal traffico istantaneo, non solo dalla quantità ma anche dalla tipologia del traffico.

## 9.2 Contromisure

Per riuscire a garantire la qualità del servizio sono necessarie molte risorse, una alta capacità trasmisiva, buffer più grandi nei nodi e infine capacità di commutazione (switching).

A livello di pacchetto è necessario applicare delle particolari *policy* o *traffic shaping* (tecnica di gestione della congestione), mentre a livello di flusso è necessario effettuare delle segnalazioni per riservare le risorse attraverso protocolli come **RSVP** (*Resource reSeRVation Protocol*) per IP e *UNI* (*User to Network Interface*) per ATM.

Si vede necessario applicare a priori il *network engineering*, dimensionando la rete in accordo al traffico previsto e limitando il numero di utenti che possono accedere alla rete, e il *traffic engineering* per controllare la distribuzione del traffico sulla rete. Tali azioni hanno natura preventiva per dimensionare il caso peggiore in base alle statistiche sul traffico degli utenti, dunque viene determinata la matrice del traffico in modo da distribuirlo. Lo stato della rete viene costantemente monitorato e, se necessario, ridimensionato.

Le contromisure che si possono realizzare nella rete sono le seguenti:

- **Classificazione del traffico:** si identificano i pacchetti che necessitano di QoS.
- **Algoritmi sofisticati di scheduling:** scegliere quali pacchetti prendere dal buffer per inviarli all'esterno.
- **Controllare il traffico che entra nella rete:** se i pacchetti in arrivo sono molti, non è possibile fare altrimenti se non scartare dei pacchetti, e in tal caso non si può garantire QoS. Ciò può essere fatto a vari livelli, oppure effettuando un routing che tenga conto del QoS.

**In altre parole:** è necessario limitare la quantità di pacchetti che arrivano ai nodi di rete e gestire in modo appropriato i pacchetti che hanno bisogno specifico di QoS.

### 9.2.1 Classificazione

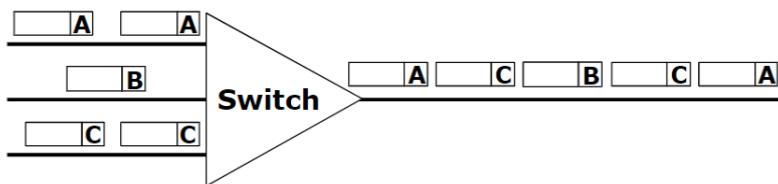
Per effettuare la **classificazione** del traffico è necessario individuare univocamente i pacchetti che appartengono a una determinata comunicazione. Per farlo sono necessari i seguenti campi:

- IP sorgente
- IP destinazione
- protocollo di trasporto
- porta destinazione
- porta sorgente

Per riuscire a classificare tali pacchetti è dunque necessario un componente hardware denominato **ASIC** (*Application Specific Integrated Circuit*) oppure le memorie **CAM** (*Content Addressable Memory*) dove si può dare un indirizzo e invece di ricevere in risposta il contenuto di una cella si può dare una quintupla per avere indietro il tipo di QoS.

### 9.2.2 Scheduling

Per effettuare lo scheduling è possibile in prima approssimazione utilizzare una coda *FIFO*, ma non risolve il problema in quanto l'ultimo pacchetto a uscire sarà sempre l'ultimo a essere entrato, senza imporre alcuna precedenza. Una alternativa è l'utilizzo del **multiplexing statistico**: i pacchetti vengono sequenziati sul link di uscita in modo casuale in base all'ordine di arrivo, ma anche in questo caso non siamo in grado di soddisfare tutte le richieste.



**Figura 9.1:** Multiplexing statistico

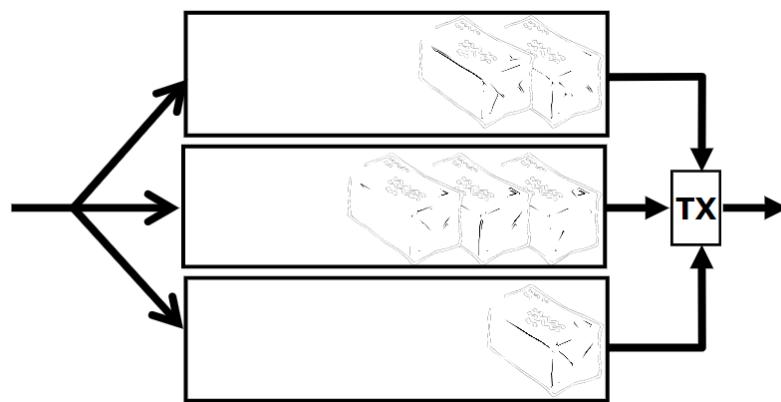
Per garantire la *QoS* è necessario analizzare tutti i pacchetti e inserirli in **code multiple** servite in base alla priorità utilizzando degli algoritmi di scheduling, come:

- *Priority Queuing*
- *Round Robing*
- *Class Based Queueing (CBQ)*
- *Weighted Fair Queueing (WFQ)*: sapendo il tipo di traffico di ogni applicazione è possibile configurare i nodi in modo da rispettare i requisiti.
- *Deadline queuing*: è possibile impostare un deadline per ogni pacchetto, in modo da garantire che il pacchetto venga inviato entro un certo tempo.

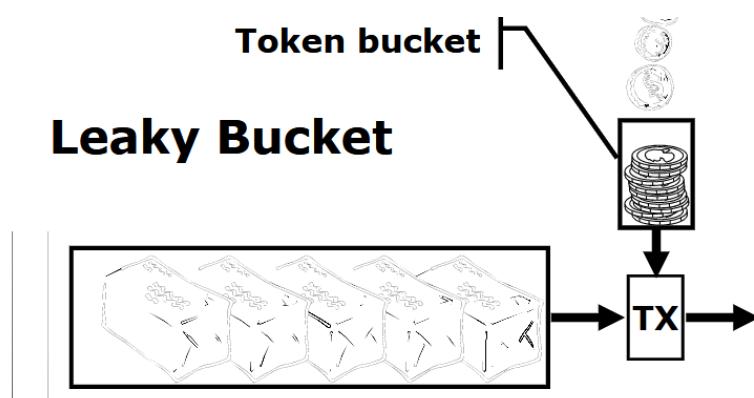
## 9.3 Controllo del traffico

Al fine di garantire un *QoS*, è possibile definire un azione di *policy* e *shaping* per stabilire se il traffico in ingresso nella rete ha il profilo adatto per entrare. In particolare si può adottare la tecnica del **leaky bucket** che permette di controllare il flusso di ingresso.

I pacchetti non conformi vengono rinviati, diminuiti di priorità (best effort) oppure scartati.



**Figura 9.2:** Multiple Queue



**Figura 9.3:** Shaping

### 9.3.1 Call Admission Control

La **Call Admission Control** (CAC) consente di eseguire *signalling* mediante la descrizione del traffico generare e del servizio richiesto ed effettua reservation.

## 9.4 Routing

Le scelte di routing sono prese in base alla disponibilità delle risorse, dunque non solo sulle informazioni inerenti alla topologia.

E' presente instabilità in caso di trasferimento dati *connectionless*.

## 9.5 Frameworks

Esistono due standard per supportare il QoS:

- **IntServ**
- **DiffServ**

### 9.5.1 IntServ

**IntServ** garantisce il QoS, effettua la prenotazione delle risorse (RSVP) riuscendo a garantire QoS a ogni singolo flusso.

Ha come criticità la complessità e la bassa scalabilità (non utilizzabile su scala elevata). Per quanto lo standard sia pronto e implementato nei router, non viene utilizzato.

La soluzione è stata standardizzata in modo da consentire alle applicazioni di richiedere e ricevere dalla rete una qualità del servizio in accordo alle proprie esigenze.

### 9.5.2 DiffServ

**DiffServ** non garantisce il QoS e non consente di riservare le risorse.

Distingue il traffico in classi diverse identificate dal campo *DS Field*. Combinando questa differenziazione, insieme al *network/traffic engineering* e all'accesso controllato, è possibile limitare i pacchetti presenti nel buffer.

Purtroppo soffre di bassa efficienza (*best effort*), ma consente semplicità e scalabilità.

Nell'ultimo periodo è sempre più utilizzato.