



POLITECNICO DI TORINO

Tecnologie e Servizi di Rete
Appunti del corso 02KPNOV del Prof. Guido Marchetto

A.A. 2020/21

Autore: Marco Smorti

Sommario

Ripasso protocollo IPV4	8
Indirizzamento classful	8
Indirizzamento classless	9
IP addressing: metodologia	13
Esercizi	15
IP Multicast	26
Addressing Multicast	27
IP Version 6 (IPV6)	29
Indirizzi IPv6.....	31
Indirizzi Host – Routing e principi di indirizzamento	31
Global Unicast (indirizzi pubblici)	33
Special Addresses	34
Protocolli modificati	34
Formato dell'header.....	35
ICMPv6 – Internet Control Message Protocol version 6	39
Multicast Communication in the Internet	41
Configurazione dei dispositivi IPv6	43
Mapping da EUI-48 a EUI-64.....	43
Prefisso di un indirizzo.....	44
ICMP Redirect	46
Scoped Addresses.....	48
Routing in IPv6	49
Transizione da IPv4 a IPv6	53
Host-centered Solutions	55
IPv4-compatible Addresses	55
IPv4-compatible Address con Router Dual Stack	56
6over4.....	56
ISATAP: Intra-site Automatic Tunnel Addressing Protocol.....	57
Network-centered Solutions	58
6to4	58
Teredo	59
Tunnel Broker	59
Scalable, Carrier-grade Solutions	59
DS-Lite	60

A+P.....	61
MAP	62
Port Set	63
CPE IPv6 Address	63
Mapping Rule.....	63
Border Relay	64
MAP-E	65
MAP-T	65
NAT64 + DNS64.....	67
Soluzioni basate su MPLS	68
Esercizi su IPv6.....	70
Reti cellulari	74
Splitting.....	75
Cell shaping.....	76
Power Control.....	76
Open loop	76
Allocazione della frequenza.....	76
Architettura di rete.....	77
Procedure base.....	77
Registrazione	77
Mobility management	77
1G: First Generation	78
2G: Second generation	78
2.5G: Extended second generation	78
3G: Third generation	78
3.5G: Extended third generation	78
4G: Fourth generation	78
5G: Fifth generation	78
GSM – Global System for Mobile Communications	79
Identificazione del MT.....	80
Aree GSM.....	81
Frequenze GSM in Europa	81
Canali fisici GSM	81
Frame GSM	82
Frequency Hopping (FH).....	82

Ostacoli tecnologici.....	82
Timing advance	82
Ramp up/down	82
Tipi di burst nella trasmissione GSM	83
Struttura del “Regular” burst.....	83
Struttura dell’Access burst.....	83
Interleaving.....	84
GSM framing.....	84
Canali logici GSM	85
Control Channels	85
Network signaling	85
Uso del broadcast control channel	85
User signaling.....	86
Common control channels (CCCH).....	86
Dedicated Control Channels (DCCH).....	86
Traffic channels	87
Mapping dei canali logici sui canali fisici	87
TCH e SACCH	87
FACCH	87
Gli altri canali	87
Procedure GSM.....	88
Chiamata (da telefono fisso) a MT.....	88
Handover	89
Locating.....	89
Tipi di handover.....	89
Handover tra BTS di diverse BSC e diverse MSC.....	89
General Packet Radio Service (GPRS)	90
Enhanced Data rate for Global Evolution (EDGE).....	90
Universal Mobile Telecommunication System (UMTS)	90
UTRAN.....	90
Macrodiversità.....	90
Rispetto a GSM e GPRS.....	91
Core Network.....	91
Procedure per accedere ai servizi UMTS	92
Tecnologie UTRAN	92

Stack protocollare dell'interfaccia radio.....	93
Canali UMTS.....	93
UMTS soft handover	94
Vertical Handover	94
High Speed Packet Access (HSPA)	94
Canali condivisi	94
HSPA+	94
LTE	95
LTE-A.....	96
Core Network 5G	96
Software Defined Networking (SDN)	97
Network Function Virtualization (NFV).....	98
Architettura SDN + NFV (ETSI)	98
Routing.....	99
Gli algoritmi non adattivi.....	100
Algoritmi adattativi.....	100
Routing centralizzato.....	100
Routing isolato.....	101
Routing distribuito.....	101
Algoritmo Distance Vector.....	101
Cold Start	102
Count to infinity.....	103
Bouncing Effect.....	103
Split Horizon.....	104
Path Hold Down.....	105
Route Poisoning.....	105
Path Vector	106
Algoritmo Link State	106
Internet Routing Architecture	109
IGP – Distance Vector	110
IGP – Link State	110
EGP	110
RIP	110
IGRP	110
OSPF.....	111

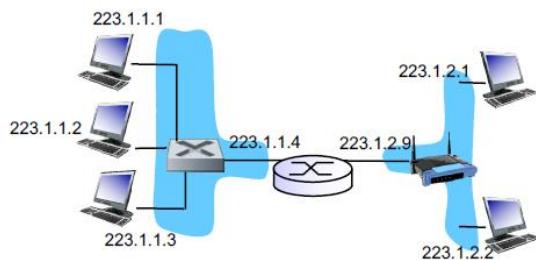
Broadcast Network	111
Integrated IS-IS	112
BGP	112
IDRP	112
VPN	113
Motivazioni.....	113
Accesso a internet	115
Caratteristiche di una VPN	116
Site 2 site VPN Tunneling.....	116
End 2 end VPN Tunneling	116
Remote VPN Tunneling.....	116
Overlay Model.....	117
Peer Model.....	117
Customer Provisioned VPN	117
Provider Provisioned VPN.....	117
Access VPN Customer Provisioned.....	117
Access VPN Provider Provisioned.....	118
Layer 2 VPN	119
Layer 3 VPN	119
Tunneling IP in IP	119
Layer 4 VPN Tunneling: s2s	119
Layer 4 VPN Tunneling: e2e.....	119
GRE – Generic Routing Encapsulation	120
L2TP – Layer 2 Tunneling Protocol	121
Header L2TP.....	122
PPTP Point-to-Point Tunneling Protocol.....	123
Connessioni PPTP.....	124
IPsec.....	124
SSL VPN.....	126
Caratteristiche delle SSL VPN.....	126
VPN Gateway Positioning & Anomalie	127
MPLS – Multi-Protocol Label Switching.....	128
Componenti chiave di MPLS	129
Header MPLS	130
Setup LSP: selezione del path e delle etichette.....	130

Static label binding (and mapping)	131
Dynamic Label Binding.....	131
Control Driven Label Binding	131
Protocolli per la distribuzione delle etichette	131
Protocolli di routing.....	132
Modi di routing.....	132
Hop-by-hop routing.....	132
Explicit constraint based routing	132
Traffic Engineering.....	133
Fast Fault Recovery.....	134
Label Stack hierarchy and scalability	134
Penultimate Hop Popping (PHP).....	135
Soluzioni MPLS-based	135
MPLS/BGP VPN	137
MPLS/Virtual Router VPN	137
Reti ottiche	138
Optical Core	138
Quality of Service.....	139
Classificazione.....	139
Scheduling	140
Controllo del traffico	140
Standard	140

Ripasso protocollo IPV4

Un indirizzo IP è un identificatore su 32 bit che deve essere assegnato alle interfacce degli host e dei router. Un indirizzo IP è diviso in due parti: **network part** (bit più significativi, i primi bit dell'indirizzo IP) e **host part** (i bit meno significativi). Una **rete IP** è un insieme di dispositivi IP le cui interfacce hanno la stessa network part nell'indirizzo IP e sono connesse allo stesso link fisico, appartenendo dunque alla stessa rete di

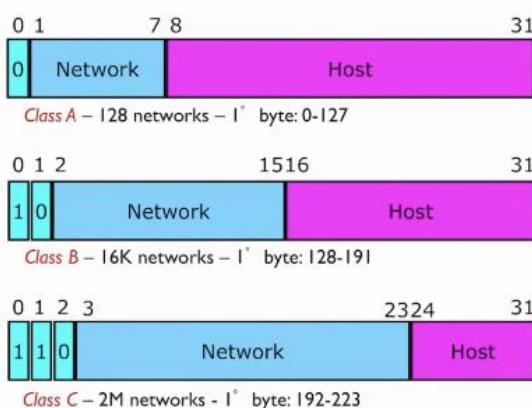
livello 2. Ipotizzando che la parte di rete degli indirizzi sia come in figura, la rete così configurata ha un indirizzamento IP valido. Sono presenti due reti IP (una a sx e una a dx) e queste due reti IP sono costruite su due reti di livello 2 diverse, di cui una ethernet e una Wi-Fi. Questo lo si vede dal fatto che la prima parte dell'indirizzo IP è comune (a sx 223.1.1, dall'altra parte 223.1.2). La host part cambia perché in quella parte è necessario scrivere un numero diverso per ogni interfaccia.



diverso per ogni interfaccia.

- **Network ID** (o *subnetwork ID*): è composto da alcuni valori, mentre il resto dei bit è tutto a 0. Serve per identificare una determinata rete. Tornando all'esempio di prima il network ID è 223.1.1.0.
- **Limited broadcast**: 255.255.255.255 indirizzo speciale che non può essere assegnato alle interfacce. Ha uno scopo analogo a quello del livello 2: si tratta di una trama broadcast che si propaga all'interno di una certa rete. **Attenzione!** La trama broadcast non attraverserà il router, ma la invierà a tutti gli host della stessa rete.
- **Directed broadcast**: presenta alcuni valori nella parte di rete e tutti 1 (255) nella parte di host. Serve per raggiungere tutti i dispositivi di una certa rete. Ritornando all'esempio precedente, serve per raggiungere gli host di una qualsiasi rete, ad esempio l'host 223.1.1.3 per comunicare con la rete 223.1.2.0 invia una trama con indirizzo IP 223.1.2.255.
- Gli indirizzi IP 223.1.1.0 e 223.1.1.255 (directed broadcast) non possono essere assegnati agli host.
- **Loopback**: 127.0.0.1 serve per raggiungere sé stessi. Usato in passato per scopi di debug.

Per capire dove finisce la parte di rete e inizia quella di host si può utilizzare una definizione **statica** o **dinamica**.



Indirizzamento classful

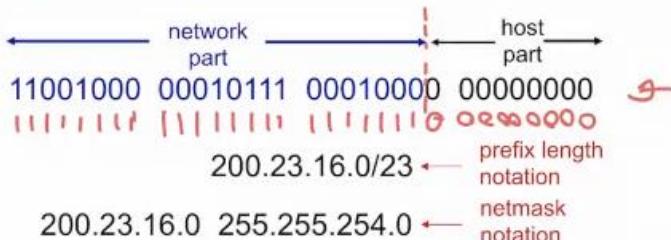
Col metodo statico si esaminano i primi bit: ad esempio, tutti gli indirizzi IP che iniziano con 0 (sul MSB) hanno la network part che finisce dopo 8 bit. **110.0.0.0** → se lo si converte in binario si ha 0 sul primo bit: $10000000 = 128$ (fino a 127 è 0 nel primo bit, da 128 in poi è 1). **110.0.0.0** → Network ID (3 zeri negli ultimi byte) **110.1.0.0** → Indirizzo di host (uno dei 2^{24} combinazioni di bit rimanenti).

Se all'inizio si hanno i bit 10 si identificano gli indirizzi di **Classe B** dove la network part finisce dopo 16 bit.

Se si ha 110 nei primi 3 bit la network part finisce dopo 24 bit. Questo era l'**indirizzamento classful** che oggi non è più utilizzato.

Indirizzamento classless

Lo standard che lo definisce è detto **CIDR (Classless InterDomain Routing)**. Si tratta di utilizzare una seconda stringa di 32 bit che possa permettere di definire dove finisce la network part. Questa seconda stringa è detta **netmask**. La netmask è una stringa di 32 bit dove si hanno tutti 1 sulla network part e tutti 0 sulla host part.

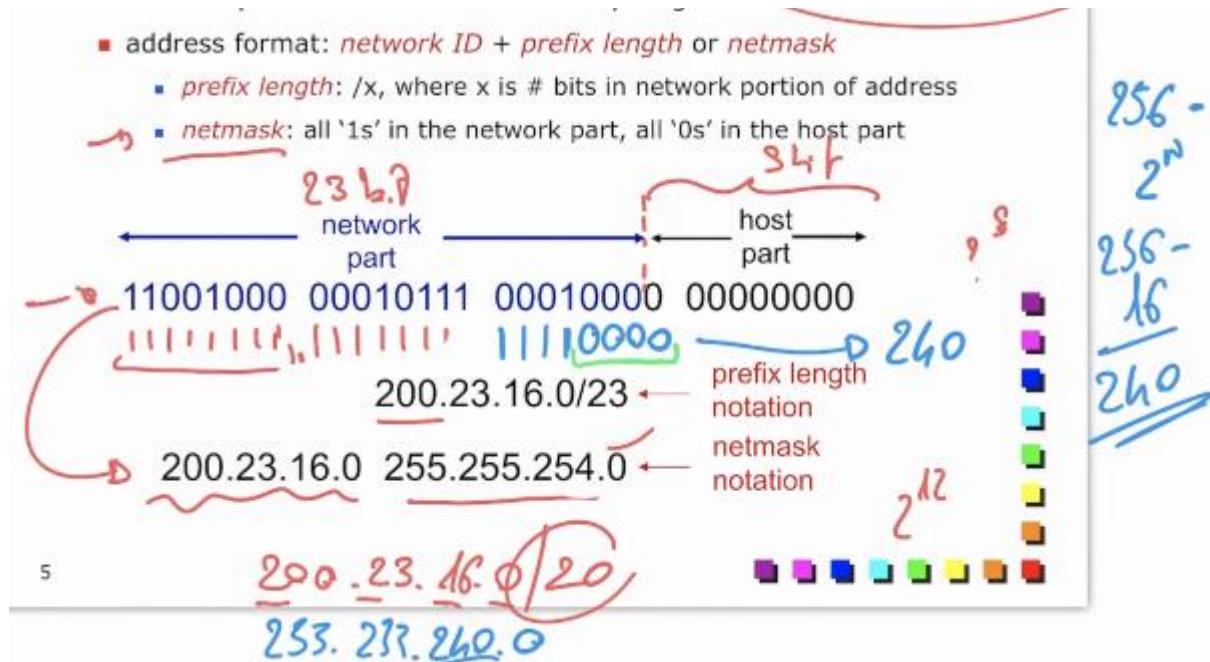


Volendo scrivere la netmask associata all'indirizzo in figura si procede come segue. Si scrivono tutti bit a 1 dove c'è la network part e 0 dove c'è l'host part. Successivamente ci si accorge che la network part finisce dopo 23 bit. Ovviamente non è necessario scrivere gli indirizzi in binario ma basta scriverli in notazione decimale puntata.

In questo standard è stato anche introdotto il **prefix length**, facendo diventare il tutto ancora più user friendly: la **netmask notation** della foto precedente diventa:

200.23.16.0/23 → ID di rete valido perché sono presenti 9 zeri nella host part. Siccome l'indirizzo IP ha più di 9 zeri, si potrebbe definire un'altra netmask ancora valida ma più grande. Per esempio: **200.23.16.0/20** → ID di rete valido. Se l'indirizzo 200.23.16.0/23 viene allocato, non è possibile allocare 200.23.16.0/20 perché si creerebbero delle sovrapposizioni degli spazi di indirizzamento.

Pro Tip: $256 - 2^n$ dove n è il numero di bit a 0 come in foto:



(la prossima pagina in fondo contiene esempi esplicativi)

Le possibili stringhe nei vari byte di indirizzo possono essere:

- **Valid netmasks:** possible values in the 4 bytes composing the address

0	0000 0000	(256)
128	1000 0000	(128)
192	1100 0000	(64)
224	1110 0000	(32)
240	1111 0000	(16)
248	1111 1000	(8)
252	1111 1100	(4)
254	1111 1110	(2)*
255	1111 1111	(1)

*not valid in the 4° byte

Aumentando il numero di bit a 1 si hanno le rispettive netmask e numero di host disponibili.

Si nota che la netmask 255.255.255.254 non esiste perché andrebbe a identificare una rete con soltanto 1 bit nella parte di host. Bisogna ricordarsi degli indirizzi speciali (primo e ultimo) che non si possono utilizzare. Per collegare due nodi serve almeno una netmask /30, dunque una 255.255.255.252.

■ Some examples

- 130.192.0.0/16 - 130.192.0.0 255.255.0.0
- 130.192.0.0/24 - 130.192.0.0 255.255.255.0
- 130.192.0.0/25 - 130.192.0.0 255.255.255.128
- 130.192.2.0/23 - 130.192.2.0 255.255.254.0
- 130.192.1.4/30 - 130.192.1.4 255.255.255.252
- ~~130.192.1.0/31 - 130.192.1.0 255.255.255.254~~

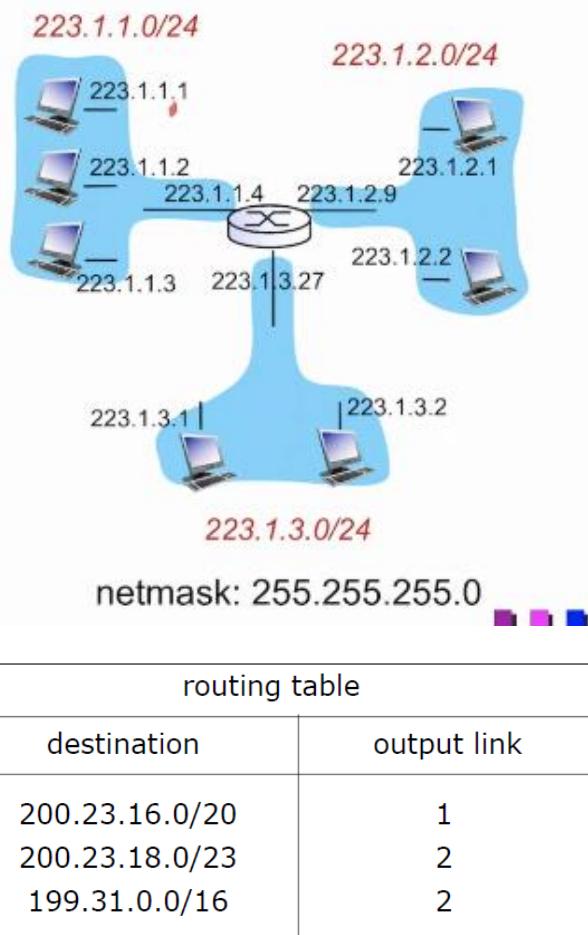
Each IP network must contain at least the network ID and the broadcast address!

Il 255 finale, con netmask 255.255.255.255 in realtà è valida ma non nella realizzazione di piani di indirizzamento, poiché in realtà identifica il singolo dispositivo. Viene utilizzata nelle tabelle di routing per identificare il singolo dispositivo. Ad esempio: con 130.192.1.1 - 255.255.255.255 sto esprimendo una route specifica per quella particolare destinazione.

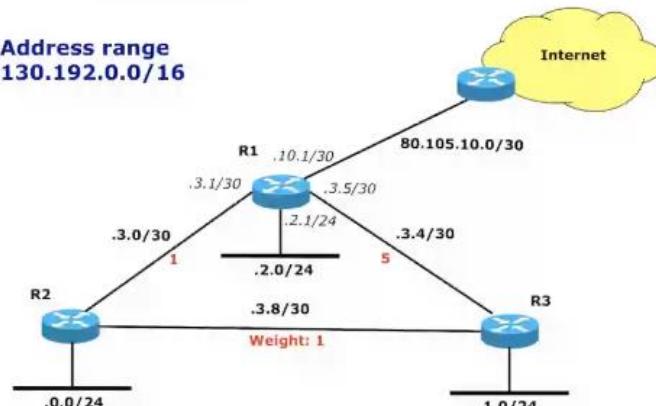
Guardando la figura si analizzano alcuni esempi:

- **130.192.2.0/23**: il 2 in binario diventa 00000010, mentre con lo 0 finale diventa: 00000010 00000000 ovvero 9 bit di host part. Faccio dunque $256 - 2^1 = 254$ e ottengo la netmask corrispondente: 255.255.254.0.
- **130.192.1.4/30** è valido perché 4 è un multiplo di 2^2 (la netmask /30). Questo è un altro **Pro Tip**. /30 -> 2 bit a 0 nella parte di host. Prendo questi 2 bit, che sarebbe il valore di "n". 2^n è 4 e tutti i multipli di 2^2 vanno bene.
- 130.192.1.16/30 è valido -> 16 multiplo di 4
- 130.192.1.16/29 è valido -> 16 multiplo di 8
- 130.192.1.16/28 è valido -> 16 multiplo di sé stesso (16)
- L'ultimo esempio della foto non è valido perché le /31 non esistono.

Altro esempio (Regole generali)



Esempio



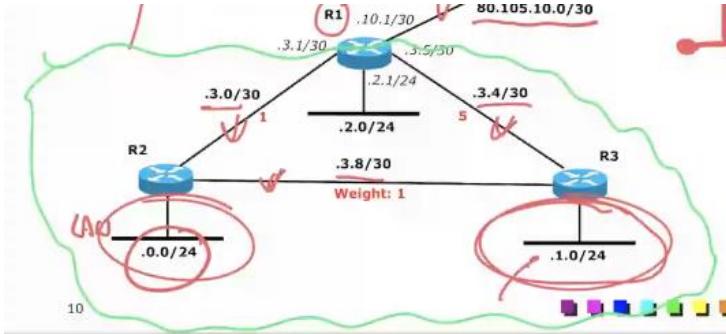
Dato un indirizzo IP da raggiungere, un dispositivo IP cerca nella sua routing table un **match**. Avere un match nella routing table vuol dire che unendo l'indirizzo IP destinazione e le netmask delle routing table, si ottiene proprio l'indirizzo IP scritto sulla riga. Ad esempio: arriva un pacchetto con IPDEST: 200.23.16.1. Se si fa un AND con la netmask della prima riga /20 : 255.255.240.0 viene fuori: 200.23.16.0, che è proprio la stringa della riga della routing table e dunque si ha un **match**. Per fare ciò si utilizza il **bitwise and** (and bit a bit). Facendo l'and dello stesso IPDEST con la /16 si ottiene 200.23.0.0 e non si ha un match.

Se invece si utilizza come IPDEST: 200.23.18.1 facendo un AND bit a bit col primo risultato si avrebbe un match perché viene fuori 200.23.16.0.

Se si fa la stessa operazione con la seconda entry, ovvero con il /23, si ottiene 200.23.18.0 ovvero un match. In caso di **match multiplo** si sceglie il **più specifico**, ovvero si utilizza la tecnica del **longest prefix matching**. Si sceglie quello con la netmask più lunga. In questo caso si sceglie quello in /23.

Scrivendo la tabella di routing di R1 si nota che sono state assegnati alcuni indirizzi IP alle reti di livello 2. Anche i link punto-punto hanno bisogno di reti IP (3.0/30, 3.8/30, 3.4/30). La linea nera è il classico simbolo per rappresentare una rete ethernet. Simboli “standard” per il design delle reti:

Router
 Switch



La dicitura $0.0/24$ in foto vuol dire in realtà $130.192.0.0/24$. La parte in verde è l'address range $130.192.0.0/16$, mentre la parte che va verso internet ha un indirizzo IP diverso e va specificato. Il router R1 possiede 4 reti fisiche connesse direttamente. Nella foto compaiono anche i "pesi" che sono sostanzialmente i costi dei link. Il concetto di costo è variabile. Un esempio può essere: più è alto il peso, meno banda ho a disposizione oppure più è alto il peso e più il costo monetario per tale link è alto. Osservando ad esempio la route diretta tra R1 ed R3, anche se essa possiede peso 5 verrà comunque preferita rispetto al "giro" alternativo. Questo accade perché *la consegna diretta ha la priorità rispetto alla route statica*.

Routing table di R1:

R1 routing table			
Type	Destination	Next-hop	Cost
S	130.192.0.0/24	130.192.3.2	2
S	130.192.1.0/24	130.192.3.2	2
S	130.192.3.8/30	130.192.3.2	2
S	0.0.0.0/0	80.105.10.2	2
D	130.192.2.0/24	130.192.2.1	1
D	130.192.3.0/30	130.192.3.1	1
D	130.192.3.4/30	130.192.3.5	1
D	80.105.10.0/30	80.105.10.1	1

remote interface!!
local interface!!

Per raggiungere internet devo passare attraverso il **next-hop** (prossimo passo da effettuare) e devo trovare il modo di rappresentare le reti connesse a Internet, ciò formalmente funziona ma **non è scalabile**. Per ovviare a questo problema è stata introdotta la **default route** ($0.0.0.0/0$). Questa route ha un senso perché nella longest prefix matching vince quello "più vicino". Dunque, questa default route è una entry che permette di fare match con qualsiasi cosa e risulta sempre valido tranne

quando l'indirizzo IP destinazione appartiene a qualcuna delle entry più specifiche (cioè interne alla rete in questo caso specifico). Con poche entry si riesce quindi a mettere in piedi il router R1 correttamente.

Questo discorso vale anche per il **default gateway**, che si ricorda essere il router che serve per uscire dalla propria rete IP. Per scrivere il default gateway dentro la tabella di routing si utilizza lo stesso modo: corrisponde alla **default route**. Ma cosa va scritto nel campo next-hop? Quando si deve dire a R1 che la rete $0.0/24$ si raggiunge attraverso R2 bisogna scrivere in qualche modo che è necessario passare da R2. Bisogna dunque scrivere l'indirizzo IP di R2 (cioè di quell'interfaccia), poiché quello è il next-hop, cioè il prossimo passo per raggiungere la destinazione. Nella foto di cui sopra non è presente l'indirizzo dell'interfaccia, ma in realtà l'indirizzo di quell'interfaccia è $3.2/30$, poiché avendo 4 host possibili e $3.3/30$ è il broadcast, sarà per forza $3.2/30$.

Attenzione! Nel next-hop **non va inserita la netmask, ma solo l'indirizzo IP**. La maschera è necessaria solo per identificare le destinazioni (che sono sostanzialmente delle reti). Non ha senso utilizzare un'interfaccia locale come next-hop, ma bisogna sempre usare quella remota.

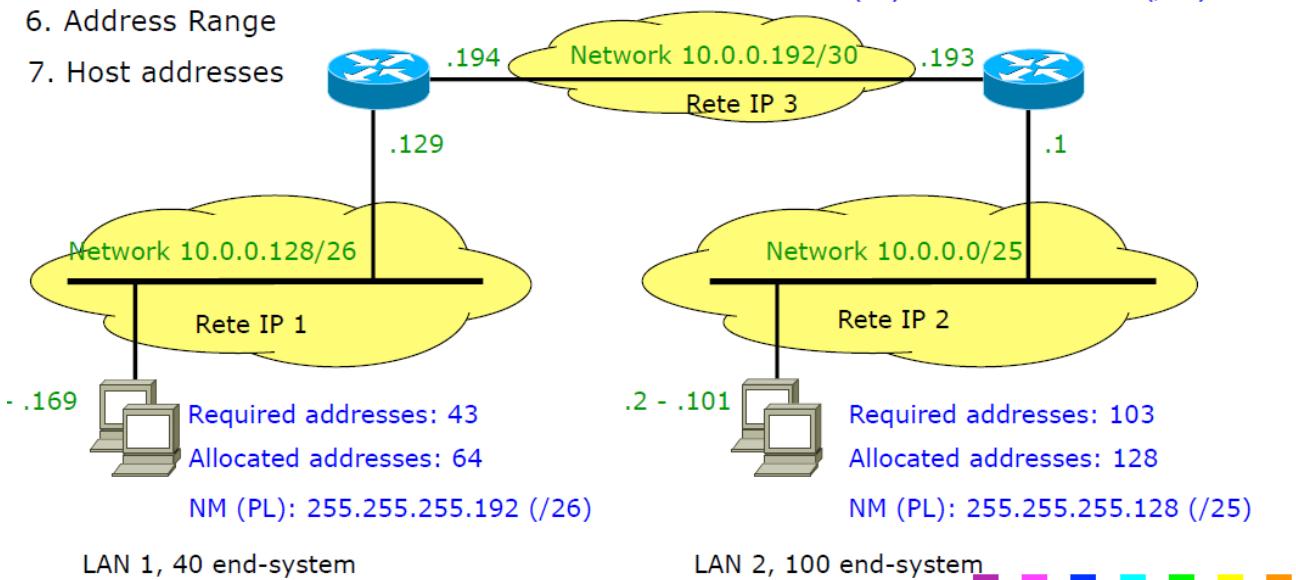
IP addressing: metodologia

IP Addressing: methodology

1. Location of IP networks
2. Amount of required addresses
3. Amount of allocated addresses
4. Address range validity
5. Netmask / Prefix Length
6. Address Range
7. Host addresses

Minimum amount of addresses: 196
Address range selected: 10.0.0.0/24 → OK

Required addresses: 4
Allocated addresses: 4
NM (PL): 255.255.255.252 (/30)



Osservando la figura si nota che sono presenti 3 reti di livello 2: 3 reti fisiche, cioè 3 reti a cui bisogna assegnare una rete IP.

- *Di quante reti IP si ha bisogno?* 3.
- *Quanti indirizzi IP servono?* **LAN1:** Essendoci 40 dispositivi collegati e contando anche i router, servono 41+2 (ricordando che il primo e l'ultimo indirizzo non si possono usare) indirizzi IP. Siccome non esiste una rete da 43 host, è necessario utilizzare una rete da 64 indirizzi IP (solo potenze di 2). Gli indirizzi IP in eccesso non possono essere usati altrove, ma solo in quella rete. Ripetendo lo stesso ragionamento servono in totale $64+128+4 = 196$ indirizzi IP.

Giunti a questo punto sono possibili varie scelte: è possibile scegliere tra indirizzi privati oppure indirizzi pubblici in base a cosa viene richiesto. Nel caso in cui si scegliersero gli indirizzi pubblici bisognerebbe comprarli oppure si è già in possesso di tali indirizzi? Potrebbe dunque essere richiesto di verificare quanti indirizzi servono per poter poi verificare che il numero di indirizzi disponibili per quell'azienda sia sufficiente.

- *L'address range è valido?* Nell'esempio, l'address range viene già dato ed è di tipo privato: *10.0.0.0/24* e bisogna solo verificare che vada bene. Questo address range è corretto perché si è calcolato che sono necessari 196 indirizzi, a cui corrisponde un address range di 256 (2^8), che è la /24. Per la rete da 64 si usa /26, per quella da 128 si usa /25 e per quella da 4 la /30.

A questo punto si è pronti per assegnare gli indirizzi (ID di rete) alle varie reti fisiche. L'unico vincolo è che gli indirizzi che si assegnano devono essere interni all'address range. Avendo *10.0.0.0/24* è necessario stare

attenti a non sforare tale range (quindi non bisogna utilizzare indirizzi come $10.0.1$, ...). Si consiglia di partire dalla rete a più grande dimensione, verso quella a più piccola dimensione.

- La prima rete in ordine di grandezza è quella da 128 quindi andrò ad assegnare la rete: $10.0.0.0/25$ alla rete IP 2. Usare questo metodo è più semplice perché questa rete finirà a $10.0.0.0.127$, che corrisponde al broadcast di quella rete.
- Il prossimo indirizzo libero sarà $10.0.0.128$ e si è sicuri che questa nuova rete sia una rete valida /25 perché avrà nuovamente sette 0 finali. Dunque, $10.0.0.128/26$ sarà la rete che si assegnerà a IP 1 (successiva in ordine di grandezza). La rete finisce dopo 64 indirizzi, ovvero a $10.0.0.191$ (broadcast seconda rete).
- Il prossimo indirizzo utilizzabile è $10.0.0.192$ che sarà per definizione una /26 valida e sarà anche sicuramente una /30. Dunque, si assegna $10.0.0.192/30$ alla rete IP 3. La rete finirà a $10.0.0.195$, che corrisponde al broadcast.
- Il prossimo indirizzo valido è $10.0.0.196$ e da qua fino a $10.0.0.255$ sono tutti indirizzi LIBERI, non ancora utilizzati e saranno utilizzabili **all'interno della rete**.

Di seguito un **controesempio** in cui si parte dalla rete più piccola:

- Si assegna $10.0.0.0/30$ che termina a $10.0.0.3$ (broadcast).
- Volendo poi assegnare la rete in /25, non la si può assegnare da $10.0.0.4$ perché non possiede 7 bit a 0 finali (per la regola delle potenze vista in precedenza, deve essere almeno multiplo di sé stesso). Si andrà dunque a selezionare $10.0.0.128/25$ fino a $10.0.0.255$ (finendo l'address range).
- Volendo poi assegnare la rete in /26 si può cadere in errore andando a selezionare la rete $10.0.1.0/26$ ma è **fuori dall'address range**. Devo dunque tornare indietro e cercare $10.0.0.64/26$ che termina a $10.0.0.127$ e da $10.0.0.4$ fino a $10.0.0.63$ saranno **liberi**.

In conclusione: **METODO COMPLICATO E CON POSSIBILI ERRORI.**

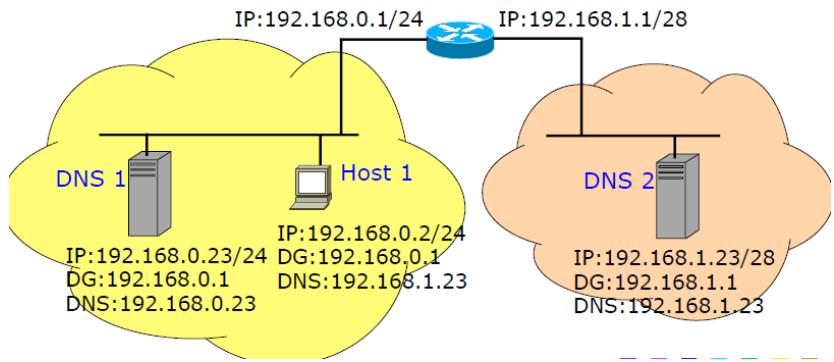
Esercizi

Esercizio 3: “Trovare l’errore di configurazione nel setup della seguente rete e spiegare perché le reti non funzionano correttamente.”

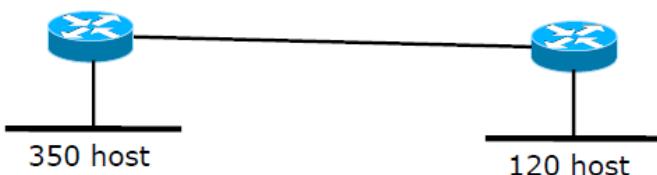
In questo esercizio è presente l'**Host 1**

che può dover parlare con un DNS all’interno della rete e con il DNS fuori dalla rete. Il DNS dell’host è quello del DNS 2, ovvero 1.23, mentre l’host 1 è in 0.2/24 ma ciò non è un problema (ricordando DNS da Reti di Calcolatori).

Il default gateway dell’Host 1 è dentro la propria rete IP (poiché la maschera è /24) e dunque l’host riesce a parlare col proprio Default Gateway. Guardando la rete del DNS 2 bisogna verificare altrettanto: il DG del DNS 2 è 1.1/28 ma si trova a dover parlare con 1.23/28. Ma 1.1/28 non è nella stessa rete di 1.23/28 perché 1.1/28 è nella rete 1.0/28 ed il broadcast di questa rete è 1.15. Dunque, 1.23 è fuori dalla rete: il router avrebbe bisogno di una route più specifica o di una route statica perché non appartiene alle reti ad esso direttamente corrette. Per tale motivo, il router non sarà mai in grado di inoltrare i pacchetti al DNS 2. Cambiando maschera alla rete 1.1 da /28 a /27 tutto funzionerebbe (possibile soluzione). Un’altra soluzione è mantenere la 1.1/28 e cambiare l’indirizzo IP del DNS 2 in 1.10/28 (ad esempio).



Esercizio 4: “Definire un piano di indirizzamento IP per la rete in figura. Considerare entrambi come piani di indirizzamento “tradizionali” e cercare una soluzione che potrebbe minimizzare il numero di indirizzi inutilizzati. Si assuma di voler utilizzare l’address range 10.0.0.0/16.”

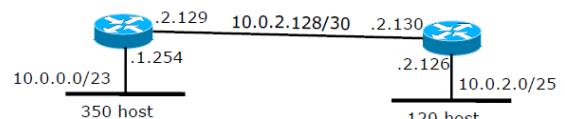


con una /25. L’address range fornito è 10.0.0.0/16 ed è più che sufficiente per gli indirizzi da gestire. A questo punto si può applicare il metodo già visto, assegnando le reti partendo dalla rete più grossa e andando verso quella più piccola. Si assegna 10.0.0.0/23 alla rete di sinistra, che terminerà a 10.0.1.255/23. Il prossimo indirizzo utile sarà 10.0.2.0/23 ed essendo valido in /23, sarà valido anche in /25 e viene assegnato alla rete di destra. Questa rete finirà all’indirizzo 10.0.2.127/25 e per il collegamento punto-punto utilizzerò la rete 10.0.2.128/30. Problema concluso.

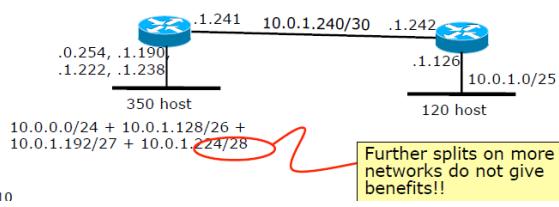
Soluzione 2: nella prima soluzione si sono utilizzati più indirizzi IP di quelli necessari. Se non si dovranno aggiungere più host a quella rete, gli indirizzi saranno sprecati. Una strategia potrebbe essere quella di dividere la rete fisica in più reti logiche. Bisogna iterare il procedimento fintanto che ci si accorge che aggiungere un’altra logica non porta

Soluzione 1: sono presenti due reti ethernet e un link punto-punto tra i due router, dunque un totale di 3 reti fisiche a cui assegnare reti IP. Nella rete di sinistra sono necessari 512 indirizzi IP, ovvero una maschera /23. Quella da 120 host richiede 128 indirizzi IP, ovvero

Solution1



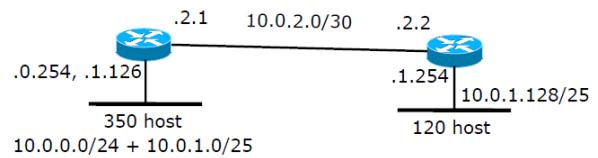
Solution2



alcun beneficio. Ogni rete logica aggiunta porta con sé un nuovo network, un nuovo broadcast e un nuovo router. Non bisogna dimenticare che se si vuole avere più reti logiche sulla stessa rete fisica l'interfaccia del router avrà bisogno di 2 indirizzi IP perché quelle sono reti IP come tutte le altre ed ogni rete IP ha bisogno di un DG per permettere agli host di uscire dalla propria rete. Ecco perché i DG saranno 4.

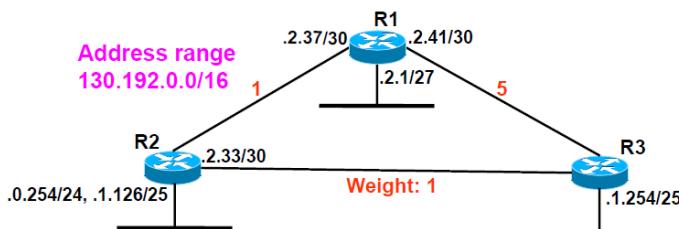
Soluzione 3: Esistono dei casi in cui non è possibile utilizzare 512 indirizzi (perciò la soluzione 1 non va bene). Ad esempio, la specifica avrebbe potuto dire che l'address range era $10.0.0.0/23$ (e sarebbe allora necessaria la soluzione 2). Una soluzione di buon senso però può essere la seguente: l'esercizio chiede di indirizzare 350 host; servirebbe dunque una /23 ma la più grossa rete che non sia la /23 che permette di far stare il maggior numero di host è la /24. Si prendono poi gli indirizzi che rimangono e si rifà la stessa domanda. Prendo $256 - 3 = 253$ host nella /24 e ne avanzano $350 - 253 = 97$ host. Per questa seconda rete si utilizza la /25 e si risparmieranno comunque un buon numero di indirizzi. Nel conto precedente si fa -3 perché bisogna conteggiare anche l'interfaccia del router (oltre a network e broadcast).

Solution 3



Quando si invia un pacchetto broadcast al livello 2 questo viene comunque inviato a tutti gli host (perché la rete fisica è singola), per esempio una ARP Request. Al livello 3, invece, un eventuale *limited broadcast* dovrebbe essere considerato da tutte le interfacce (funziona come il livello 2), invece il *directed broadcast* farebbe delle distinzioni. Ad esempio, un pacchetto diretto a $10.0.0.255$, che è il directed broadcast della prima rete, viene ricevuto da tutti ma considerato solo dagli host che sono sulla rete $0.0/24$ (perché solo quelli hanno tale l'indirizzo come broadcast).

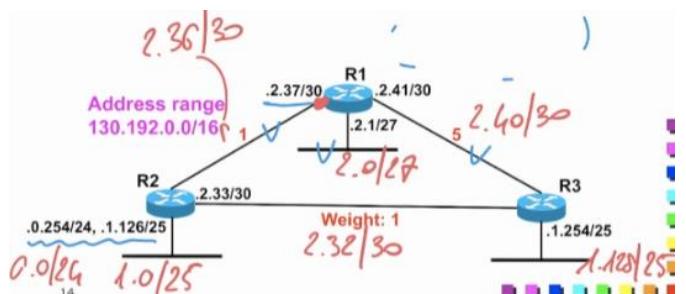
Esercizio 6: “Massimizzare l’aggregazione di route della rete in figura e definire la routing table di R1. Gli indirizzi IP in figura sono riferiti alla più vicina interfaccia del dato router.”



quella interfaccia che collega R1 a R2. Dunque è necessario calcolarsi le reti. Per ricavare la rete, si deve trovare il più grande multiplo di 4 che sia più piccolo di 37. Dunque la rete tra R1 e R2 sarà $.2.36/30$. Quella di R1 con la rete ethernet sarà $2.0/27$, quella tra R1 ed R3 $2.40/30$, quelle di R2 saranno $0.0/24$ e $1.0/25$. Su R3 invece la rete sarà $1.128/25$ e nel canale tra R2 e R3 sarà $2.32/30$.

Una volta trovate le reti bisogna capire come scrivere la routing table di R1.

Le route dirette di R1 in questo caso sono solo 3: la rete tra R1 e R2, la rete ethernet di R1 e la rete tra R1 e R3. La complicazione aggiuntiva di questo esercizio è che non viene data la rete. Gli indirizzi in figura non sono ID di rete ma sono indirizzi assegnati all'interfaccia più vicina di un dato router. Ad esempio, $.2.37/30$ si riferisce a



Type	Destination	Next-hop	Cost
S	0.0.0.0/0	130.192.2.38	2

D	130.192.2.0/27	130.192.2.1	1
D	130.192.2.36/30	130.192.2.37	1
D	130.192.2.40/30	130.192.2.41	1

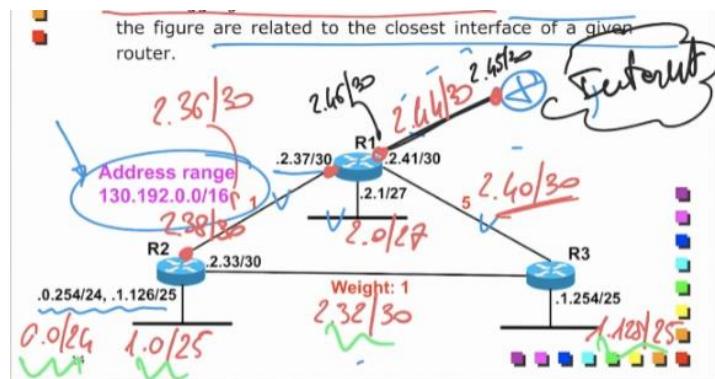
Le tre reti dirette sono banali: si scrive nella destinazione l'IP della rete direttamente connessa al router e come next-hop l'interfaccia locale. **Errore comune:** scrivere nella destinazione l'interfaccia. Questo errore accade perché si scambiano gli ID delle interfacce (che vengono date in figura) per ID di rete. Bisogna dunque calcolarsi le reti e quelle vanno scritte nelle destinazioni. Inoltre, scrivere l'indirizzo di un'interfaccia con la maschera non ha alcun senso, quindi nel next-hop va scritto l'indirizzo senza la maschera.

“Massimizzare l'aggregazione di route” → Scrivere il minor numero possibile di route. Unire e aggregare le eventuali entry che ci sono in una routing table per scriverne il meno possibile. Se non aggregassi, in questo esercizio di route statiche ne dovrei trovare 4: le due del router R2, il canale che collega R2 a R3 e la rete di R3.

Un metodo per aggregare, non essendoci internet, è quello di utilizzare la **default route** per raggiungere tutte le destinazioni, perché tanto qualunque sia la destinazione bisognerà passare da R2 (per via dell'albero di instradamento visto in un esercizio simile precedente).

Si reinserisce adesso Internet alla figura e si ricomincia l'esercizio.

Esercizio 6 (con Internet)



Questo è di nuovo il caso visto la scorsa volta. Per il canale R1-Internet si assegna l'indirizzo **2.44/30** che è il primo indirizzo disponibile. Si assegna poi all'interfaccia lato internet il **2.45/30** mentre all'interfaccia lato R1 si assegna **2.46/30**.

In questo caso la default route deve essere utilizzata per andare verso internet, perciò la prima riga statica della soluzione precedente

non va più bene. Deve dunque essere sostituita dalla seguente:

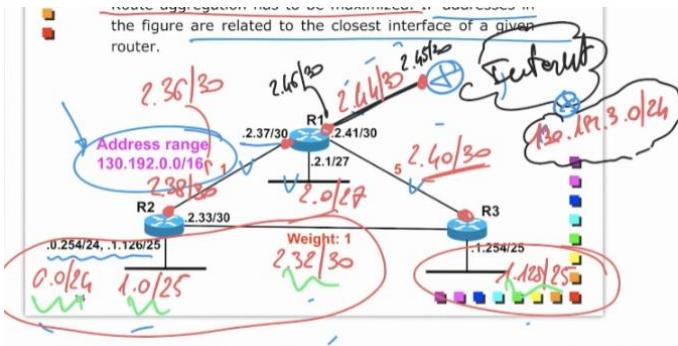
S	0.0.0.0/0	130.192.2.45	2
---	-----------	--------------	---

L'obiettivo deve essere comunque quello di massimizzare l'aggregazione. Sapendo che per definizione tutte le reti (sia statiche che dinamiche) della figura appartengono al dato address range **130.192.0.0/16** e hanno tutte in comune i primi 2 byte. Per tale motivo è possibile scrivere, molto velocemente, la seguente riga:

S	130.192.0.0/16	130.192.2.38	2
---	----------------	--------------	---

Questo perché tutte le reti appartengono a tale address range. **2.38/30** è l'indirizzo dell'interfaccia di R2 (perché tutto il traffico passa da lì). Se Il link R1-R3 non avesse avuto peso 5 non si sarebbe potuto aggregare in questo modo.

Il costo, per convenzione si assegna un costo superiore alle route statiche rispetto alle dirette. Il costo interviene quando esistono due strade per raggiungere la stessa destinazione (ma non viene trattato).



Questo metodo può generare problemi nel caso in cui la stessa azienda con address range **130.192.0.0/16** voglia aprire un'altra sede altrove ma comunque usare una parte degli indirizzi disponibili dell'address range. Se supponiamo di avere aggiunto altrove una sede **130.192.3.0/24**, la tabella di R1 non funzionerebbe più, perché invierebbe verso R2 tutti i pacchetti magari diretti verso **130.192.3.0/24**.

La tabella potrebbe dunque essere migliorata rinunciando a un po' di aggregazione (e di semplicità) e andando a fare i conti per fare in modo che la tabella sia meno a rischio nel momento in cui qualcuno usi "pezzi" di address range in giro per il mondo. Per arrivare a ciò si dice che *la routing table non deve contenere indirizzi non ancora utilizzati* (meglio detta *routing table equivalente*). Per risolvere dunque questo tipo di problema non c'è altra strada se non quella di scriversi le reti e ragionarci su.

Lista Reti da aggregare

0.0/24

1.0/25

1.128/25

2.32/30

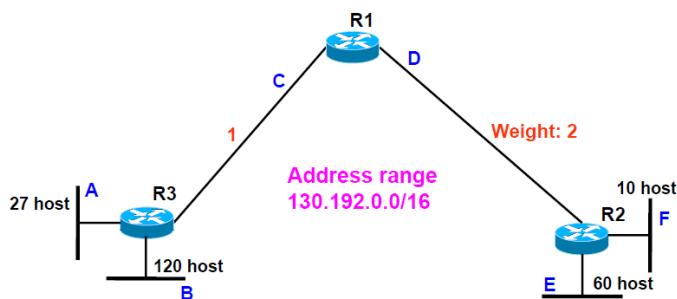
Si può osservare che **1.0** e **1.128** insieme (visto che sono entrambe /25) generano un **1.0/24** (address range che contiene quelle due entry). Successivamente mi accorgo che **1.0/24** e **0.0/24** mi danno una **0.0/23**. Se voglio aggiungere anche la **2.32/30** posso provare a vedere se la **0.0/22** è sufficiente per la **2.32/30** e la risposta è affermativa perché **0.0/22** va da **0.0** fino a **3.255** (e quindi è incluso anche **2.32**). Però attenzione, non rispetta il vincolo imposto perché contiene anche **3.0/24** che non è ancora utilizzata ma che potrebbe essere utilizzata in futuro (in pratica nei casi di prima quando si aggregava tutti gli indirizzi erano già stati utilizzati, come ad esempio **1.0/24** e **0.0/24** che insieme danno una **0.0/23** tutta già "piena"). Quindi la nuova routing table di R1 sarà:

S	0.0.0.0/0	130.192.2.45	2
D	130.192.2.0/27	130.192.2.1	1
D	130.192.2.36/30	130.192.2.37	1
D	130.192.2.40/30	130.192.2.41	1
S	130.192.0.0/23	130.192.2.38	2
S	130.192.2.32/30	130.192.2.38	2

La 5° riga contiene **0.0/23** poiché contiene il "risultato migliore" a cui si è arrivati e poi successivamente viene inserita la non aggregata **2.32/30**. Entrambe le righe avranno lo stesso next-hop. La disottimizzazione è evidente: due entry con stesso next-hop. Con questa struttura il router gestirà

correttamente la rete, a prescindere dall'esistenza o meno della “nuova” rete ***130.192.3.0/24***. Perché pacchetti indirizzati per quest'ultima verranno correttamente indirizzati fuori dalla rete (cioè verso Internet) ed eventuali provider sapranno raggiungerla.

Esercizio 8: “Definire un piano di indirizzamento IP che massimizzi il routing di R1. Derivare la risultante routing table di R1.”



nell’area 1 si ha bisogno di $32+128 = 160$ indirizzi, dunque l’address range per l’area 1 avrà bisogno di contenere 256 indirizzi. Nell’area 2 si hanno $64+16 = 80$ indirizzi. I due address range potrebbero essere:

- Area 1: 130.192.0.0/24
- Area 2: 130.192.1.0/25

Sono entrambi pezzi dell’address range 130.192.0.0/16 che possono essere usati per questo tipo di rete. All’interno delle aree si può utilizzare il solito metodo: all’interno dell’area 1 si considera la rete B, poi la A a seguire e si ottiene il seguente risultato:

B: 130.192.0.0/25

A: 130.192.0.128/27

C: 130.192.0.160/30

D: 130.192.0.164/30

E poi si passa all’area 2:

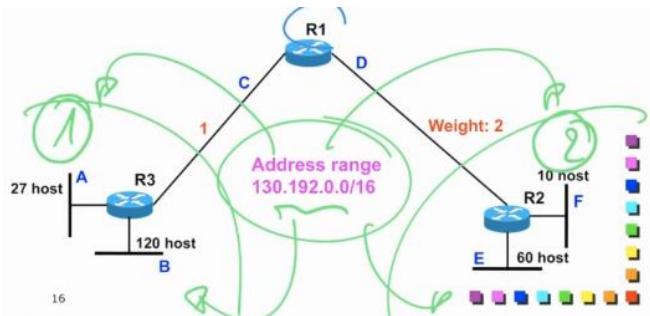
E: 130.192.1.0/26

F: 130.192.1.64/28

Realizzando questo piano di indirizzamento si ottiene la seguente Routing Table per R1:

Type	Destination	Next-hop	Cost
S	130.192.0.0/24	130.192.0.161	2
S	130.192.1.0/25	130.192.0.165	2

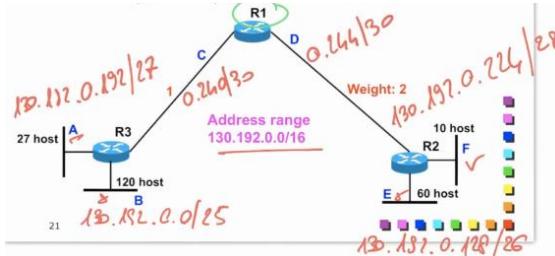
Guardare prima esercizio 9 e poi tornare qui. Visto che ci sono due next-hop sicuri, per risolvere l’esercizio si può dividere la rete in due aree e dare una parte dell’address range alle rispettive due aree (come in figura in basso). Si va ad applicare il metodo già visto non all’interno della rete ma all’interno delle singole aree. Si va a calcolare il numero di indirizzi che servono nell’area 1 e



D	130.192.0.160/30	130.192.0.162	1
D	130.192.0.164/30	130.192.0.166	1

Si inseriscono semplicemente gli indirizzi delle due aree.

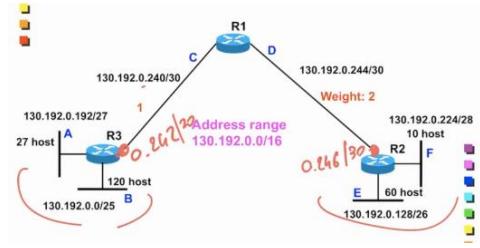
Esercizio 9: “Ripetere l'esercizio precedente definendo un piano di indirizzamento IP che minimizzi il numero totale di indirizzi IP inutilizzati. Spiega come la routing table di R1 cambia rispetto all'esercizio precedente.”



fianco.

Si passa ora a scrivere la routing table massimizzando le aggregazioni. Per una delle due si può utilizzare la default route. Si assegnano gli indirizzi IP alle due interfacce di R2 ed R3 e si suppone che con la default route si reindirizza a R3. Dall'altro lato non si può fare la stessa cosa.

La rete più grossa è la rete B a cui si va ad assegnare 130.192.0.0/16. Dopo si assegna 130.192.0.128/26 alla rete E. Poi si passa alla rete A assegnando 130.192.0.192/27. Infine rimane la rete F che è 130.192.0.224/28. Restano poi le due /30 C e D: 130.192.0.240/30 per la rete C e 130.192.0.244/30 per la rete D. Il risultato è riportato nella figura a fianco.



Type	Destination	Next-hop	Cost
S	0.0.0.0/0	130.192.0.242	2
S	130.192.0.0/16	130.192.0.246	2

Se per l'altro ramo si utilizza l'address range proprio come fatto in precedenza, questa soluzione funziona?

S	130.192.0.0/16	130.192.0.246	2
---	----------------	---------------	---

Questa riga non va bene perché se arrivasse un pacchetto diretto a 130.192.0.1, andrebbe verso R2, perché la riga sopra include anche le reti raggiungibili verso R3. Una cosa che si potrebbe fare è ragionare un po' di più e provare a fare i conti e scrivere come prima la più piccola aggregazione possibile. Dunque si avrà:

Lista Reti da aggregare

0.224/28

0.128/26

In questo caso si può utilizzare **0.128/25** per aggregare entrambe (non si può fare di meglio).

S	130.192.0.128/25	130.192.0.246	2
---	------------------	---------------	---

Questa route funziona? Se arriva un pacchetto per 130.192.0.200, purtroppo si farà match con la entry per R2 (quando invece deve andare su R3), dunque non funziona.

L'idea potrebbe essere quella di unire la route appena indicata (con la 0.128/25) e la default route insieme ad una route specifica per la rete /27 di R3 che crea il problema sopra citato.

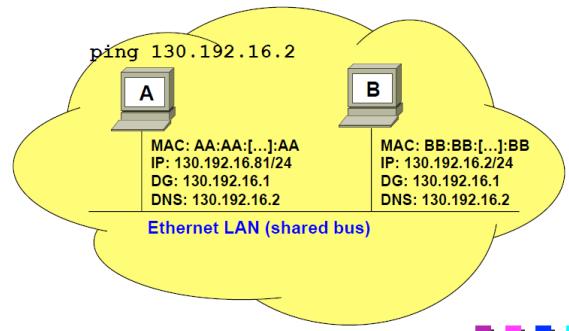
S	130.192.0.192/27	130.192.0.242	2
---	------------------	---------------	---

Questa soluzione funziona. La soluzione migliore però sarebbe quella di **evitare** la default route:

Type	Destination	Gateway (next-hop)	Weight
S	130.192.0.0/25	130.192.0.242	2
S	130.192.0.192/27	130.192.0.242	2
S	130.192.0.128/25	130.192.0.246	2
D	130.192.0.240/30	130.192.0.241	1
D	130.192.0.244/30	130.192.0.245	1

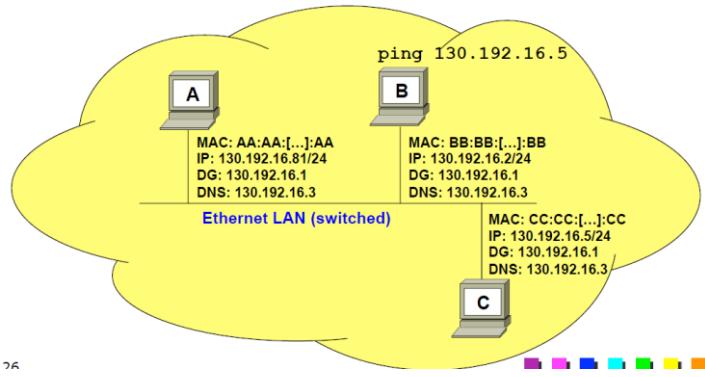
Esercizio 10: “Assumendo che tutte le cache siano vuote, indicare il numero e il tipo dei frame catturati da uno sniffer situazione al cavo di rete dell’Host A.

L’host A lancia il comando ping 130.192.16.2”



N	MAC SRC	MAC DST	IP SRC	IP DST	Descrizione
1	Mac A	Broadcast	/	/	ARP Request A → B
2	Mac B	Mac A	/	/	ARP Reply B → A
3	Mac A	Mac B	IP A	IP B	ICMP Echo Request
4	Mac B	Mac A	IP B	IP A	ICMP Echo Reply
5	Mac A	Mac B	IP A	IP B	ICMP Echo Request
6	Mac B	Mac A	IP B	IP A	ICMP Echo Reply
7	Mac A	Mac B	IP A	IP B	ICMP Echo Request
8	Mac B	Mac A	IP B	IP A	ICMP Echo Reply
9	Mac A	Mac B	IP A	IP B	ICMP Echo Request
10	Mac B	Mac A	IP B	IP A	ICMP Echo Reply

Esercizio 11: “Assumendo che le cache siano vuote, indicare il numero e il tipo di frame catturati da uno sniffer situazione sul cavo di rete dell’host A. L’host B lancia il comando ping 130.192.16.5.”

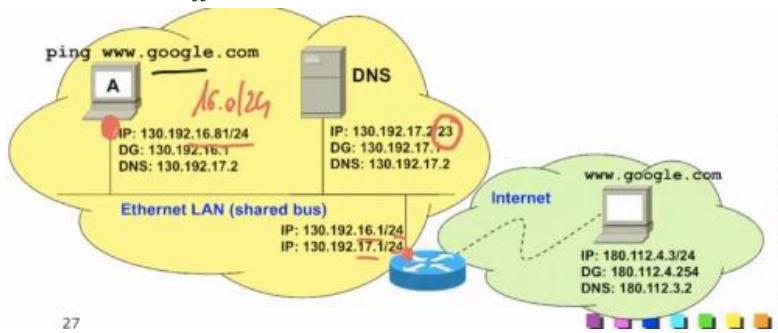


26

In questo caso la rete è **switched** (prima era **shared**). Differenza: lo switch si differisce rispetto all'hub perché è in grado di effettuare forwarding selettivo alle porte. Wireshark su A non sarà in grado di vedere la trama da B verso C (perché verrà indirizzata solo a C). L'hub invece effettua **floding** e una trama diretta solo a C viene inviata anche ad A. L'ARP Request è una trama broadcast, dunque non ci sono differenze tra switch e hub. In questo esercizio, dunque, viene vista da A solo l'ARP Request.

N	MAC SRC	MAC DST	IP SRC	IP DST	Descrizione
1	Mac B	Broadcast	/	/	ARP Request B → C

Esercizio 13: “Assumendo che tutte le cache siano vuote (tranne quella del server DNS), indicare il numero e il tipo dei frame catturati da uno sniffer situato sul cavo di rete dell’host A.”



27

Sul DNS la configurazione è errata. (c’è la maschera /23 invece della /24). Cosa succede in rete a fronte di questo errore? Lo si analizza con la seguente tabella (il commento è in fondo).

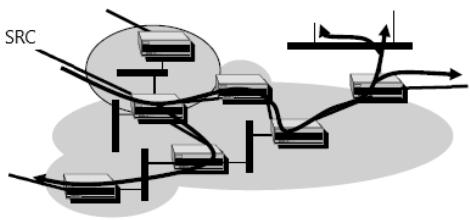
N	MAC SRC	MAC DST	IP SRC	IP DST	Descrizione
1	Mac A	Broadcast	/	/	ARP Request A → DG
2	Mac DG	Mac A	/	/	ARP Reply DG → A
3	Mac A	Mac DG	IP A	IP DNS	DNS Query google
4	Mac DG	Broadcast	/	/	ARP Request DG → DNS
5	Mac DNS	Mac DG	/	/	ARP Reply DNS → DG
6	Mac DG	Mac DNS	IP A	IP DNS	DNS Query google (TTL-1)
7	Mac DNS	Broadcast	/	/	ARP Request DNS → A
8	Mac A	Mac DNS	/	/	ARP Reply A → DNS
9	Mac DNS	Mac A	IP DNS	IP A	DNS Response google
10	Mac A	Mac DG	IP A	IP Google	ICMP Echo Request
11	Mac DG	Mac A	IP Google	IP A	ICMP Echo Reply
12	Mac A	Mac DG	IP A	IP Google	ICMP Echo Request
13	Mac DG	Mac A	IP Google	IP A	ICMP Echo Reply
14	Mac A	Mac DG	IP A	IP Google	ICMP Echo Request
15	Mac DG	Mac A	IP Google	IP A	ICMP Echo Reply
16	Mac A	Mac DG	IP A	IP Google	ICMP Echo Request
17	Mac DG	Mac A	IP Google	IP A	ICMP Echo Reply

Essendo presente un nome, bisogna tradurre tale nome tramite il DNS. Se A deve raggiungere 17.2 (cioè il DNS) si accorge che non è nella propria rete e allora andrà dal default gateway, per tale motivo i primi 2

pacchetti sono delle ARP Request/Reply al DG. Successivamente A invierà una DNS Query al DG e quest'ultimo la inoltrerà al DNS (se tutto funzionerà). **Nella DNS Query ci sarà come MAC DST il DG, ma il pacchetto IP è per il DNS.** Il DG ha una porta in 17.1/24, l'indirizzo IP DST è il 17.2 (del DNS) che è direttamente raggiungibile da 17.1/24 e grazie alla route diretta il DG sarà in grado di effettuare l'inoltro. Per farlo, visto che le cache sono vuote avrà bisogno dell'indirizzo MAC del DNS. Per tale motivo effettua una ARP Request (seguita da Reply) e poi inoltra la DNS query. Una volta arrivati sul DNS, avendo l'indirizzo di Google, crea la DNS Response. La rete di appartenenza che il DNS vede è 16.0/23 quando invece dovrebbe essere 17.0/24. Essendo dunque l'indirizzo 16.81/24 nella propria rete (erroneamente) il DNS prova una consegna diretta (pacchetto 7 in cui c'è una ARP Request). L'host A però possiede una configurazione 16.0/24 quindi non vede direttamente il DNS. ARP è però un protocollo di livello 2 quindi non sa cosa vi è al piano superiore e non guarda la configurazione IP per rispondere. Nel momento in cui arriva l'ARP Request, il dispositivo risponde se l'ARP è per sé stesso (DNS e Host A sono sulla stessa rete fisica). A questo punto, il DNS conosce l'indirizzo MAC di A e allora può inviare la DNS Response. L'host A la processa e a quel punto può pingare Google e vi saranno le ICMP Echo Request e Reply ripetute 4 volte.

IP Multicast

Utilizzato soprattutto in IPV6 al posto degli indirizzi broadcast per fare il mapping tra gli indirizzi di livello 3 e di livello 2 (questo perché con IPV6 non c'è più ARP, ma il tutto viene affidato a ICMP). L'equivalente dell'ARP Request è in realtà un pacchetto multicast.



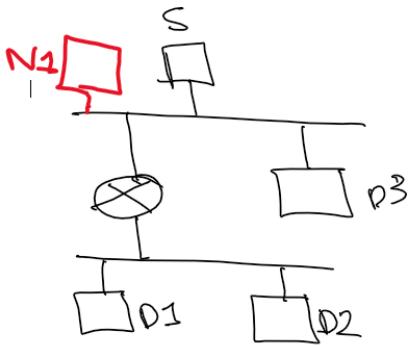
Un traffico si dice Multicast quando da una sorgente bisogna raggiungere più destinazioni. Tutti gli host tranne quello a fianco alla parola "SRC" sulla destra sono interessati al traffico multicast. Tutti gli host "interessati" formano un **gruppo Multicast**. Si può quindi manifestare l'interesse a partecipare a un certo gruppo. La particolarità del traffico multicast è che la sorgente genera un'unica copia del pacchetto che viene

duplicato dal router che invierà una copia verso l'alto e un'altra verso il basso. Si viene a creare dunque un **albero di instradamento** multicast. Questo è molto efficiente per la rete perché nonostante i dispositivi da raggiunge siano 4, la sorgente invierà un'unica copia del dato e sarà poi la rete a crearne il numero di copie necessario. Se in unicast dovessi mandare lo stesso dato a 4 destinazioni, dovrei effettuare io stesso 4 copie dello stesso pacchetto.

Il traffico multicast è molto comodo per comunicazioni video oppure per la IPTV. In realtà però il multicast è usato molto poco perché col multicast la rete (service provider) non ha il totale controllo del traffico. Nel caso unicast vi è sempre nella destinazione l'indirizzo IP da raggiungere, mentre nel caso multicast non è noto a priori il numero di persone interessate a tale traffico. Inoltre, è necessario che i service provider si mettano d'accordo per accordarsi sull'albero di instradamento per i pacchetti multicast. Questo però non avviene mai perché i service provider non si mettono mai d'accordo. Il multicast viene utilizzato in scenari limitati: ad esempio molto tempo fa in aula magna quando venivano effettuate conferenze, il tutto veniva trasmesso in multicast all'interno della rete del Politecnico. Ciò si poteva fare perché il fornitore del servizio e l'amministratore della rete erano la stessa cosa. Un altro caso possibile da citare è Sky insieme a Fastweb: Sky su rete Fastweb viene offerta in multicast perché Fastweb ha l'esatta conoscenza di chi sono i destinatari del traffico multicast ed è in grado di dimensionare la rete. Inoltre, *Sky ≠ SkyGo* poiché Sky Go viene offerta ovunque sul suolo italiano, mentre Sky viene offerta solo dalla rete su cui si ha anche fastweb. SkyGo, infatti, non è fatto in multicast.

Addressing Multicast

La classe D di indirizzi viene utilizzata per l'indirizzamento Multicast. Iniziano tutti con 1110 e vanno da 224.0.0.0 – 239.255.255.255. Ogni indirizzo identifica un gruppo di host e il pacchetto viene inviato a tutti gli host in quel gruppo. Ciò può avvenire dovunque nella rete. Gli host possono effettuare la join ed il **leave** dal gruppo in maniera **dinamica** (Esempio IPTV: ogni canale a disposizione può essere identificato da un gruppo multicast e cambiando canale si fa la leave dal gruppo in cui si era e si fa la join in quello nuovo).



Guardando l'esempio in foto si suppone di avere una sorgente S che vuole inviare traffico multicast a D1, D2 e D3 ma tale traffico non deve raggiungere N1 che non è interessato a quel traffico.

Per gestire il traffico al livello 2 la soluzione banale ma che funziona (e che non viene utilizzata poiché inefficiente) è quella di utilizzare il broadcast. Per definizione la trama arriverebbe a tutti i nodi (N1, D3 e il router) ed il router potrebbe essere configurato per inoltrare tale pacchetto anche ai dispositivi sottostanti D1 e D2.

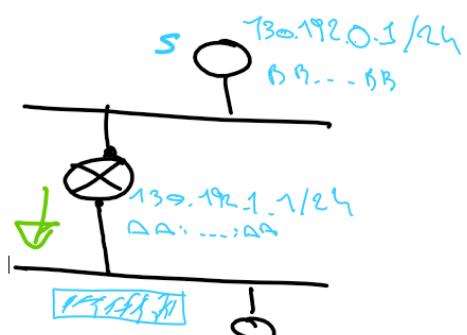
La soluzione non è ottimale perché anche N1 vede la trama e la processa facendola salire al livello 3 (per poi scoprire che tale pacchetto non serve). Per risolvere questo problema sono stati introdotti gli **indirizzi MAC multicast**.

Tutti gli indirizzi MAC Multicast hanno l'ottavo bit a 1, che in realtà è il primo inviato sul canale perché ethernet trasmette in little endian. Gli indirizzi multicast servono a varie cose e quando bisogna mappare un gruppo multicast di livello 3 su un indirizzo MAC multicast si utilizza una stringa di bit come la seguente: **01-00-5E-0** dove si tratta di 6 cifre esadecimali e l'ultimo 0 è un bit (cioè solo che il primo bit è 0). Si tratta quindi di 25 bit e gli ultimi 23 bit sono realizzati con i 23 bit meno significativi dell'indirizzo IP. Ad esempio, se si ha un indirizzo IP del tipo **224.1.1.1** per scrivere il MAC Multicast si prendono gli ultimi 23 bit dell'indirizzo IP per ottenere: **01-00-5E-00000001-01-01** che è l'indirizzo MAC multicast che mappa l'indirizzo IP **224.1.1.1**. Per far funzionare il tutto bisogna comunicare alla scheda di doversi mettere in ascolto per ricevere pacchetti indirizzati al corrispettivo indirizzo MAC multicast.

Ritornando all'esempio precedente se S invia traffico per l'indirizzo IP **224.1.1.1** visto che D1, D2, D3 sono interessati diranno tutti alla scheda di mettersi in ascolto all'indirizzo MAC calcolato prima. L'host N1 invece non effettuerà questa comunicazione e quindi eventuali pacchetti multicast verranno scartati (senza essere ricevuti e processati come nel caso broadcast).

Ponendoci in una rete **shared bus** il traffico multicast funziona come sopra, ovvero il traffico arriva a tutti i dispositivi e quindi bisogna decidere se accettare o scartare tale pacchetto multicast in base al MAC multicast. Su uno **switch** invece le cose sono leggermente diverse: sugli switch esiste il protocollo **IGMP Snooping**, che

permette allo switch di fare forwarding selettivo anche sui gruppi multicast. Tramite questo protocollo è possibile capire (ritornando all'esempio) che N1 non ha fatto la join al gruppo multicast. **Senza tale protocollo, lo switch si comporterebbe come un hub** e invierebbe tale pacchetto in broadcast (sulla stessa rete, *floding*).



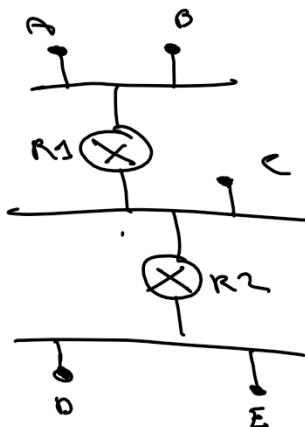
Esempio: supponiamo di avere un router che effettua il forwarding di un pacchetto multicast. L'IP del router è 130.192.1.1/24 e MAC address AA:...:AA mentre un host connesso al router possiede indirizzo IP 130.192.0.1/24 e MAC BB: ... :BB e questo host genera traffico multicast verso 224.1.1.1. Il pacchetto che viaggia nella rete inferiore (guarda foto precedente per chiarimento). Quel pacchetto sarà composto come segue:

MAC SRC	MAC DST	IPS	IPD
AA: ... : AA	01:00:5E:01:01:01	120.192.0.1	224.1.1.1

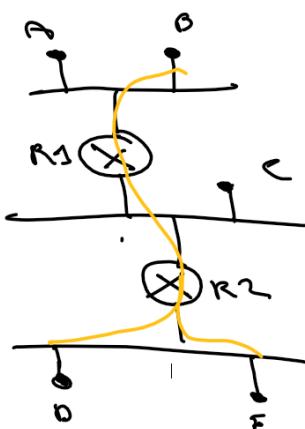
Si nota che anche il MAC address diventa un indirizzo multicast.

Facendo un po' di prove è possibile notare che l'indirizzo IP 224.1.1.1 e 224.129.1.1 vengono mappati sullo stesso indirizzo MAC multicast. In una stessa rete gli host possono dover processare non solo i pacchetti multicast di proprio interesse, ma anche quelli di altri gruppi per cui il MAC multicast risulta lo stesso, ma ciò è molto raro.

Per poter effettuare il **join** di un gruppo multicast è necessario utilizzare **IGMP** (*Internet Group Management Protocol*). Questo sostanzialmente informa i router di inoltrare un certo gruppo multicast su una certa LAN.



Esempio: quando C è interessato a un certo gruppo multicast, C informa R1 ed R2 di essere interessato a un gruppo multicast tramite una trama IGMP. Se D invia traffico multicast, questo invierà il traffico nella rete che andrà a finire anche su R2 ed il router, avendo ricevuto il messaggio da C di voler partecipare al gruppo, effettuerà il forwarding anche sulla rete centrale sempre con il MAC multicast (R1 e C). Se anche B è interessato a farne parte, R1 avendo ricevuto la richiesta da B farà l'inoltro anche sulla rete superiore (A e B). Ciò però non è sufficiente, perché il traffico inviato da D deve essere ricevuto da R2 per poterlo inoltrare sull'altra rete ma R2 non ha effettuato alcuna join (e non la effettua), dunque le interfacce dei router di default **ricevono tutti i gruppi multicast**.



Esempio: riprendendo l'esempio precedente, se B,D ed E effettuano il join a un gruppo multicast ma C non lo effettua, il traffico multicast non potrebbe essere ricevuto. È necessaria quindi una "intelligenza superiore" oltre ad IGMP che generi l'albero multicast e che permetta che i router si scambino informazioni tra di loro. L'albero in questo esempio sarebbe quello in foto. Tra i due router però non sarà presente IGMP, poiché quest'ultimo serve solo per effettuare la join. Per permettere ai router di scambiarsi informazioni tra loro si utilizzano i **Multicast routing protocols**.

IP Version 6 (IPV6)

Una nuova versione di IP si è resa necessaria perché permette di avere uno spazio di indirizzamento molto più grande (rispetto ai 2^{32} di IPV4). Altri possibili motivi possono essere:

- IPV4 non è Plug and Play, ma è necessario effettuare una configurazione manuale, mentre su IPV6 c'è una configurazione automatica degli indirizzi, ma questo problema in IPV4 è risolto tramite i server DHCP.
- Anche la crittografia in IPV4 è possibile implementarla (mentre in IPV6 c'è già)

Anche molte altre motivazioni che potrebbero “avvantaggiare” IPV6 sono in realtà già implementate in IPV4, come:

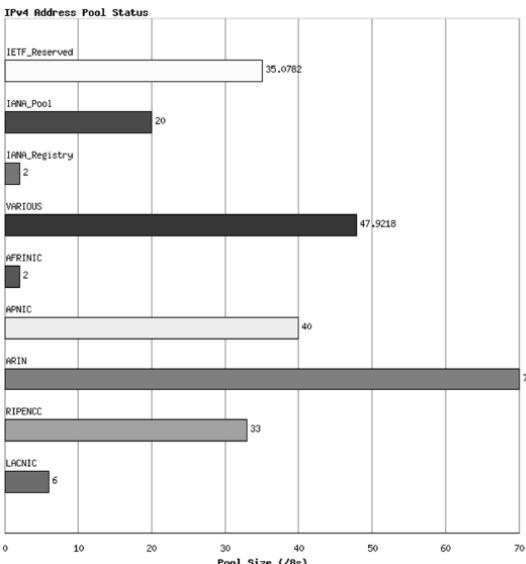
- *Efficienza nelle reti LAN*
- *Multicast ed anycast*
- *Politiche per il routing*
- *Differenziazione del traffico*
- *Mobilità*
- *Supporto al QoS*

In sostanza, IPV4 e IPV6 sono evoluti insieme. Per adottare IPV6 è stato necessario molto tempo perché migrare non è così semplice, ed inoltre visto che non vi è necessità di fare il passaggio e visto che le cose per il momento funzionano, IPV6 difficilmente riesce ad essere ampiamente adottato. Inoltre cambiare il livello 3 può avere effetti anche al livello applicativo: ad esempio i socket sfruttano gli indirizzi IP, e dunque è necessario cambiare anche il socket.

Anche se gli indirizzi IPV4 sono oltre 4 miliardi, bisogna comunque ricordare che ad esempio gli indirizzi di classe D sono esclusi per via degli utilizzi multicast, e ne restano dunque 3.5 miliardi. Inoltre, gli indirizzi IP sono distribuiti in maniera gerarchica ma un tempo gli indirizzi erano assegnati in maniera scorretta. Per tale motivo, gli indirizzi IP stanno terminando.

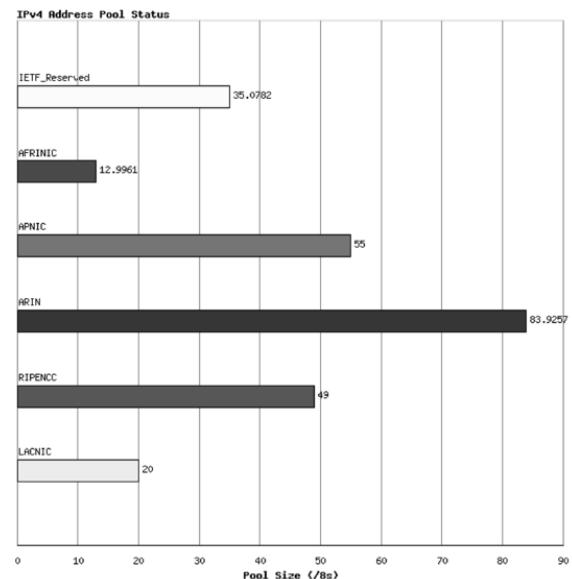
Il vero problema è che esistono tanti indirizzi IP allocati ma non utilizzati. Il piano di indirizzamento IP è di tipo gerarchico, il che vuol dire che per tutti i nodi all'interno di una certa area (intesa come rete fisica) ci vogliono indirizzi con lo stesso prefisso. In sostanza, se si allocano troppi indirizzi a tale rete fisica quegli indirizzi non potranno essere utilizzati da altre parti e sono dunque sprecati. Ed è ciò che è successo prima che introducessero l'indirizzamento gerarchico vero e proprio (possibilità di avere un grosso address range che viene diviso in tante reti più piccole).

Esistono soluzioni al problema della saturazione dello spazio di indirizzamento: uno è l'introduzione delle maschere che ha mitigato il problema, l'altro è l'indirizzamento privato con l'unione del NAT.



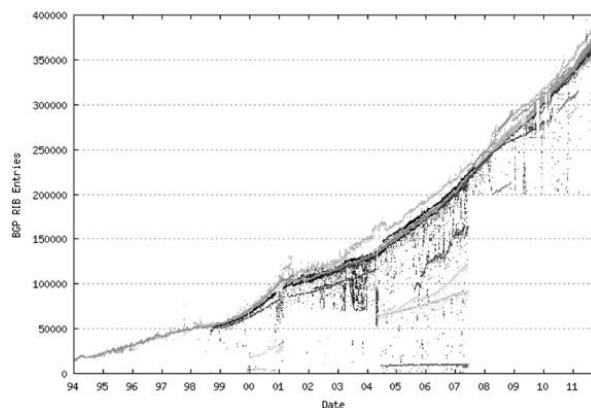
L’organizzazione IANA ha il compito di distribuire le reti IPV4/IPV6 ma non lo fa in maniera centralizzata. Si è deciso che lo IANA distribuisse degli address range /8 IPV4 a delle sotto-organizzazioni che vanno a distribuire gli indirizzi all’interno delle varie aree geografiche (ad esempio RIPE si occupa dell’Europa).

Il pool ancora in dotazione allo IANA nel 2010 era di circa 20 address range /8 (seconda riga), mentre la distribuzione delle /8 nelle varie zone è segnata nelle righe successive. ARIN è quella col maggior numero poiché si tratta dell’organizzazione del Nord America, mentre l’APNIC ne aveva 40 (che si occupa del sud-est asiatico).



Analizzando l’anno successivo si nota che la riga per lo IANA è scomparsa, e quelle reti sono finite in America e nel sud-est asiatico, dove si sono sviluppati tutta una serie di internet service provider che prima non c’erano. Il problema è nato quando l’APNIC ha terminato tutte le 55 reti che nel 2011 erano ancora disponibili, dandole a dei provider nati in tale zona. Ciò non vuol dire che gli indirizzi sono terminati, ma soltanto che sono stati allocati. Un nuovo ISP che vuole indirizzi in Cina non può ottenerli in IPV4. Le soluzioni possono essere quelle di utilizzare indirizzamenti privati all’interno della rete del provider. L’utilizzo del NAT sul provider porta a delle disottimizzazioni: sono presenti NAT multipli (sia sul provider che all’interno delle sotto-reti) e spesso a causa di ciò alcuni servizi funzionano male; non è possibile fare port forwarding sul NAT centralizzato.

L’alternativa è creare una rete IPV6: ai nuovi ISP l’unica strada percorribile è quella di creare una rete in IPV6. La rete del provider viene dunque gestita in IPV6, mentre le reti ed internet sono ancora in IPV4. Bisogna dunque trovare un modo di far comunicare tra loro i due diversi protocoli.



Tutto ciò detto prima si unisce alla mancanza di molte reti nel rispettare le regole di indirizzamento gerarchico (ad esempio quando si sposta un pezzo di address range da un’altra parte) che genera un aumento della grandezza delle routing table (dimensionamento elevato). Questo perché tutte le sotto-reti devono essere annunciate e in alcuni casi è possibile farlo utilizzando il grosso address range, ma in altri casi non è possibile farlo. Tutto ciò comporta un accrescimento della dimensione della tabella di routing, mostrato in figura (400.000 entry nel 2012).

Conclusione: gli indirizzi stanno terminando e il routing non scala.

Si è iniziato a parlare di IPV6 nel 1992 quando l'IETF ha iniziato a standardizzare IPV6. Una delle proposte è stata quella di usare un protocollo della OSI CLNP. Tra le altre proposte esiste anche SIPP che prevedeva una naturale evoluzione di IPV4, che è stata la più sensata ed è la proposta che ha vinto. Essendo gli indirizzi IPV4 un problema si sono “allungati” i campi per l’indirizzamento portandoli a 128 bit.

Indirizzi IPv6

Per selezionare il numero di indirizzi che IPv6 doveva avere, si è utilizzato un approccio scientifico calcolando l'**efficienza di indirizzamento**, calcolata come

$$H = \frac{\log_{10}(\text{number of addresses})}{\text{number of bits}}$$

Utilizzata per calcolare tutte le reti attive nel mondo (non solo la rete IP ma anche la Apple Talk e altre reti IBM). In sostanza, l'efficienza era molto bassa (variante tra 0.22 e 0.26) e dunque il numero di indirizzi sprecati in queste reti era molto elevato. Assumendo che l'efficienza resti tale nonostante il nuovo protocollo, assumendo anche un numero elevato di indirizzi, è venuto fuori che il numero di bit necessario è di 68 bit. Non essendo però una potenza di 2 byte, invece di utilizzare 64 bit che è inferiore al risultato ottenuto, si è sfruttato la potenza successiva: **128 bit**.

Con 128 bit non è più possibile utilizzare la notazione decimale puntata, ma al suo posto si utilizzano 8 numeri esadecimali divisi da “:” (a gruppi di 2). Esistono però delle scorticatoie: gli zeri più significativi di ogni gruppo possono essere omessi, e inoltre i gruppi di 0 contigui si possono omettere inserendo direttamente “::” (non più di una volta).

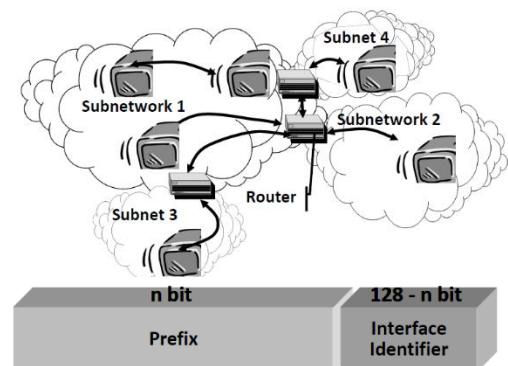
- **Notazione:** 1080:0000:0000:0007:0200:A00C:3423:A089
- **Notazione con gli 0 più significativi omessi:** 1080:0:0:0:7:200:A00C:3423
- **Notazione con gruppi di 0 contigui omessi:** 1080::7:200:A00C:3423

Organizzazione dello spazio di indirizzamento

Il **multicast** è posizionato al fondo dello spazio di indirizzamento: sono tutti quelli che iniziano con due blocchi da 1 sul primo byte (e dunque con **le prime due cifre decimali FF**). Scrivo FF00 per completare il primo gruppo e poi inserirò tutta una serie di 0 e tutto ciò che sta dopo al prefisso identificherà gli indirizzi multicast. Un esempio: FF00 ::/8.

Indirizzi Host – Routing e principi di indirizzamento

Anche in IPv6 sono presenti delle subnet e i principi per il routing sono identici a quelli su IPv4. Anche in IPv6 è presente un prefisso di rete detto **prefix** ed una parte di host detta **Interface Identifier**. La divisione tra queste due parti non è arbitraria, ma la cosa apparentemente strana è che per il momento l'interface identifier è fisso a **n = 64** bit (perciò all'interno di ogni II saranno presenti 2^{64} indirizzi). Ciò può sembrare poco ottimizzato, ma su IPv6 vi sono talmente tanti indirizzi che è permesso sprecarli per semplificarsi le cose. Come detto in precedenza, le **sub network** sono gli insiemi di host con lo stesso prefisso, mentre le reti fisiche vengono dette **link**. Vale sempre la corrispondenza biunivoca tra subnetwork e link.



- La comunicazione diretta tra host dello stesso link vengono dette **on-link**.
- Quando invece la comunicazione avviene attraverso un router (host su diversi prefissi) effettuano una comunicazione **off-link**.

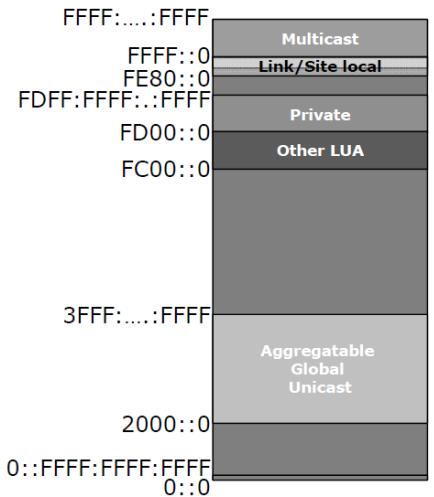
Il prefisso si gestisce allo stesso modo che su IPv4 ma **non** esistono più le maschere. Si utilizza solo il **prefix length**. Esempio: *FEDC:0123:8700::/40* (*ogni lettera/numero sono 4 bit*).

Subito dopo gli indirizzi multicast, sono stati introdotti gli *indirizzi privati (link local o site local)* e sono l'equivalente degli indirizzi privati in IPv4. I Link local/site local si identificano dai primi 9 bit definiti come segue: 1111 1110 1 e se ne distinguono 3 tipi:

- **Link local** posseggono uno 0 sul 10° bit 1111 1110 10
- **Site local** posseggono un 1 sul 10° bit 1111 1110 11

I link local sono degli indirizzi IP (inesistenti su IPv4) che vengono assegnati automaticamente alle interfacce sempre e con questi link è possibile effettuare **solo** le comunicazioni on-link, ovvero hanno validità solo all'interno del link.

Gli indirizzi site local hanno validità anche all'esterno del link (se si vuole) ma hanno la stessa validità degli stessi indirizzi IPv4, dunque se si vuole uscire dal proprio link con il proprio indirizzo privato lo si può fare solo se nell'altra rete si stanno usando indirizzi privati diversi, altrimenti si avrebbe una sovrapposizione di spazio di indirizzamento.



Il link local è una aggiunta fatta per inserire un modo per poter abilitare la stazione ad avere in maniera completamente automatica (plug and play) un indirizzo IP. Ciò può servire per accedere a dei servizi disponibili nella rete senza dover fare la configurazione IP.

I bit citati prima, con la nuova notazione esadecimale si traducono in:

- **Link local:** *FE80::/64*. Si utilizza /64 perché si identifica la network che si va ad associare ai singoli link. Sarà presente in tutte le reti IPv6.
- **Site local:** *FEC0::/10*. Dal bit 11 fino al bit 64 si può scrivere qualsiasi cosa, per poter definire varie reti IPv6 di tipo site local.

	10	38	64
site local	1111-1110-11	any	Interface ID
	FE[C-F]x		

	10	54	64
link local	1111-1110-10	0	Interface ID
	FE80		

Gli indirizzi site local sono però deprecati (sconsigliati) perché possono emergere alcuni problemi: se ad esempio si deve unire due reti (per esempio, acquisizioni di società) possono sorgere conflitti di indirizzamento (ad esempio tutti su IPv4 hanno come rete locale *192.168.0.0*). Può essere anche un problema nel caso in cui si trovano a dover comunicare due host con lo stesso prefisso ma su reti diverse (perché magari entrambe hanno scelto lo stesso indirizzo site local). Per risolvere questo problema esiste un 3° modo di indirizzamento privato:

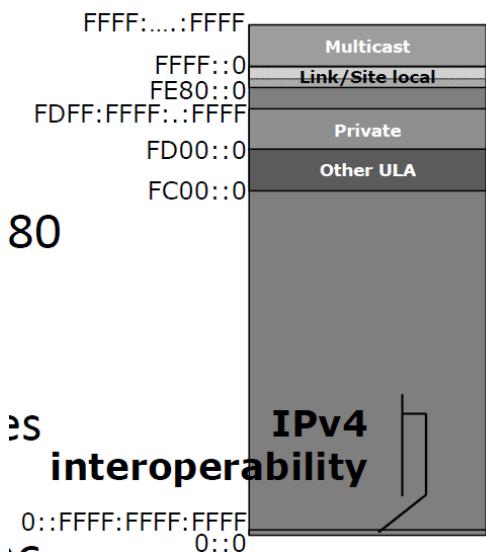
- **Unique local addresses (ULA)** con indirizzi *FC00::/7* che in binario significa 1111 110.

La metà di questi indirizzi /7,

nello specifico $FC00::/8$ (la prima metà) è riservata per utilizzi futuri. Nella seconda

metà ($FD00::/8$) sono stati inseriti, nei primi 40 dei 56 bit rimanenti, dei bit generati casualmente. Esiste un algoritmo che bisogna applicare in fase di definizione di queste reti private che, sulla base di alcune chiavi di inizializzazione, genera una stringa di 40 bit che con altissima probabilità genera una stringa univoca. Non è sicuro che due reti non avranno mai lo stesso prefisso, ma è altamente improbabile. I 16 bit rimanenti (non generati casualmente) sono stati lasciati per creare delle sottoreti. Con questo metodo si risolve il problema di unire due reti menzionato nel caso degli indirizzi site local.

Global Unicast (indirizzi pubblici)



80

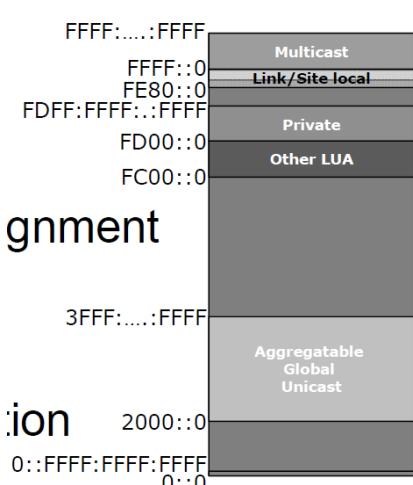
Dagli indirizzi Global Unicast è stata rimossa una parte iniziale per definire degli indirizzi “speciali” che servono per garantire l'**interoperabilità con IPv4**. Se si ha un indirizzo IPv4 che deve essere raggiunto da una stazione IPv6, è possibile (in alcuni casi) dover rappresentare la rete IPv4 nella corrispettiva rete IPv6. Il prefisso è $0::/80$. Esistono vari modi per gestire i bit che rimangono per arrivare ai 32 dell’indirizzo IP e ne sono stati definiti un paio: IPv4-mapped e IPv4-compatible.

Con **IPv4-mapped** si aggiungono altri 16 bit (per arrivare a 96) che sono tutti 1. In questo modo si arriva a $0:0:0:0:0:FFFF::/96$ e successivamente dopo i 96 andranno aggiunti i 32 bit dell’indirizzo IPv4.

Gli **IPv4-compatible** inseriscono invece altri 16 bit a 0 per arrivare a $0::/96$ (e poi si procede come prima). Un esempio: $0:0:0:0:0:A00:1$ i cui ultimi 32 bit in notazione decimale viene tradotto in 10.0.0.1 ($0A00:0001$). Chiaramente la notazione dell’esempio può essere scritta in forma compatta come $::A00:1$ oppure è possibile scriverla come $::10.0.0.1$. Nel caso dei mapped la notazione speciale è definita come $::FFFF:10.0.0.1$.

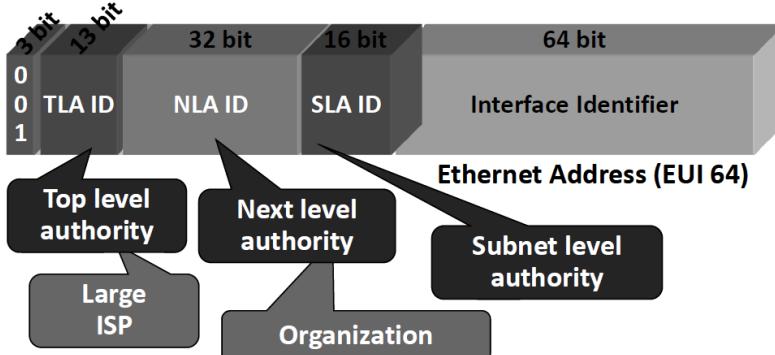
In totale esistono 2^{16} modi per mappare un indirizzo IPv4 in un indirizzo IPv6.

Messi da parte gli indirizzi speciali per l’interoperabilità con IPv4, tutta la parte restante è dedicata ai Global Unicast. Per il momento si sta utilizzando solo una parte degli indirizzi IPv6, in particolare quelli che in binario iniziano con 001 (cioè quelli che sulla prima cifra hanno 2 o 3) e nessun altro [nella foto indicati con **Aggregatable Global Unicast**]. Tutti gli altri indirizzi non sono allocati.



Questi indirizzi, proprio per risolvere i problemi di IPv4, sono assegnati per via topologica ma rispettando una gerarchia di service provider. In questo modo si vuole evitare che le routing table siano corpose, anche se, non essendo la topologia completamente ad albero, il problema sarà comunque presente (ma ridotto).

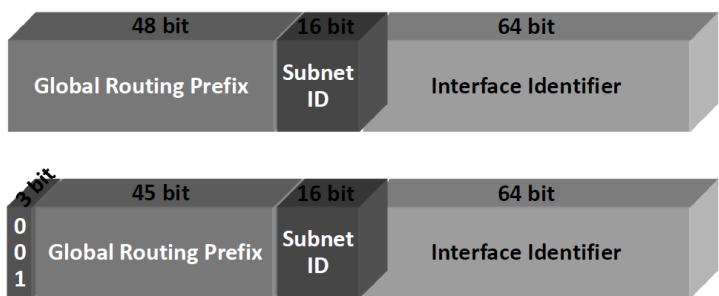
Per garantire la gerarchia multilivello all'inizio è stata pensata una struttura di assegnamento ben definita e rigida.



utilizzando 16 bit per dei provider ancora più piccoli. I restanti 64 bit invece identificano l'Interface Identifier.

Questa struttura è stata però abbandonata perché troppo rigida e si è passati a una struttura più flessibile.

I primi due campi adesso sono stati “uniti” e dunque lo IANA definirà un opportuno prefisso per i provider più grandi, il quale al suo interno potrà definire degli address range da assegnare ai provider più piccoli. Rimangono i 16 bit che i provider possono utilizzare per creare delle reti da assegnare ai clienti.



Special Addresses

Come in IPv4 sono presenti alcuni indirizzi speciali.

- **Loopback address** → indirizzo con un unico 1 finale, definito come `::1`
- **Non** esiste più il broadcast, ma si tenta di risolvere tutto con il multicast.
- **All nodes (multicast) address** → definito come `FF02::1`
- **All routers (multicast) address** → definito come `FF02::2`
- **Unspecified address** → definito come `::`

Protocolli modificati

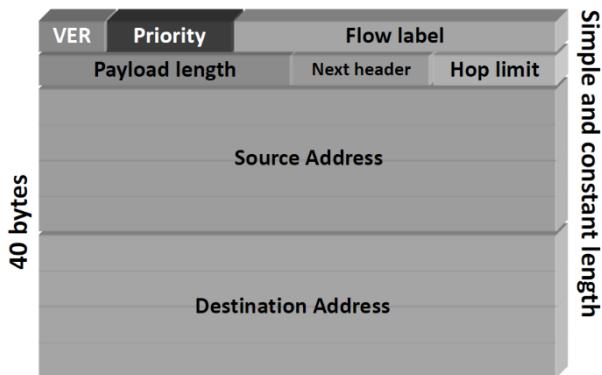
Con il passaggio da IPv4 a IPv6 sono stati modificati alcuni protocolli:

- IP → da IPv4 a IPv6
- ICMP → diventato ICMPv6
- ARP → non più esistente, ma integrato in ICMP
- IGMP → integrato in ICMP

Infine, sono stati aggiornati ma non modificati i seguenti protocolli:

- DNS → è stato introdotto il record con AAAA (che vuol dire “fornire indirizzo IPv6) mentre la singola A era riferita a IPv4.
- RIP e OSPF (protocolli di routing)
- BGP e IDRP (protocolli di routing)
- TCP e UDP → TCP deve fare demultiplexing utilizzando gli indirizzi IP e dunque va opportunamente modificato per adattarlo a IPv6.
- Socket interface → stesso motivo di cui sopra

Formato dell'header



L'header IPv6 è molto più semplice di quello IPv4 ma anche più lungo. Si tratta di un header a 40 bytes a lunghezza fissa poiché non sono più presenti le opzioni.

- **Priority** → equivalente del ToS
- **Flow label** → serve per identificare i flussi in maniera rapida. Un host alla creazione di una nuova sessione definisce un nuovo numero (che cambia sempre in base a ogni nuova sessione) e per ogni coppia sorgente-flow label si è sicuri di identificare un flusso diverso (una stessa sorgente non creerà mai due flussi con lo stesso flow label)
- **Payload length** → dimensione del Payload che sta all'interno del datagram IPv6
- **Hop limit** → equivalente del Time To Leave
- **Next header** → novità di IPv6. Non essendoci più le opzioni e i campi per la frammentazione (perché sostanzialmente non si usa) il tutto viene gestito da questo nuovo campo. Con il next header si va ad aggiungere altri header (esterni a IPv6), ma da un punto di vista del processing sono presenti dei vantaggi (ad esempio si evita il campo header length visto che è a lunghezza fissa).

È stato rimosso anche il campo checksum perché non necessario (chi può introdurre errore come il Wi-Fi ha già un checksum).

Gli **extension headers** vengono aggiunti solo se necessari e non processati ad ogni pacchetto. Gli extension headers sono:

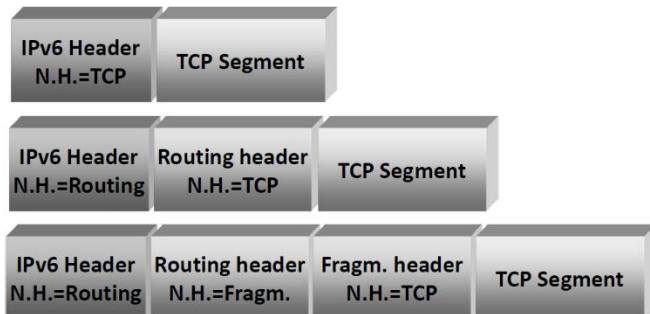
- Hop By Hop Option → equivalente delle opzioni del datagram IPv4
- Routing
- Fragment
- Authentication
- Encrypted Security Payload
- Destination Option

Gli extension headers vanno utilizzati nel modo in cui sono stati scritti (ad esempio se prima utilizzo l'encrypting dei dati e poi scrivo il routing non funzionerà). Il formato degli extension headers è il seguente:



Come primo campo è presente nuovamente il campo next header, per eventuali altri extension headers. Se non ve ne sono altri, tale campo punterà al contenuto del payload. In questo caso è presente anche il campo **length** perché non tutti gli extension header hanno la stessa lunghezza. Successivamente sono presenti i dati.

Header chaining



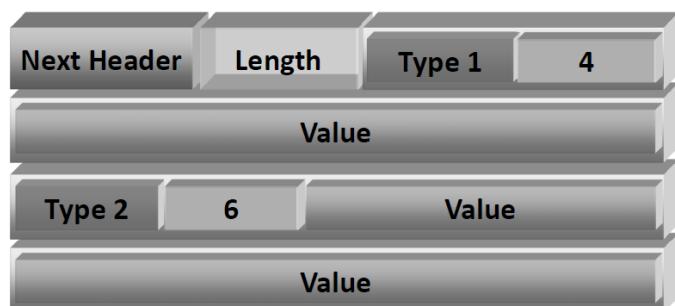
Questo è un esempio di come possono essere concatenati gli header IPv6.

Options



Le opzioni possono essere utilizzate con l'extension header **hop-by-hop** e viene definito così perché è possibile usarlo su

tutti i nodi, ovvero ad ogni hop se le opzioni sono presenti vanno considerate ed eventualmente modificate. Sono tutte standardizzate nel formato TLV (Type – Length – Value): tipo di opzione, lunghezza di tale opzione e il valore. Dentro l'extension header Options è possibile utilizzare più di una opzione.



Extension header option: il campo next header informa cosa ci sarà al fondo; sarà presente la lunghezza dell'intero header e poi sarà presente un primo campo Type che ha lunghezza 4 e dunque si sa che leggendo tale valore non corrisponde a length e dunque sarà presente anche un altro campo Type.

Esempi di possibili opzioni

- **Jumbo Payload** → serve per estendere la dimensione massima del Datagram.
- **Padding Options** → serve per mantenere l'allineamento a 64 bit degli extension header, per garantire un processing ottimale in base al parallelismo dei processori. È dunque presente l'opzione **PadN** (di Tipo 1) che dice quanti byte devono



essere aggiunti alla fine per mantenere l'allineamento. Infine, è presente l'option **Pad1** (di Tipo 0) che aggiunge un solo byte. Le options non devono essere necessariamente capite e gestite da tutti i router (anche perché il meccanismo TLV è stato creato per aggiungere facilmente ulteriori opzioni).

I primi due bit del campo Type forniscono una certa informazione utile al router nel caso in cui l'opzione non è riconosciuta, nello specifico:

- **00** → L'opzione può essere ignorata e si può procedere con la successiva.
- **01** → Il pacchetto deve essere scartato
- **10** → Il pacchetto deve essere scartato e bisogna generare un messaggio di errore ICMPv6
- **11** → Il pacchetto deve essere scartato e bisogna generare un messaggio di errore ICMPv6 solo se la destinazione non è di tipo multicast.

Invece, il terzo bit del campo Type indica se l'opzione può essere modificata on-the-fly:

- **0** → L'opzione non può essere modificata on-the-fly
- **1** → L'opzione può essere modificata on-the-fly

Routing Header

Il routing header serve per effettuare il **source routing**. Il routing in IPv4 è basato sulla destinazione, perciò, questo protocollo può sembrare “strano”. A livello geografico, infatti, il source routing non viene mai utilizzato ma è comunque importante sapere che in ambito Data Center, per fare ciò che viene detto **service chaining**, oggi il source routing è utilizzato. Il service chaining è la creazione di catene di servizi. Per far andare il pacchetto nel posto in cui si preferisce il source routing può essere un’idea. Il source routing dunque può essere utile.

Indipendentemente dal tipo di implementazione, nel source routing il pacchetto viene inviato al router e all'interno è contenuto l'indirizzo a cui forwardare il pacchetto (non si usa più la tabella di routing), e dunque il **destinatario del pacchetto è il router**.

Interfacciamento con il livello 2

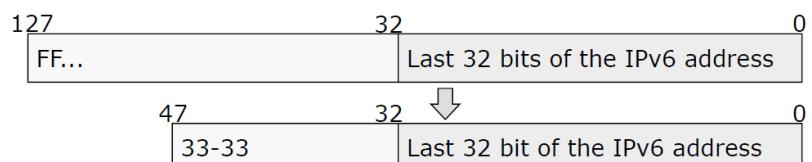
Con IPv6 viene utilizzato un EtherType diverso da quello per IPv4 (86DD). Questo è stato fatto per poter gestire IPv6 in maniera totalmente indipendente da IPv4. Questo permette l'approccio **dual stack**, ovvero vengono mantenuti due “stack” di rete: uno TCP-IPv4 e l’altro TCP-IPv6.

Il mapping tra indirizzo IPv6 e MAC avviene come segue:

- Se un indirizzo è **unicast**: esiste una procedura detta **Neighbor Discovery**
- Se un indirizzo è **multicast**: si utilizza un algoritmo di **mapping**.

Trasmissione Multicast IPv6

Se si ha un indirizzo IPv6 multicast da mappare sul corrispondente indirizzo MAC multicast lo si fa utilizzando una stringa iniziale che definisce i gruppi multicast: **33-**



33. Successivamente vengono aggiunti i 4 byte meno significativi dell'indirizzo IPv6 (in pratica gli ultimi 32 bit, cioè metà dell'Interface Identifier).

Ad esempio, se si vuole inviare un pacchetto all'indirizzo IP multicast *FF0C::89:AABB:CCDD* questo viene encapsulato in un frame MAC come *33:33:AA:BB:CC:DD*.

Neighbor Discovery

La Neighbor Discovery sostituisce ARP ed è basata sul multicast. Ciò implica che in teoria solo una stazione viene coinvolta da questo procedimento, ma non è sicuro.

Quando si vuole creare un mapping tra indirizzo IP e corrispondente indirizzo MAC per effettuare una consegna, bisogna calcolarsi il **Solicited Node Multicast Address**. Si utilizza la seguente stringa: *FF02::1:FF/104 + 24 bit meno significativi dell'indirizzo IP*. È molto probabile (ma non sicuro) che ogni stazione sia appartenente a un gruppo multicast diverso.

Ogni nodo si deve “iscrivere” tramite questo Solicited Node ma, visto che vengono utilizzati solo i 24 bit meno significativi, è molto probabile che per ogni nodo il gruppo sia diverso.

Se in un link vi sono 100 nodi, è probabile ma non sicuro che vi siano 100 gruppi diversi chiamati Solicited Node Multicast Address.

Questo gruppo serve per effettuare l'**Address Resolution**. Viene preso il pacchetto **ICMP Neighbor Solicitation** e viene inviato utilizzando come indirizzo IP destinazione il Solicited node multicast address. Essendo probabile che vi sia un unico host in tale gruppo, è dunque altamente probabile che sia proprio la stazione che si vuole interrogare. Chi riceve la Solicitation se questo risulta iscritto al gruppo la processa, altrimenti non viene processata. In caso positivo, si risponde con il pacchetto **ICMP Neighbor Advertisement** inviato in unicast e inviato a chi ha effettuato la domanda (se per caso esistono due host con lo stesso indirizzo nello stesso link entrambi processeranno il pacchetto).

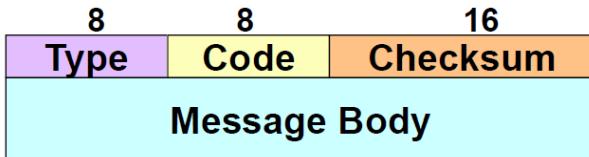
Esempio

- Si vuole trovare l'indirizzo MAC dell'host *2001::ABCD:EF98*
- Il relativo Solicited Node Multicast Address è *FF02::1:FF:CD:EF98*
- Viene encapsulato in un frame MAC come *33:33:FF:CD:EF:98*

Il pacchetto viene poi inviato e una volta ricevuto riscontro, il mapping tra indirizzo IP e indirizzo MAC di un host viene salvato nella **Host Cache** che è equivalente alla Cache ARP.

ICMPv6 – Internet Control Message Protocol version 6

ICMPv6 non è molto diverso da ICMPv4: ICMPv4 copre un sottoinsieme degli scopi per cui è stato realizzato ICMPv6. Il formato è identico, ma sono state aggiunte infatti alcune funzionalità come la **Neighbor Discovery** e il **Multicast group management**. Le funzioni incluse da IPv4 sono **ARP** e **IGMP**.



Nella struttura sono presenti i campi **Type** e **Code** per definire i tipi e i sottotipi messaggi. Il **Message Body** è presente per aggiungere ulteriori funzioni (opzioni) a seconda del particolare messaggio ICMP. Nei pacchetti ICMP di errore all'interno del Message Body si trova il

pacchetto di errore che ha generato tale messaggio (per esempio *network unreachable* su un router che invia indietro il pacchetto ICMP indicando cosa ha causato l'errore). In altri messaggi il Message Body avrà uno scopo più importante come nella Neighbor Discovery, dove andrà scritto l'indirizzo MAC. È stato standardizzato in modo tale che la lunghezza max sia **576 bytes**, dimensione ragionevole per essere sicuri di non dover frammentare (IPv6 parte dal presupposto che non si debba frammentare).

Campo Type

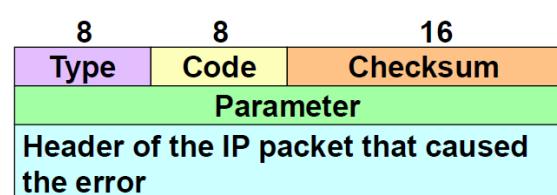
I principali tipi sono i seguenti:

- 1 Destination Unreachable
- 2 Packet too big
- 3 Time exceeded
- 4 Parameter Problem
- 128 Echo Request
- 129 Echo Reply
- 130 Multicast Listener Query
- 131 Multicast Listener Report
- 132 Multicast Listener Done
- 133 Router Solicitation
- 134 Router Advertisement
- 135 Neighbor Solicitation
- 136 Neighbor Advertisement
- 137 Redirect

Molti dei tipi esistevano già, con l'aggiunta di ulteriori tipi che implementano IGMP e la Router Solicitation/Advertisement.

Messaggi di errore

- Destination Unreachable (Type = 1)
- Packet too big (Type = 2)
- Time exceeded (Type = 3)
- Parameter Problem (Type = 4)



Il **Parameter Problem** serve quando si genera un errore perché è presente qualche campo del pacchetto che il nodo non è in grado di capire. Il pacchetto viene scartato e si invia il pacchetto ICMP con il Parameter Problem, indicando l'header del pacchetto che ha causato l'errore (vedi figura).

Echo messages

- Echo request (Type = 128)
- Echo reply (Type = 129)

Type	Code	Checksum
Identifier	Sequence Number	
Data		

Gli Echo messages sono simili a ICMPv4. Il parametro

Sequence Number serve per identificare a quale pacchetto fa riferimento la Request (*associa Request/Reply*).

Neighbor Solicitation

Type	Code	Checksum
Reserved		
Target Address		
Options		

Il campo **reserved** contiene dei bit introdotti per mantenere l'allineamento a 64 bit, per gestire meglio la ricezione dei pacchetti sull'host. Potrebbe anche essere chiamato **padding**. Il **target address** è lo stesso dell'ARP Request, cioè l'indirizzo IP per cui serve l'indirizzo MAC, un tempo contenuto nella ARP Request ma ora inserito dentro ICMP e detto **Neighbor Solicitation**. Il target address è un campo da 128 bit (per essere compatibile con gli indirizzi IPv6). I campi nella prima riga della foto sono 32 bit e per mantenere un allineamento fatto da "slot" di 64 bit sono stati aggiunti 32 bit di campo Reserved.

Neighbor Advertisement

I 3 flag indicano:

- **R** se la Neighbor Advertisement arriva da un router. Può essere utile saperlo.
- **S** sta per **Solicitation**, dice se la Advertisement è stata generata in risposta a una Solicitation. Di solito è così ma c'è l'opzione perché un Host potrebbe effettuare un advertisement in maniera asincrona (può essere utile per qualche applicazione) per informare la rete.
- **O** serve per sovrascrivere l'informazione contenuta nella host cache, che è l'equivalente dell'ARP cache. In ARP questa opzione non c'era ed **ogni** ARP reply andava a sovrascrivere il contenuto.

Type	Code	Checksum
R	s	O
Reserved		
Target Address		
Options		

Dopo i flag viene ripetuto il Target Address, solo per gestire meglio la ricezione e il processing dei pacchetti. Quando il modulo ICMP elabora la Advertisement ricevuta, si trova direttamente all'interno del pacchetto i bit che contengono l'informazione su qual era l'indirizzo IP a cui corrisponde l'indirizzo MAC che sarà scritto all'interno del pacchetto. Questo evita di andarlo a leggere dall'indirizzo IP sorgente.

Il campo **options** è inserito per contenere la risposta alla domanda (cioè l'indirizzo MAC che viene trasportato dal Neighbor Advertisement, che corrisponde alla ARP Reply).

Si nota dunque che non si tratta più di una semplice trama a livello 2 con ARP che gestisce il mapping. ICMP viaggia dentro IPv6 quindi non è più detto che sbagliando le configurazioni a livello 2 tutto continua a funzionare.

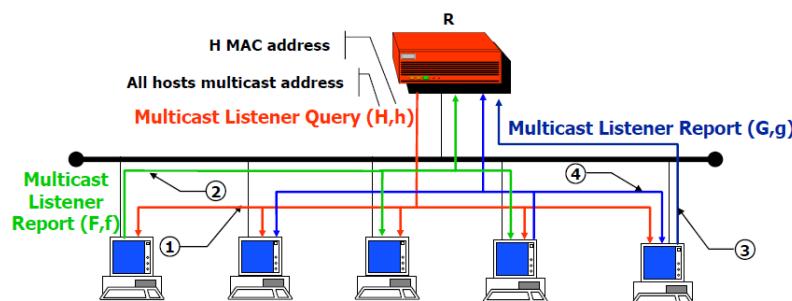
Multicast Communication in the Internet

Per gestire la comunicazione Multicast nella rete locale bisogna effettuare il mapping dell'indirizzo MAC multicast. Per fare la join dei gruppi all'interno di una rete locale si usava IGMP (e servivano anche i protocolli di routing multicast per gestire l'albero su scala geografica). Tutto rimane ancora valido ma adesso si utilizza ICMPv6 invece di IGMP, che svolge questo lavoro attraverso dei messaggi: **Multicast Listener Query**, **Multicast Listener Report** e **Multicast Listener Done**. Questi tre messaggi sono identici a IGMP.

Type	Code	Checksum
Maximum Response Delay		Unused
Multicast Address		

Host Membership Discovery

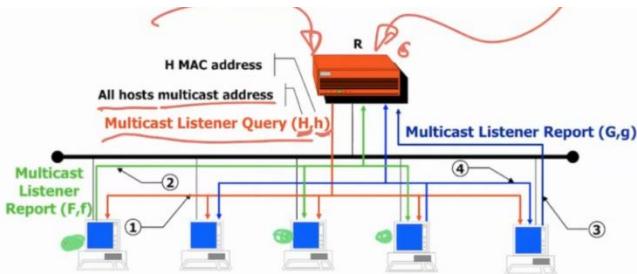
Tutto si basa sull'invio da parte di un router del messaggio **Multicast Listener Query**. Serve per chiedere all'interno della rete se sono presenti dei nodi interessati a un certo gruppo multicast. Supponiamo che R riceva da "fuori" un certo messaggio multicast. R può dunque effettuare una **Multicast Listener Query Specifica** per un determinato gruppo multicast. È però possibile effettuare anche una **Multicast Listener Query Generale**. La query generica serve per chiedere agli host a quali gruppi sono interessati in generale.



motivo, il router invia un messaggio in cui l'indirizzo IP destinazione è esattamente l'AHMA, indicato in figura con **H**. Questo pacchetto viene ricevuto da tutti gli host, evitando di inviarlo ai router. Al livello 2, l'indirizzo MAC address corrispondente utilizzato verrà generato con la procedura di mapping vista in precedenza (**h** piccolo in figura).

La Listener Query generica deve raggiungere tutti gli host. La query non è però un messaggio broadcast, ma in IPv6 si cerca di fare tutto col multicast. È possibile, infatti, utilizzare l'indirizzo speciale **"All Hosts Multicast Address"** ($FF02::1$) da inserire come indirizzo IP destinazione. Per tale motivo, il router invia un messaggio in cui l'indirizzo IP destinazione è esattamente l'AHMA, indicato in figura con **H**. Questo pacchetto viene ricevuto da tutti gli host, evitando di inviarlo ai router. Al livello 2, l'indirizzo MAC address corrispondente utilizzato verrà generato con la procedura di mapping vista in precedenza (**h** piccolo in figura).

Alla query inviata dal router, gli host dovranno rispondere. In teoria ha senso che tutti quanti gli host non appena ricevono la Query rispondano con la relativa Report, perché in questo modo il router può avere la lista completa di tutti gli interessati. Ma nel tentativo di ottimizzare le trasmissioni, si è pensato che il router non dovesse possedere l'intera lista delle query con tutti i gruppi multicast, questo perché il router dovrà semplicemente forwardare nella rete il pacchetto multicast quando è presente almeno 1 host nel gruppo multicast (ma che ce ne siano 10 o 100 al router non cambia nulla, basta che ce ne sia almeno 1). È stato dunque inserito un **timer** randomico sugli host, che viene fatto partire quando viene ricevuta una query. Il primo host a cui scade il timer invierà indietro la Report, in modo tale che tutti gli altri interessati al gruppo multicast vedano che tale host l'ha inviata, utilizzando l'indirizzo multicast del gruppo.



se sono interessati a un certo traffico multicast, avranno modificato opportunamente la scheda di rete in modo da “accettare” tali pacchetti). Gli altri 2 host vedranno che qualcuno ha già risposto e allora fermeranno il proprio timer e non invieranno la Report. Il router, in questo modo, saprà che c’è 1 interessato al gruppo, anche se i messaggi verranno inviati a tutti gli altri. Quando uno degli host si disconnette invia una **Multicast Listener Done** per uscire dal gruppo. Ma a quel punto per il router non ci saranno più host interessati alla rete. Per tale motivo, quando si vede passare una Report, gli altri host interessati interrompono il timer solo dopo che sono passati N pacchetti con la Report. Ad esempio, in una rete locale, dopo che se ne vedono 10 passati, si interrompe il timer. Il problema può comunque presentarsi se tutti e 10 gli host vanno via, ma è molto più difficile che ciò succeda. Inoltre, per evitare che ciò succeda, il router in ogni caso periodicamente invia una **Listener Query** e la procedura riparte. I router tengono traccia di chi ha inviato il report per ottimizzarne la gestione. È un ulteriore modo per mitigare il problema.

Se un host lascia la rete, non sempre viene effettuato con una **Graceful Leave** (shutdown del sistema inviando una Done). Quando viene interrotta “di colpo” la comunicazione, la Multicast Listener Done non viene inviata. Il router, per gestire questa cosa, fa **scadere periodicamente le informazioni in tabella**. La lista viene refreshata dopo un certo lasso di tempo per capire se è ancora valida o meno, questo evento viene chiamato **Gestione soft-state** della tabella, spesso usato nei sistemi distribuiti.

Le Multicast Listener Report non sono sempre e solo generate in risposta alle query: se si effettua la join a un gruppo multicast è possibile dirlo subito al router in maniera asincrona (senza attendere la query dal router). In questo modo i nuovi host entrano immediatamente a far parte del gruppo di host.

Il Solicited Node Multicast Address è un gruppo multicast speciale, un gruppo a cui il singolo host si deve iscrivere per poter ricevere la Neighbor Solicitation. In linea teorica non ha dunque senso inviare una Report per quel gruppo multicast perché da “fuori” non arriverà mai traffico verso quel gruppo, visto che ha solo validità locale sul singolo host. Però i sistemi operativi in realtà inviano comunque la Report molto spesso, per semplificare il codice ed evitare di gestire le eccezioni.

Il Maximum Response Delay è una ulteriore opzione nelle Query per chiedere una risposta entro un certo numero secondi (non ha senso nelle Multicast Listener Report, ma il formato è stato mantenuto comunque).

Supponendo che gli host col pallino verde in foto siano interessati al gruppo rappresentato dall’indirizzo multicast generico **F**, tutti e 3 gli host una volta ricevuta la Query generica fanno partire il timer. Sul primo host il timer scade per primo, perciò questo invia la Multicast Listener Report che verrà ricevuto e processato anche dagli altri 2 host (perché

Configurazione dei dispositivi IPv6

Per configurare un dispositivo IPv6 servono le stesse cose di IPv4: un indirizzo IP, che in IPv6 si traduce in prefisso e identificatore di interfaccia. Serve anche il Default Gateway, un DNS server, Hostname (non obbligatorio) e Domain Name, ed una MTU (Maximum Transmission Unit).

Per fare ciò si può configurare il tutto manualmente oppure utilizzare un approccio plug and play di tipo **Stateful** (tramite DHCP, come su IPv4), ma la novità si ha con la **stateless configuration** e la **hybrid** configuration.

La **Stateless Configuration** prevede che l'host può configurare in automatico tutti i parametri già citati. Un esempio per un primo passo di *stateless configuration* potrebbe essere il fatto che "accendendo" un host viene automaticamente generato un indirizzo *link local* (che ha solo validità locale). Con questa configurazione si riesce a ottenere in modo automatico anche un prefisso da un router (che può essere privato, side local, global unicast), grazie a questa novità non è necessario neanche configurare l'host per potergli permettere di utilizzare il DHCP.

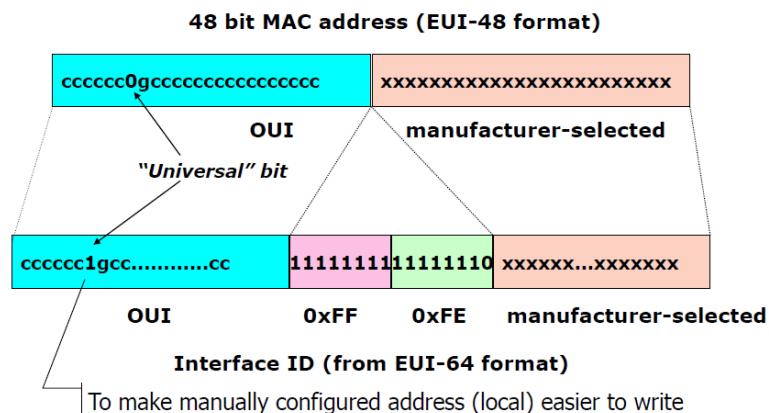
È possibile prendere un indirizzo *global unicast* fornito dal router che non deve essere necessariamente configurato dall'amministratore di rete, ma potrebbe essere comunicato da altri router (come i router del provider). I router del provider comunicano al router in esame l'indirizzo da utilizzare, il quale lo comunicherà a sua volta agli host nella rete. Solo il provider deve dunque configurare la rete (novità rispetto a IPv4).

La **Hybrid Configuration** è una via di mezzo: alcune informazioni vengono recuperate in maniera automatica mentre altre vengono recuperate col DHCP.

Interface identifier: per configuararlo è possibile *scrivere a mano* (64 bit da scrivere); si può *ottenere dal DHCPv6* (identico alla v4); si può *generare automaticamente* andando a prendere l'indirizzo MAC oppure generando una stringa di 64 bit in maniera automatica che garantisca la privacy. Si intende che anche cambiando connessione una parte dell'indirizzo rimane la stessa perché viene generata dal MAC address (che non cambia) e dunque chiunque sarebbe facilmente tracciabile. Per risolvere il problema si utilizza il metodo **privacy aware**.

Mapping da EUI-48 a EUI-64

EUI = Extended Unique Identifier



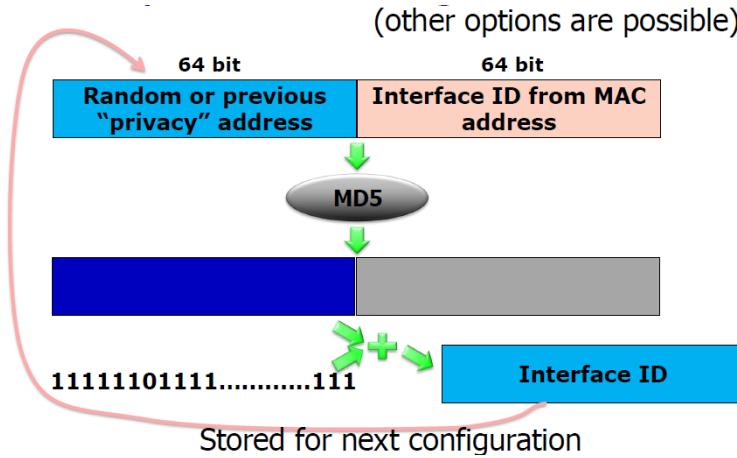
dichiarando che è un II ma che qualcun altro potrebbe utilizzarne un altro esattamente identico.

Viene utilizzato proprio il 7° bit per riprendere ciò che si fa negli indirizzi MAC: gli indirizzi MAC che hanno 0 sul 7° bit sono quelli univoci, mentre quelli che hanno 1 non sono univoci. Ciò è stato fatto perché se si vuole

Tra l'OUI, la prima parte dell'indirizzo MAC che identifica il vendor, e la seconda parte che è scelta dal costruttore, viene aggiunta una stringa di bit 0xFF e 0xFE per passare da 48 a 64 bit. In IPv6 è stato definito che il 7° bit dell'Interface Identifier vada a caratterizzare il fatto che tale II sia univoco o meno. In IPv6 si ha che se il 7° bit di un II è 1 si sta dichiarando che quell'indirizzo è univoco. Se invece vi è uno 0 si sta

configurare a mano (o tramite DHCP) l'II, definendo lo standard così ci si semplifica le cose nel fare un'operazione di questo tipo. Se devo scrivere un II a mano, visto che lo sto scrivendo io non posso garantire che è univoco e dunque si setta il bit. Questo metodo è **privacy unaware**: tutti possono leggere la stringa.

Privacy Extension Algorithm



Si concatenano 64 bit che possono essere: bit dell'Interface ID generato in precedenza oppure bit generati a caso. Questi 64 bit vengono concatenati con quelli del MAC address per ottenere un totale di 128 bit. Si utilizza il **MD5** per ottenere una stringa “criptata”. Anche se servono soltanto 64 bit si utilizza la concatenazione di ulteriori 64 bit per complicare a terzi la possibilità di scoprire l'ID. Si prende dunque la stringa di bit casuali generati dall'MD5 e si applica una maschera sui primi 64 bit, facendo attenzione a mettere a 0 il 7° bit. Questo perché non è possibile garantire che tale bit sia univoco.

maschera sui primi 64 bit, facendo attenzione a mettere a 0 il 7° bit. Questo perché non è possibile garantire che tale bit sia univoco.

Utilizzo degli indirizzi

Nel momento in cui si va a configurare una stazione IPv6 si avranno più indirizzi su una interfaccia, cosa che in IPv4 non veniva mai utilizzata. Su IPv6 invece si avranno almeno 2 indirizzi: un link local, configurato automaticamente; un indirizzo private; un indirizzo global unicast; indirizzo calcolato utilizzando il MAC come Interface ID (non il privacy aware), per avere un modo per essere tracciato a livello locale.

La scelta è lasciata anche all'utente/applicazione. Considerando ad esempio una rete aziendale, una certa applicazione per accedere a dei servizi interni può forzare e scegliere l'indirizzo IP.

Prefisso di un indirizzo

Per quanto riguarda il prefisso, le due opzioni sono le solite: è possibile **configurarla a mano** e assegnarsi ad esempio il prefisso 2001 (se si sa che quelli sono indirizzi global unicast) oppure ottenerli dal **DHCPv6**. La novità sta nel fatto che possono essere **generati automaticamente**, ovvero viene assegnato in modo automatico un indirizzo link local. Ma la funzionalità più interessante è la possibilità di **ottenere gli indirizzi dal router**. Per fare ciò viene utilizzato nuovamente **ICMPv6: Router Advertisement e Router Solicitation**. Sono dei messaggi che servono per configurare un prefisso sugli host tramite il router, in maniera completamente automatica.

- Dentro il messaggio **ICMP Router Advertisement** sono contenuti uno o più prefissi, che possono essere utilizzati nella rete ma non solo. Vengono inviati utilizzando l'indirizzo All Nodes. Vengono inviati periodicamente.
- Nel caso della **ICMP Router Solicitation** sono gli host a chiedere un indirizzo, invece di attendere l'Advertisement periodico. Visto che è un host ad aver fatto la domanda, in questo caso la Advertisement viene inviata con una trasmissione unicast (e non più tramite All Nodes).

Type	Code	Checksum
Reserved		
Options		

La Router Solicitation a sinistra ha un formato standard con un po' di opzioni. Nel momento in cui il router vede il Type sa già cosa deve fare perciò non sono contenute molte informazioni nel messaggio. Questa viene inviata utilizzando l'indirizzo multicast **All Routers** (viene inviata a tutti i router nella rete).

La Router Advertisement, invece, è molto più complicata e possiede tutta una serie di parametri [NON NECESSARI DA RICORDARE].

Type (134)	Code (0)	Checksum
Cur Hop Limit	M O Reserved	Router Lifetime
Reachable Time		
Retrans Timer		
Options		

A sinistra vi è lo schema di un Router Advertisement. Il parametro **Router Lifetime** viene utilizzato per comunicare all'host (o agli host) che si sta contattando che il router in realtà sta per essere spento. Il **Retransmission Timer** dice ogni quanti secondi la Advertisement viene reinvia.

Con il flag **M**, se settato a 1, il router informa che la configurazione va chiesta al server DHCP. Questo è interessante perché in questo caso, a differenza di IPv4, gli host non devono essere minimamente configurati, visto che in IPv6 l'interfaccia di rete va configurata per utilizzare il DHCP, altrimenti la DHCP Discovery non può essere inviata. Invece in IPv6 il funzionamento è diverso ed il comportamento standard dell'interfaccia è di inviare una Router Solicitation, oppure anche se non viene inviata si sa che verrà ricevuta una Advertisement. Se il router per qualche motivo non è stato configurato per offrire questi servizi lo comunica mettendo il flag M a 1. In altre parole, basta configurare solo il router e non più ogni singolo host.

Il flag **O**, che in genere si utilizza in combinazione con M, informa che è necessario ottenere le informazioni che il router non può dare dal DHCP. Ad esempio: dalla Router Advertisement è possibile prendere l'indirizzo, ma il DNS Server dal DHCP. Questa è una funzionalità ibrida (un po' senza, un po' con DHCP).

Il campo **options** in questo caso è importante perché una delle opzioni contiene i *prefissi*. Anche in questo caso si utilizza il formato **TLV** (come per l'extension header) ed ha una lunghezza che è multiplo di 8 bytes.

Type	Length	...
...		

Type (3)	Length	Prefix Length	L	A	Reserved
Valid Lifetime					
Preferred Lifetime					
Reserved					
Prefix					

L'opzione più importante è la **Prefix Information Option**, cioè l'opzione che permette di comunicare i prefissi. Sono presenti tutta una serie di campi all'interno di questa opzione: nuovamente informazioni utili per gestire la riconfigurazione della rete (*Valid Lifetime* e *Preferred Lifetime*). Queste opzioni forniscono la validità del prefisso, oltre il quale non può essere più utilizzato. Il *preferred* è il "consiglio" oltre il quale è consigliato non scegliere più quel prefisso. Ad esempio: la validità di un prefisso è 7 giorni ma è sconsigliato sceglierlo se sono passati più di 3 giorni. Comunque tramite il router advertisement periodico queste informazioni vengono refreshate periodicamente (perciò l'host mantiene aggiornata la sua configurazione). Questa è un'altra novità utile e interessante rispetto a IPv4. Anche in questo caso sono presenti dei flag **L** e **A**. Il flag **L** informa che il prefisso contenuto all'interno è sul link (serve perché le Router Advertisement possono essere utilizzate non solo per configurare l'indirizzo dell'interfaccia ma anche quando L=0 per segnalare dei prefissi che non sono sul link e che servono per andare a configurare la tabella di routing). Il flag **A** viene utilizzato per specificare

all'host che tale indirizzo può essere utilizzato per una configurazione automatica. Tale flag non lo si setta a 1 quando si vuole comunicare un certo prefisso a un host, oppure se si vuole forzare più host ad avere uno stesso prefisso.

Type (5)	Length	Reserved
MTU		

certa MTU si limita la frammentazione.

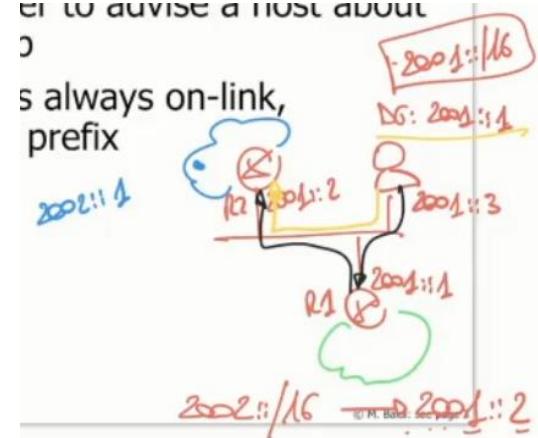
Un'altra opzione è la **MTU Option** per permettere al router di forzare agli host una certa MTU. Questo può succedere perché magari tramite una

Il **Link Layer Address Option** è un'altra opzione in cui il router scrive il suo *MAC Address*. Si utilizza una opzione di ICMP (e non il campo MAC Sorgente presente nell'header) per mantenere la struttura del layer. Si può utilizzare anche la Neighbor Discovery ma se all'interno di una Router Advertisement è presente anche questa opzione, si risparmia la Neighbor Solicitation per il router ed è dunque una ottimizzazione.

Type	Length	Link-Layer Address
Link-Layer Address ...		

ICMP Redirect

Il Redirect serve per configurare in maniera rudimentare la routing table di un host in uno scenario particolare, cioè quando è presente una destinazione migliore sullo stesso link per raggiungere una certa destinazione. Il caso tipico è quello presente nella foto a destra, in cui è presente un host con due router. Il DG dell'host è lo stesso del router in basso (R1). Se però l'host vuole comunicare con un host presente nella rete del router in alto (R2), il traffico verrà inviato dall'host verso il proprio default gateway, cioè a R1. Tutto funziona se nella routing table di R1 è presente l'informazione relativa a come raggiungere l'host desiderato ($2002::/16 \rightarrow 2001::2$) cioè si dice al router R1 che la rete $2002::/16$ si raggiunge attraverso R2.



Su R1 succede che

- R1 invia il traffico (il pacchetto ricevuto dall'host) sulla stessa interfaccia da cui lo ha ricevuto;
- Lo invia utilizzando come next hop un indirizzo che sta nella stessa subnet/network da cui lo ha ricevuto, perché si è ricevuto dalla subnet $2001::/16$ e lo si vuole inviare a $2001::2$, che sta nella stessa subnet. Dunque l'host può raggiungere direttamente R2 ma l'host non lo conosce.

Per tale motivo, l'ICMP Redirect comunica all'host tale presenza. La ICMP Redirect parterà da R1 verso l'host che va a riconfigurare la routing table dell'host inserendo la stessa entry presente in R1 ($2002::/16 \rightarrow 2001::2$). Ovviamente il primo pacchetto verrà processato da R1 e inviato su R2, ma dal secondo pacchetto in poi l'host non comunicherà più con R1 ma invierà direttamente a R2.

Type	Code	Checksum
Reserved		
Target Address		
Destination Address		
Options		

Il formato della ICMP Redirect è quello della foto a sinistra: *Target Address* e *Destination Address* sono rispettivamente la prima e la seconda colonna della routing table, ovvero, il *Target Address* è l'indirizzo da raggiungere, mentre il *Destination Address* è il next-hop.

Anche la ICMP Redirect contiene delle opzioni, tra cui la possibilità di comunicare all'host qual è l'IP che ha fatto generare la Redirect.

Duplicate Address Detection

Un'altra novità rispetto a IPv4 è la DAD, che viene lanciata ogni volta che viene effettuata una configurazione. Appena il processo visto prima termina, parte la DAD che si assicura che non ci siano utenti che stanno utilizzando lo stesso indirizzo IPv6 nella rete locale. Esiste un modo abbastanza semplice per scoprire se è presente un altro host nella rete che sta usando il mio indirizzo IP, ovvero inviare una Neighbor Solicitation con l'indirizzo IP che si è scelto. Si effettua quindi una Neighbor Discovery per il proprio indirizzo IP e ci si aspetta che nessuno risponda.

Stateless Configuration (host)

- All'accensione del dispositivo con IPv6 viene generato un indirizzo Link Local.
- Viene lanciata una DAD per l'indirizzo Link Local.
- Ci si iscrive al corrispettivo Solicited Node Multicast Address IPv6 (configurando la ricezione del corrispettivo MAC multicast e inviando una ICMP Multicast Listener Report)
- Le comunicazioni On-Link vengono abilitate

Stateless Configuration (Router)

- Possibilmente viene inviata una Router Solicitation
- Si ascolta alle varie Router Advertisement
- Si crea un indirizzo dal prefisso annunciato
- Si prova la sua unicità con una DAD
- Ci si iscrive al corrispondente IPv6 Solicited Node Multicast Address (configurando la ricezione del corrispettivo MAC multicast e inviando una ICMP Multicast Listener Report)

Dopo aver fatto la configurazione dell'interfaccia. Si continua a ricevere le Router Advertisement e vengono processate e si possono osservare i possibili cambiamenti: Il preferred lifetime può essere cambiato, dunque è in corso una reconfigurazione della rete.

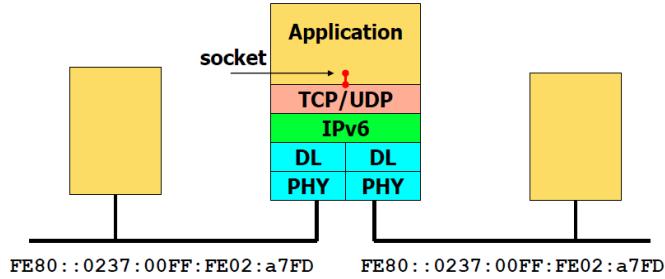
I router possono essere in comunicazione con l'esterno (con l'ISP), il quale può comunicare al router qual è il nuovo prefisso da usare su quella rete. Diventa dunque anche più semplice una riconfigurazione globale (ad esempio di una azienda). Si riesce, inoltre, a cambiare facilmente prefisso quando si cambia ISP.

È infatti più semplice gestire il cambio di indirizzi da parte di un provider, che può farlo in maniera abbastanza semplice poiché sono presenti dei protocolli per la gestione automatica del **Router Renumbering**, cioè i router si scambiano informazioni sui prefissi da utilizzare, inclusi quelli che dovranno essere usati sulla nuova rete.

Stateful Configuration: Dynamic Host Configuration Protocol

Si ha la staeful configuration nel momento in cui si va ad utilizzare il DHCP. In questo caso le cose sono molto simili a IPv4. La Discover cambia nome in Solicit mentre la Offer è diventata Advertise, ma i concetti restano identici. L'unica differenza è che non essendoci più il broadcast, viene utilizzato un indirizzo Multicast (nella Solicit e nella Request). L'indirizzo **all-agents** è un indirizzo particolare $FF02::1:2$ che indica tutti coloro che sono stati configurati come parte di questo particolare servizio (serve per limitare il traffico multicast, inviandolo solo ai server DHCP e non a tutti).

Scoped Addresses



Si tratta del simbolo % seguito da un numero in coda a un indirizzo IPv6. Questo simbolo non è presente né nel caso del Global Unicast né nel caso di Privato/Side Local. È un problema limitato ai Link Local (ed anche per questo non viene mai utilizzato).

Nelle configurazioni IP ogni rete fisica deve avere un prefisso diverso. Una stazione potrebbe avere più interfacce connesse a più reti fisiche. In IPv4 non era un problema perché non esistevano i link local, e continua a non esserlo nel momento in cui nelle varie interfacce vengono configurate dei Global Unicast o dei Side Local, ma nelle interfacce si configurano in automatico i Link Local che possono essere utilizzati per le comunicazioni On-Link. Per poter funzionare è però necessario aggiungere tale simbolo seguito dal numero. Questo perché nel momento in cui si hanno due interfacce, ci si ritrova ad avere lo stesso prefisso (FE80..) su entrambe le interfacce. Ci sono però dei casi in cui l'indirizzo è identico su entrambe le interfacce (come nella foto in alto), ma lo stesso problema si verifica anche se gli indirizzi sono diversi (perché il prefisso è lo stesso).

Per gestire il prefisso nella routing table si utilizzeranno delle route dirette per ognuna delle reti fisiche. Siccome il prefisso è $FE80::/16$, a quale next-hop viene inviato dall'host presente nel centro della foto? È ambigua, perché FE80 è presente in ambo i lati. Per tale motivo non è possibile scrivere una route diretta.

Per risolvere il problema, sono stati memorizzati gli indirizzi IPv6 su 17 byte anziché 16 (**solo all'interno del sistema operativo**, negli altri livelli è sempre 16 byte) e nell'ultimo byte aggiunto rappresenta l'interfaccia da cui è stato ricevuto. Si mescolano dunque le informazioni di livello 2 con quelle di livello 3. Questo byte viene detto **scope**. Viene rappresentato come l'esempio seguente:

- $FE80::0237:00FF:FE02:a7FD%19$

Il numero finale è un **identificatore di intefaccia**. In questo modo i prefissi in un certo modo cambiano. Non avremo più FE80 ma da un lato $FE80%10$ e dall'altro $FE80%9$.

Routing in IPv6

Per parlare di routing bisogna fare due distinzioni:

- **On-the-fly routing:** può essere utilizzato in questo contesto come sinonimo di *forwarding*, anche se non lo è. Si tratta di andare a leggere la routing table ed utilizzarla.
- **Proactive routing:** può essere utilizzato in questo contesto come sinonimo di *routing*, anche se non lo è. Si tratta di costruire la routing table e decidere quali sono i passi da seguire.

Il proactive routing può essere **manuale** (ovvero ciò che è stato discusso fino ad ora) oppure può essere svolto attraverso l'utilizzo di **protocolli di routing**, che distribuiscono informazioni sulle destinazioni (e.g. "sono capaci di raggiungere queste reti con costo X") e in maniera automatica, utilizzando gli algoritmi per la costruzione di alberi di cammini minimi, costruiscono la tabella di routing. Nulla cambia tra IPv4 e IPv6.

Ciò che va considerato in questa trattazione è che in realtà difficilmente ci si può dimenticare di IPv4, ovvero per molti anni IPv4 e IPv6 coesisteranno, perciò va garantito un supporto per entrambi i protocolli sui dispositivi di routing (router).

Di default i router Cisco hanno IPv4 attivo, mentre se si vuole abilitare IPv6 bisogna lanciare uno specifico comando:

```
Router#configure terminal
```

```
Router(config)#ipv6 unicast-routing
```

Una volta lanciato questo comando i router potranno usare anche il routing IPv6. Saranno dunque in grado di avere indirizzi IPv6 configurati sulle interfacce, ma anche di forwardare pacchetti IPv6. In futuro questa cosa potrebbe cambiare in modo da avere IPv6 abilitato di default.

Nel momento in cui IPv6 viene abilitato sui router, questi inizieranno a utilizzare sia IPv4 che IPv6 (a meno che non si decida di disabilitare IPv4), perciò vi saranno **due routing table separate**, che funzionano secondo gli stessi principi (sempre col *longest prefix matching*).

Protocolli di routing

Per far coesistere IPv4 e IPv6 in uno stesso contesto di routing dinamico esistono due approcci:

- **Integrated Routing:** un singolo protocollo effettua la *Advertisement* (l'annuncio) delle destinazioni per entrambi i protocolli di livello 3.
- **Ships in the night:** IPv4 e IPv6 gestiscono le cose separatamente. Esisterà un protocollo che gestisce IPv4 e un altro per IPv6. Ognuno non si preoccupa e non vede cosa effettua l'altro.

Pro e contro Integrated Routing

La scelta tra i due approcci dipende da numerosi fattori, poiché entrambi possiedono dei pro e dei contro. Ad esempio, nell'*Integrated Routing* il grosso vantaggio è che non si ha la duplicazione dei messaggi e dei meccanismi (unico protocollo con le stesse logiche per entrambi i protocolli). Ovviamente l'albero dei cammini minimi sarà duplicato, poiché le tabelle sono comunque due separate, ma il protocollo che gestisce lo scambio dei messaggi (da *non confondere* con l'algoritmo di routing) sarà soltanto uno. I lati negativi di questo approccio è che, volendo utilizzare un unico protocollo bisogna modificarlo per adattarlo al problema e ciò potrebbe comportare la nascita di nuovi bug e compromettere il funzionamento di entrambi i protocolli. Un

altro problema è che non tutti i router sono sia IPv4 che IPv6, perciò se un unico protocollo si ritrova a dover trasportare le informazioni di entrambi i protocolli, solo chi possiede entrambi i protocolli potrà comunicare correttamente nella rete, dunque non si può sfruttare a pieno la topologia.

Pro e contro Ships in the night

In questo caso sono presenti due diversi protocolli di routing, e per tale motivo è possibile fare il tuning dei due protocolli a seconda della specifica topologia o dello scenario da considerare. Questo risolve il problema dell'Integrated Routing relativo alla topologia. Questo protocollo consente anche una **migrazione più tranquilla**, nel senso che è possibile far continuare a lavorare IPv4 come ha sempre fatto senza apportare alcuna modifica e incrementalmente è possibile utilizzare anche IPv6 (se un baco viene introdotto in IPv6, il protocollo IPv4 continua a funzionare senza problemi). Consente anche un **riconoscimento dei problemi più semplice**, poiché si identifica subito se un nuovo bug appartiene a IPv4 o IPv6, mentre nel caso precedente non era così semplice accorgersene (unico protocollo di routing per entrambi i protocolli IPv4 e IPv6). Il vero conto di questo approccio è la necessità di **duplicare tutti i messaggi e le strutture dati** poiché sono due protocolli completamente distinti.

Protocol	Approach
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

- Il protocollo *statico* (manuale) per definizione è Ships in the night: viene configurato a mano prima IPv4 e poi IPv6, dunque per definizione effettuando le cose a mano si ha un approccio Ships in the night.

- *OSPFv3* è uno dei protocolli più utilizzati su internet e utilizza un approccio Ships in the night, anche se è stato definito uno standard per utilizzare anche l'Integrated Routing.
- *BGP* è l'altro importante protocollo utilizzato oggi, standardizzato per poter utilizzare entrambi gli approcci ma l'Integrated Routing è quello più utilizzato per una questione di praticità.

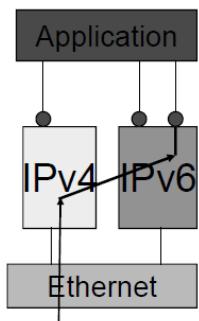
Esempio di Routing Table

```
C2800#sh ipv6 route
IPv6 Routing Table - 15 entries
Codes: C - Connected, L - Local, S - Static, R - RIP
      O - OSPF intra, OI - OSPF inter
O  2013::/112 [110/65]
  via FE80::20F:34FF:FE7:ABDE, FastEthernet1
O  2016::/112 [110/65]
  via FE80::223:EBFF:FE44:C6EE, FastEthernet0
  via FE80::20F:34FF:FE7:ABDE, FastEthernet1
C  2017::/64 [0/0] ← On-link prefix
  via ::, FastEthernet0/1
L  2017::2/128 [0/0] ← Interface address
  via ::, FastEthernet0/1
L  FE80::/10 [0/0] ← Link-local prefix
  via ::, Null0
L  FF00::/8 [0/0] ← Multicast prefix
  via ::, Null0
S  ::/0 [1/0] ← Default route (static)
  via FE80::20D:BCFF:FEB9:29A3, FastEthernet2
```

- Nella prima colonna è presente il **tipo**: in questo sistema operativo **S** sta per Static che indica le route statiche, **L** sta per Local presente solo su IPv6 [NON APPROFONDIRE], **C** si riferisce alle route dirette, **O** sta per OSPF e sono route dinamiche calcolate da un protocollo OSPF.
- Successivamente è presente la destinazione ed il next-hop che nel dispositivo in questione viene indicato con **via**.
- La colonna successiva è l'interfaccia di uscita che indica su quale interfaccia verrà configurata la route.
- I numeri tra [] si utilizzano nel routing dinamico. Il primo numero indica il costo calcolato dal protocollo di routing, mentre il secondo serve per comparare più protocolli di routing. Ad esempio, c'è RIP che si utilizza negli stessi contesti di OSPF, e possiede una metrica basata sul numero di hop, mentre OSPF si basa sul costo in euro di tale link. Un amministratore di rete può scegliere di avere entrambi i protocolli in esecuzione, ma deve avere inoltre la possibilità di scegliere uno dei due protocolli (indicato come **priorità**). Dunque, il secondo numero è la **priorità** che un amministratore di rete da al particolare protocollo di routing. Dunque, se si volesse mandare in esecuzione anche RIP ma con meno priorità, si potrebbe dare un numero maggiore di 65 (esempio in foto) in modo tale che quel router vada sempre a scegliere OSPF per raggiungere la destinazione anche se RIP propone un altro percorso con un corso totale assoluto più basso (più basso semplicemente magari perché utilizza una metrica diversa).
- Nell'esempio in figura è presente anche una /112 anche se si era detto di non avere reti più lunghe di /64. Questo è presente poiché è stato proposto uno standard che propone di non usare più le reti /64 **solo nei link punto-punto** per questioni di sicurezza. Negli anni /112 è stato deprecato e si consiglia il /127.
- La default route non cambia, ed è identica ::/0 ed è una route statica (scritta a mano). Il default gateway sarebbe l'indirizzo dopo “via” nella default route (*FE80::20D..*)
- La route diretta di tipo C è una /64, perciò indica che sulla propria rete si ha quella specifica rete IP global unicast.
- Riguardo al Link-local, si nota che per evitare di far uscire il traffico link local dalla propria rete, si è aggiunta la entry più specifica che inoltra sull'interfaccia Null0 il traffico (per bloccare tutto il traffico, tranne le entry indicate precedentemente). Null0 è un'interfaccia particolare che rappresenta il localhost.
- La stessa cosa accade con il multicast: sul router non è abilitato il traffico multicast verso l'esterno, e viene inoltrato tutto su Null0.

Transizione da IPv4 a IPv6

Per gestire la transizione verso IPv6 si utilizza un approccio **incrementale**: è necessario trovare un modo per far coesistere entrambi i protocolli. Inoltre, deve essere **seamless**, cioè per l'utente finale non deve cambiare nulla. Infine, deve essere **smooth** (cioè tranquillo) ovvero non devono esserci interruzioni del servizio.

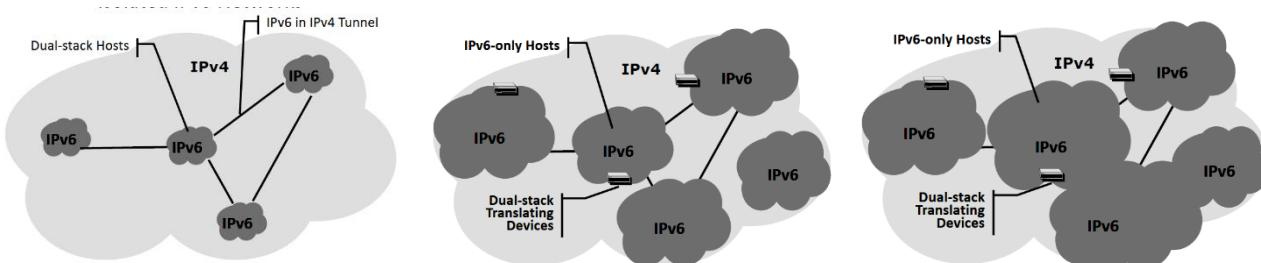


Le stazioni (ovvero gli host) devono poter garantire le caratteristiche sopra citate. La soluzione è quella di utilizzare un approccio **dual-stack**, ovvero esistono due stack separati che gestiscono i protocolli (due istanze di IP). Il vecchio stack continua a funzionare come sempre, mentre quello nuovo avrà le dovute modifiche. Il dual-stack è fondamentale perché una stazione può scegliere quale dei due protocolli utilizzare.

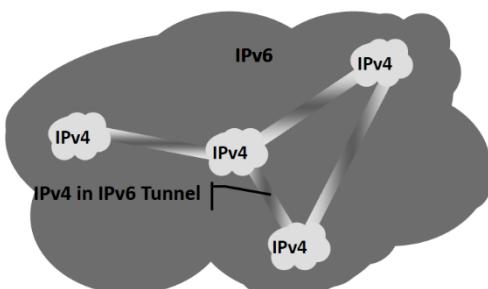
Una volta deciso l'approccio dual-stack è necessario effettuare l'**Address Mapping**, ovvero bisogna gestire il mapping degli indirizzi da IPv4 a IPv6 (e viceversa), ad esempio se si ha un indirizzo IPv6 ma si deve parlare con un server che ha un indirizzo IPv4.

Sono necessari anche meccanismi di **Tunneling**, ovvero un modo per far comunicare due stazioni IPv6 che però sono collegate da stazioni IPv4. Si crea dunque una pipe tra le due reti IPv6 che sia in grado di attraversare la rete IPv4.

Se ho necessità di raggiungere una stazione IPv4 da una stazione IPv6, dopo aver fatto il mapping resta il problema di dover uscire dalla rete IPv6 per entrare nella rete IPv4 (non mi serve il tunneling perché il destinatario non è una stazione IPv6). Devo dunque **tradurre** (usare una sorta di NAT, ma molto più evoluto) e rimuovere l'header IPv6 e sostituirlo con un header IPv4 sensato.

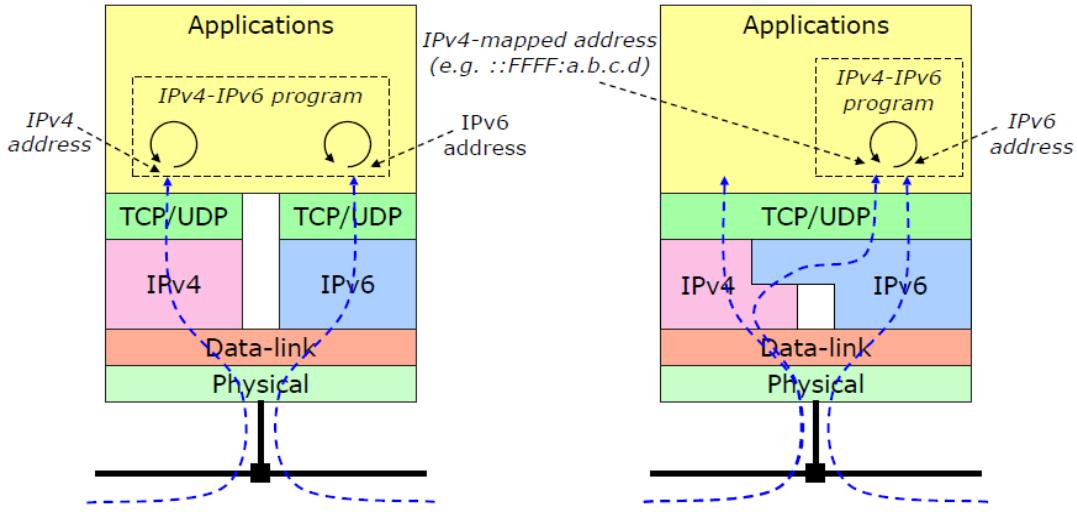


Inizialmente le reti IPv6 sono nate con delle piccole reti (figura 1) che per comunicare avevano bisogno di Tunnel creati tra le “isole” IPv6 per far parlare i nodi, che erano in modalità dual-stack. Successivamente, la situazione si è evoluta come in figura 2, che rappresenta la soluzione attuale. Una vasta porzione della rete è funzionante con IPv6, ma molti dei nodi sono **IPv6-only** (un esempio è Facebook che all'interno della rete aziendale utilizza oggi solo IPv6). Per questo motivo entrano in gioco i **dispositivi di traduzione**, che effettuano la traduzione in indirizzi IPv4. Vale comunque il discorso del tunneling per raggiungere altre reti IPv6. Le “isole” si stanno man mano allargando e unendo (figura 3) poiché sempre più provider si stanno convertendo a IPv6.



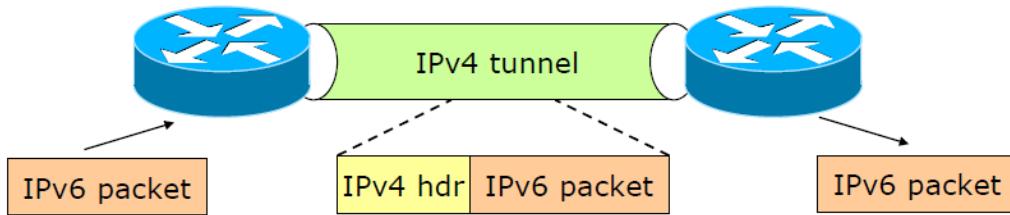
L'obiettivo finale è quello nella figura qui a sinistra, in cui la nuova internet sarà IPv6, mentre le piccole isole saranno quelle in IPv4. A quel punto sarà necessario fare dei tunnel al contrario (IPv4 che deve fare per parlare attraverso una rete IPv6).

L'approccio Dual Stack non è l'unica soluzione che è stata pensata. Il dual-stack ha delle limitazioni, la più grossa è legata al fatto che se la rete sottostante supporta entrambi i protocolli e dunque fornisce 2 indirizzi (1 IPv4 e 1 IPv6) sarà l'applicazione a decidere quale dei due utilizzare. Questo non è un problema se la rete fornisce un solo indirizzo. Il duplice indirizzo può essere un problema perché se si lascia decidere all'applicazione, il gestore della rete non avrà più il controllo su quale protocollo viene utilizzato. Per tale motivo è stato sviluppato l'approccio **Dual Layer**.

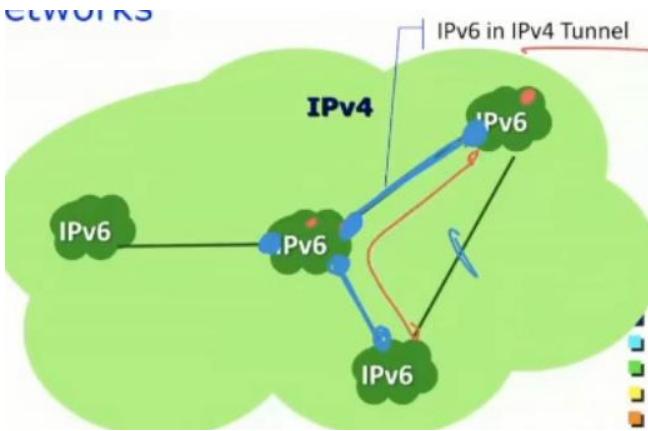


Nel dual layer a destra, vi è un'unica istanza TCP/UDP che però è stato modificato per supportare i due protocolli. L'applicazione parla col piano sottostante in un unico modo definito dal sistema operativo. Se l'applicazione vuole utilizzare IPv6, ma noi vogliamo utilizzare IPv4 è possibile effettuare la traduzione (IPv6 nella figura si appoggia su IPv4). In questo modo posso mantenere il controllo sul tipo di protocollo utilizzato nella rete. Si nota, inoltre, che in questo contesto vengono utilizzati gli **IPv4-mapped**. Se arriva un pacchetto IPv4 ma nell'applicazione si vuole utilizzare IPv6, l'indirizzo viene tradotto nell'IPv4-mapped che viene compreso dall'applicazione. Nonostante ciò, viene sempre utilizzato l'approccio Dual Stack.

Come attraversare una rete solo IPv4



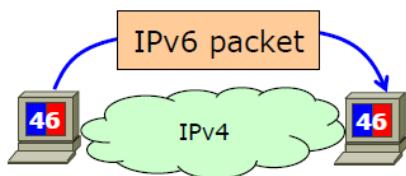
Due stazioni IPv6 per poter comunicare quando “in mezzo” è presente una rete IPv4 devono effettuare **tunneling**. Per fare ciò, banalmente si incapsula un pacchetto IPv6 in un pacchetto IPv4. Ci si troverà ad avere due header IP. I router IPv4 nella rete lavoreranno sul primo header IP che trovano, perciò il tutto funziona poiché il routing verrà effettuato su IPv4. Ciò accade finché non si raggiunge al **tunnel end-point**, che sa di essere tale e rimuove l'header IPv4 per far andare avanti nella rete l'header IPv6. Per *incapsulare* è possibile inserire IPv4 dentro IPv6 direttamente inserendo il **Protocol Type=41**. Questo serve al tunneling point per capire che va gestito in un certo modo (e non ai nodi intermedi). Un'altra soluzione prevede l'utilizzo di un ulteriore header ed è il protocollo **GRE**. Questo protocollo serve per fare tunneling. In questi contesti i due modi sono più o meno equivalenti.



Finché si tratta di fare tunnel non è difficile, ma se ad esempio la rete centrale in foto deve inviare un pacchetto IPv6 alla rete che sta in alto, si ritrova ad avere 3 Tunnel End-Point, perciò il primo problema che si pone è la scelta dell'end-point e la soluzione più immediata sarebbe quella di effettuare un routing che nella rete IPv6 permette di scegliere il tunnel corretto. Ma il problema successivo risulta quello di come creare i tunnel end-point (come identificare il tunnel di una rete IPv6, quale strada far fare ai tunnel, prendere una strada piuttosto che un'altra). Tutte

queste domande non hanno una semplice risposta, poiché la soluzione *manuale* è funzionante ma *non scalabile*. Esistono dunque tutta una serie di tecniche che si utilizzano per creare i tunnel e capire a chi inviare il traffico una volta che ci si trova sul tunnel end-point (*IPv4-compatible addresses*, *6over4*, *6to4*, *Tunnel Broker*, *ISATAP* e *Teredo*).

Host-centered Solutions



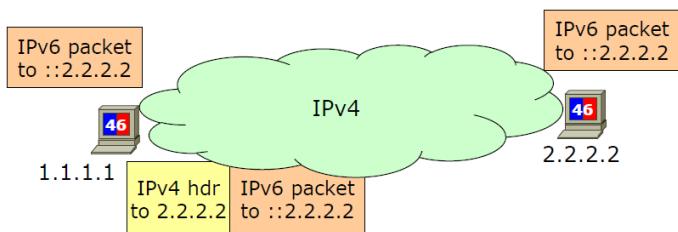
Sono state le prime soluzioni al problema che avevano uno scopo sperimentale. Queste soluzioni cercano di affrontare il problema della comunicazione in IPv6 tra due stazioni Dual Stack su una rete IPv4. Anche se la questione non ha molto senso visto che essendo entrambi dual stack ed essendoci una rete IPv4 sarebbe più conveniente farli

comunicare direttamente in IPv4, ma come già detto questo era un metodo per **sperimentare** nuovi metodi di comunicazione.

IPv4-compatible Addresses

Questo metodo è una soluzione che di default è sempre abilitata sui sistemi operativi perché molto semplice. Per inviare traffico IPv6 con un indirizzo IPv4 lo si può fare utilizzando gli indirizzi **IPv4-compatible** (::/96). Alcune volte (in maniera impropria) viene anche definito “*automatic tunneling*” ma si confonde con il metodo ISATAP (che si vedrà successivamente).

L’idea è quella di avere un indirizzo IPv4 (e.g. 1.1.1.1) e di voler inviare traffico a un altro indirizzo IPv4 (e.g. 2.2.2.2) su una rete IPv4 ma, per sperimentazione, si vuole utilizzare IPv6. Per farlo, si può “puntare” a quel determinato indirizzo IPv6 utilizzando un IPv4-compatible derivato dall’indirizzo destinazione (e.g. 2.2.2.2).



Il “trucco” sta nel fatto che nella routing table di un host che supporta questo meccanismo è presente una **route statica speciale** che invia gli indirizzi IPv4-compatible verso una *interfaccia virtuale* su cui è installato un software di tunneling, che effettua un tunnel automatico verso l’indirizzo IPv4 corretto.

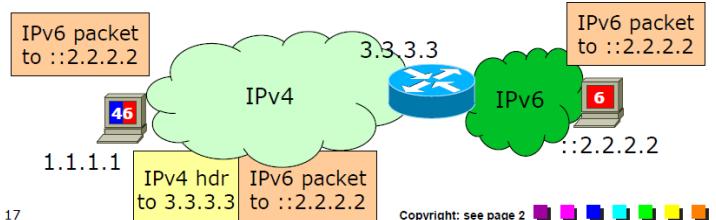
In altre parole, si fa tutto su rete IPv4 utilizzando però un indirizzo IPv6. Si creerà un tunnel tra 1.1.1.1 e 2.2.2.2 e si anteporrà l’header IPv4 seguito dall’header IPv6.

Questa situazione si può estendere al caso in cui ci sia una rete IPv6 come nell’esempio seguente.

IPv4-compatible Address con Router Dual Stack

L'esempio precedente non ha applicazioni utili e può essere visto soltanto come metodo sperimentale. È possibile però applicare lo stesso metodo in una situazione diversa, in cui può risultare utile utilizzare gli indirizzi IPv4-compatibile.

Nell'esempio in foto, oltre alla rete IPv4 è presente anche una rete IPv6 separata da quella IPv4 tramite un router Dual Stack. In questo caso la route statica non ha un tunneling che termina sull'indirizzo destinazione (e.g. 2.2.2.2), ma termina all'indirizzo del router che sta tra le due reti (e.g. 3.3.3.3). Il tutto viene "modificato" a mano.



17

over4

Questa soluzione è molto interessante poiché utilizza un "trucco" che semplifica di molto le cose, ma possiede un problema molto grande che non permette di utilizzare questo metodo a livello provider. La soluzione si basa sul multicast IPv4, ma richiedere a una rete IPv4, che sta tra le reti IPv6, di supportare il multicast è un grosso limite.

L'idea è nata come "Host-centered", ovvero come iniziativa di sperimentazione. In questo caso però è stata subito estesa per adattarla a contesti più interessanti. Questa soluzione rende la definizione dei tunnel **automatica**, facendo finta che la rete IPv4 sia un link e utilizzando il Neighbor Discovery.

Supponiamo di avere una rete IPv4 contornata da tante reti IPv6, ognuna connessa alla rete IPv4 da dei router Dual Stack. Per capire a quale indirizzo IPv4 inviare il traffico per raggiungere il destinatario (da una rete IPv6 a un'altra rete IPv6), si segue il procedimento seguente:

- Si riesce a esportare un annuncio di routing dalla rete IPv6 alla rete IPv4. Per farlo si definiscono dei particolari indirizzi IPv6 da utilizzare sui router derivanti dagli IPv4 e poi si utilizza la Neighbor Discovery per sapere qual è l'indirizzo IPv4.

Se so che devo tradurre un indirizzo come *2001::1* che sta nella rete IPv6, è possibile fare una Neighbor Discovery sulla rete IPv4 per tale indirizzo. Il router deve fare il joining al gruppo multicast derivante dall'indirizzo Solicited Node Multicast Address derivante dalla rete IPv6 da raggiungere.

Esempio



Si suppone di avere un link (dunque rete Ethernet) e degli host Dual-Stack che possiedono degli indirizzi IP link local (dunque *FE80::...*). Il "seguito" di questi indirizzi IP link local (che prima è stato specificato con dei puntini) può essere l'indirizzo IPv4 che in realtà è presente (perché in realtà non si tratta di un link ma di una rete IPv4).

Dunque, se è presente un host H1 con indirizzo IPv4 *1.1.1.1*, si può utilizzare il "trucco" di creare un indirizzo IPv6 che rappresenti tale host, creando ad esempio il link local *FE80::101:101* che è valido e che possiede come InterfaceID l'indirizzo IPv4 che poi si va a utilizzare sulle interfacce verso la rete IPv4 in maniera fittizia (facendo finta di avere IPv6, dunque l'indirizzo link local è virtuale).

Se H2, che non conosce l'indirizzo IPv4 di H1 ma soltanto quello IPv6, vuole inviare un pacchetto ad H1 sulla rete IPv4, è necessario conoscere il relativo indirizzo IPv4. Dunque, H2 prova ad effettuare un procedimento simile a una Neighbor Discovery. Si calcolerà il Solicited Node Multicast Address partendo dall'indirizzo IPv6 link local ($FE80::101:101$) di H1 per ottenere $FF01::1:FF01:101$. Essendo però la rete IPv4, la Neighbor Solicitation non può viaggiare su IPv4. Per ovviare al problema, si va ad utilizzare un gruppo multicast IPv4 "speciale": 239.192.x.y. Al posto di x e y si inseriranno gli ultimi due byte dell'indirizzo IPv6 di H1.

Per tale motivo, l'indirizzo multicast sarà 239.192.1.1. H1 si sarà preventivamente messo in ascolto su questo gruppo multicast e perciò potrà ricevere le richieste di H2. Nella Neighbor Advertisement che H1 invierà ad H2 si riceverà dunque l'indirizzo IPv4. Si potrà quindi utilizzare per raggiungere tale host sulla rete IPv4.

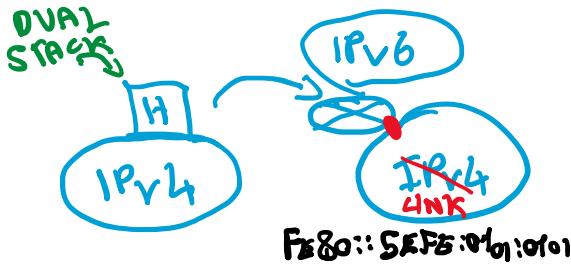
Funziona esattamente come la Neighbor Solicitation ma invece di imparare indirizzi MAC si imparano indirizzi IPv4. Qui si ha una rete IPv6 e si vuole fare una rete IPv4, dunque bisogna avere un match tra i due indirizzi.

Con un'estensione è possibile gestire anche il fatto che ci siano dei router che supportano nuvole IPv6 con in mezzo la rete IPv4, dove è semplicemente più complicato il mapping (poiché non ci si può più basare solo con l'indirizzo link local ma bisogna ragionare utilizzando le reti IPv6 che sono presenti all'interno).

Ad ogni modo, non si è obbligati ad utilizzare nell'interface ID l'indirizzo IPv4, ma è possibile utilizzare qualsiasi indirizzo si voglia.

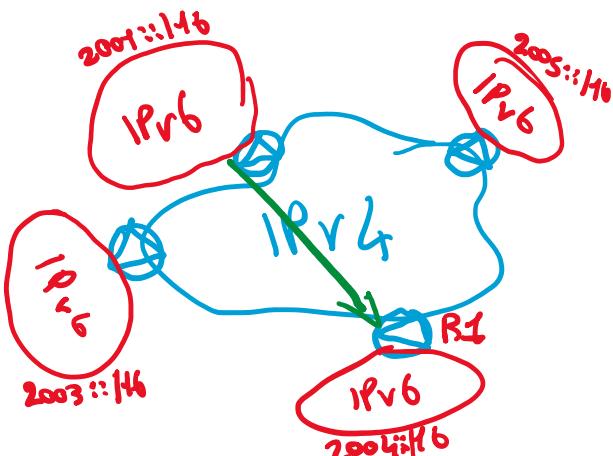
ISATAP: Intra-site Automatic Tunnel Addressing Protocol

Soluzione più utilizzata, nata sempre come host-centered. Per sviluppare l'idea si è partiti dalla considerazione che IPv4 non supporta il multicast. Si è quindi cercato un metodo simile a 6over4 ma senza multicast. Siccome su 6over4 l'utilizzo dell'indirizzo IPv4 come Interface ID non era obbligatorio (poiché poi c'era la Neighbor Discovery che permetteva di scoprire il vero indirizzo) in questo caso è obbligatorio.



(Nel disegno sono disegnate entrambe le alternative host/router, basta guardare la parte a destra della freccia)
Si parte da una situazione in cui si ha un host o un router connesso a una rete IPv4 (ma Dual-Stack) si è obbligati a definire sull'interfaccia verso IPv4 (ma non da usare verso l'interfaccia, rimane sempre un'interfaccia virtuale) un indirizzo IPv6 che è FE80 (nuovamente un Link Local) e successivamente si aggiunge arbitrariamente 5ef5 e poi l'indirizzo IPv4. Ad esempio, se l'indirizzo IP è 1.1.1.1 si ottiene: $FE8::5ef5:0101:0101$.

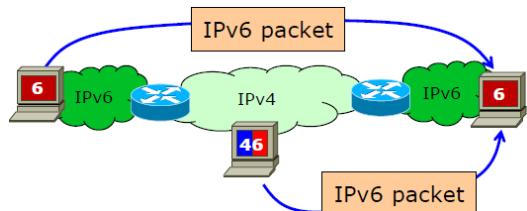
L'idea è stata quella di dire che quando bisogna raggiungere una certa stazione IPv6 si fa una Query DNS per quel particolare nome (e.g. *isatap.polito.it*). Il DNS (da configurare, ma fattibile) restituisce una **PRL** (**Potential Router List**) ovvero la lista di tutti i router che fanno parte di ISATAP e che hanno realizzato il mapping sopra specificato. Una volta ottenuta la lista è possibile effettuare delle *Router Solicitation* (in unicast) verso ognuno dei router della lista utilizzando l'indirizzo IPv6 e incapsulando i pacchetti dentro degli indirizzi IPv4 che hanno come indirizzo IP destinazione ciò che c'è scritto nell'ultima parte dell'indirizzo IPv6 (ecco perché è obbligatorio: non si ha più la Neighbor Discovery e l'indirizzo IPv6 è solo un trucco per utilizzare la Router Advertisement disponibile solo in IPv6). Ogni router risponderà con una Router Advertisement.



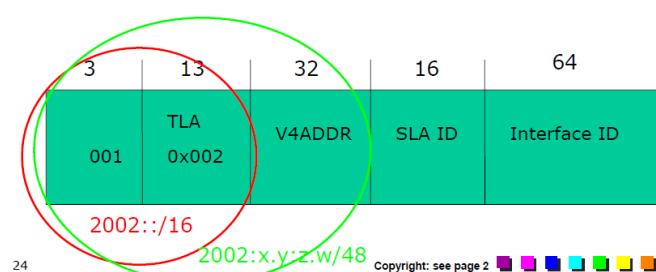
maniera veloce che può raggiungere tali host attraverso quel router.

Network-centered Solutions

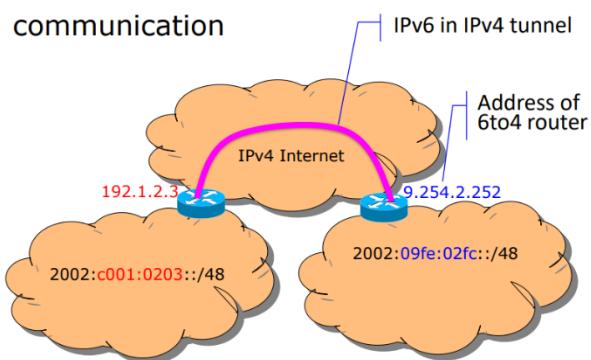
Queste soluzioni sono nate per gestire la situazione in figura, ovvero quella in cui host *Dual Stack* e nativi IPv6 si scambiano pacchetti attraverso una rete IPv4.



6to4



un'intera rete e nel network ID, che rappresenta l'intera nuvola, che rappresenta l'intera rete, rimarrebbe l'indirizzo IPv4 ed è possibile ragionare per prefissi e non più per singoli host.



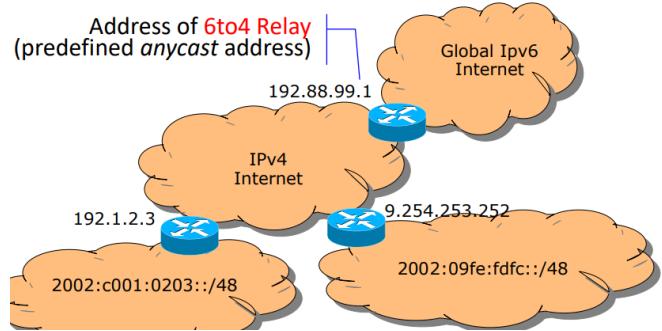
per soluzioni 6to4. Quando si vede 2002 dunque si riconosce che si sta utilizzando 6to4. A seguire si trova $x.y:z.w$ cioè l'indirizzo IPv4 dell'interfaccia del router nei successivi 32 bit. Tutte le stazioni IPv6 connesse nella nuvola IPv6 avranno un indirizzo in tale rete. Gli ultimi 16 bit sono liberi e servono per poter definire più reti IPv6 connesse allo stesso router, estendendo dunque l'immagine in alto.

Supponiamo di avere tutte reti IPv6 raggiungibili tramite i 4 router. Nella *Router Advertisement* che, per esempio, un router invia a R1 (colui che magari ha effettuato la *Solicitation*), restituisce due cose: (se serve) anche un **prefisso** che non sia *FE80* che è possibile utilizzare sul router, ad esempio per fare una comunicazione off-link (perché *FE80* ha validità solo locale, anche se è una cosa emulata); si annunciano i prefissi, ovvero le **reti raggiungibili**, ed è proprio ciò che serve. La *Router Advertisement* che arriva a R1 conterrà quel prefisso *2001::/16* come prefisso non utilizzabile ma raggiungibile. Dunque, R1 capisce in maniera veloce che può raggiungere tali host attraverso quel router.

Nelle soluzioni precedenti abbiamo sempre usato sull'interfaccia virtuale un indirizzo IPv6 dove l'indirizzo IPv4 è nell'interface ID. Se bisogna rappresentare un'intera rete IPv6 con un certo indirizzo IPv4 è possibile importare l'indirizzo dentro il prefisso dell'indirizzo IPv6 che si assegna al router. Così facendo è possibile rappresentare un'intera rete e nel network ID, che rappresenta l'intera nuvola, rimarrebbe l'indirizzo IPv4 ed è possibile ragionare per prefissi e non più per singoli host.

In questa soluzione si cerca di risolvere il problema di rappresentare intere nuvole mettendo l'indirizzo IPv4 del router che mi permette di raggiungere quelle nuvole dentro al prefisso della rete IPv6 che viene utilizzata nella nuvola. La struttura dell'indirizzo IPv6 per utilizzare 6to4 è la seguente: nella prima parte è presente 2002, trattandosi dunque di indirizzi globali unicast. I 2002 formalmente non possono essere utilizzati per soluzioni di addressing generiche ma solo

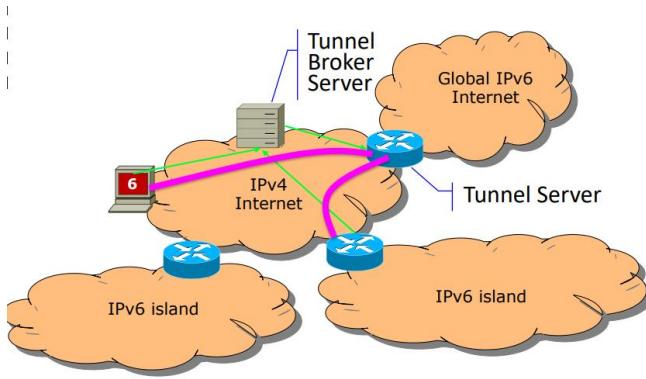
In 6to4 è possibile anche avere un default gateway verso l'eventuale Internet IPv6. Per reti specifiche rimane necessario utilizzare indirizzi che iniziano con 2002. Ma se una rete IPv6 vuole inviare un pacchetto, ad esempio `2001::1` lo può mandare a un **6to4 Relay** (in foto indicato con l'indirizzo `192.88.99.1`, uno dei tanti indirizzi possibili) e verrà dunque inoltrato sulla rete internet.



Teredo

Molto simile a 6to4 con l'unica differenza che se nella rete IPv4 è necessario usare dei NAT, una soluzione IP in IP può avere dei problemi. Teredo semplicemente per risolvere il problema utilizza **UDP** per incapsulare i tunnel in modo da essere compatibile con soluzioni NAT (poiché le soluzioni NAT utilizzano anche le porte).

Tunnel Broker



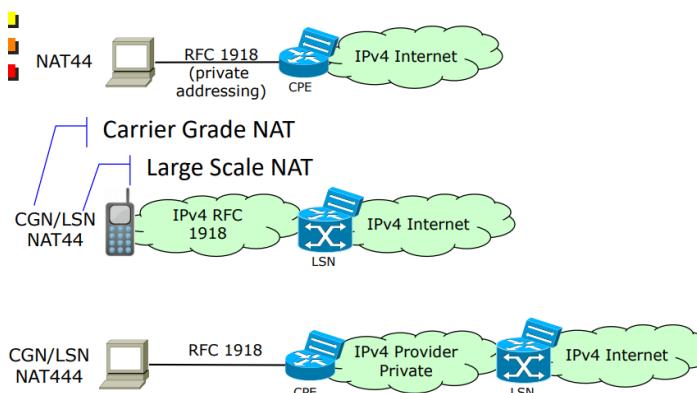
Questa soluzione risolve il problema di avere reti che iniziano con 2002, aggiungendo la possibilità di utilizzare indirizzi generici. Per fare ciò si crea un **server centralizzato** (chiamato *Tunnel Broker*) e quando si necessita di un tunnel si contatta il tunnel broker che fornisce le informazioni necessarie. Per fare ciò si utilizzano i protocolli **Tunnel Setup Protocol (TSP)** o **Tunnel Information Control (TIC)**. I router di frontiera vengono chiamati **Tunnel Server**.

Il problema di questa soluzione è la **non scalabilità**. Se le reti crescono in numero, il Server può andare in crisi. Inoltre, un server centralizzato è comunque un Single Point of Failure.

Scalable, Carrier-grade Solutions

Soluzioni utilizzate dai provider. I provider, dovendo gestire numerose reti periferiche, devono risolvere i problemi in maniera diversa. Gli obiettivi posti sono: possibilità di un host IPv6 possa continuare a raggiungere host IPv4; gestire client IPv4 privato/pubblico quando poi in mezzo è presente una rete IPv6.

Oggi per gli indirizzi privati i NAT sono molto utilizzati in vari contesti.



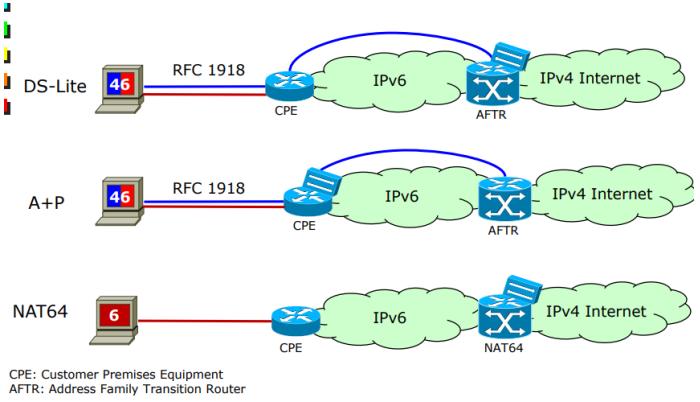
Scale NAT (LSN).

Una terza casistica è quella di avere utenti privati collegati alla internet pubblica attraverso una rete privata.

Un primo contesto comune è quello di utilizzare una rete privata nelle home network. In questo caso su un home gateway (chiamato **CPE**) vi è installato un NAT per poter uscire dalla rete privata.

Esistono anche interi provider che funzionano con indirizzamento privato, tra cui le reti mobili. In questo caso le soluzioni sono identiche (si necessita di un NAT) che però gestisce un numero elevato di client e viene chiamato **Large**

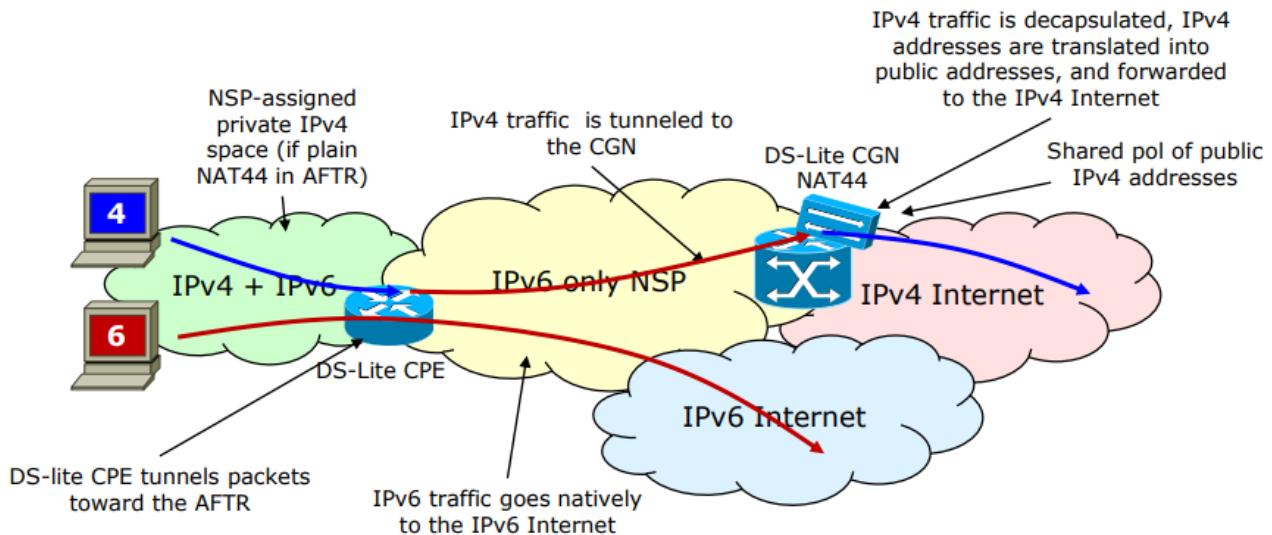
Con IPv6 si vogliono trasformare le soluzioni NAT rimuovendo le reti IPv4 private dalle reti dei provider cercando di tornare alla condizione in cui si ha indirizzi privati e pubblici in IPv6. Il problema rimane Internet che continua ad essere una rete IPv4 e perciò è necessario trasformare l'indirizzo privato IPv4 (di casa) in un indirizzo IPv6 e poi di nuovo in un indirizzo IPv4.



- La soluzione adottata da **DS-Lite** funziona come segue: da IPv4 privato si arriva sull'home gateway (CPE) che effettua tunnel verso l'AFTR tramite una rete IPv6 su cui è installato un Large Scale NAT (LSN). Un unico NAT gestisce tutti i clienti.
- Per superare alcune di queste limitazioni (tra cui la difficoltà nel far scalare il NAT) si è passati a una soluzione in cui il NAT è installato sul CPE. 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.
- Se si possiede un indirizzo IPv6 e bisogna andare sulla rete internet IPv4 bisogna installare il **NAT64** che trasforma il pacchetto IPv6, rimuovere l'header e sostituirlo con un header IPv4.

La soluzione **MAP** si pone nel problema risolto da **A+P**, ma è una via di mezzo tra la soluzione DS-Lite e A+P.

DS-Lite



Se l'host è su IPv6 e vuole parlare con una rete IPv6 non vi è alcun problema. Se invece l'host è IPv4 e deve parlare con una rete IPv4 ma in mezzo il provider utilizza una rete IPv6, bisogna adottare delle tecniche.

La soluzione è semplice: un indirizzo IPv4 viene “tunnellato” dal CPE verso l'AFTR che è conosciuto dal CPE e proprio sull'AFTR viene installato il NAT. Il problema di questa soluzione è che non è possibile avere su reti diverse lo stesso indirizzo privato. Se si lascia utilizzare l'indirizzamento privato ai clienti, questi utilizzano

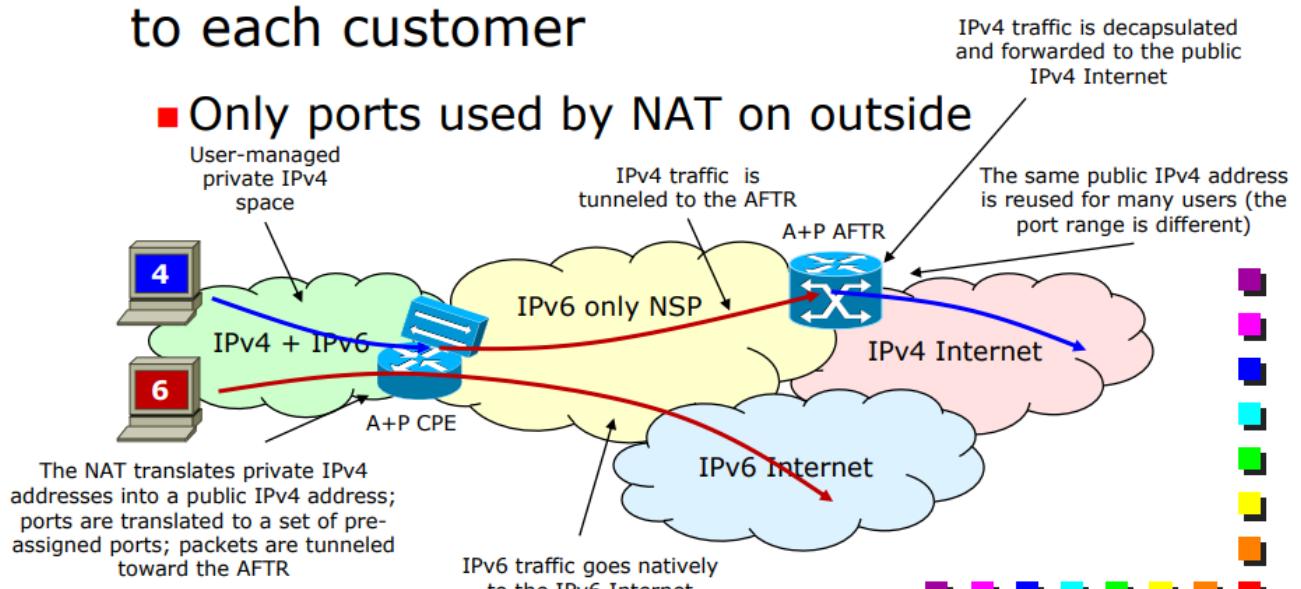
sempre la stessa rete locale ($192.168.0.0/24$). Il NAT posizionato dov'è si ritrova a gestire traffico che arriva da host diversi con lo stesso indirizzo IP, cosa che all'interno della rete di casa non si verifica.

Per risolvere questo problema bisognerebbe inserire nel NAT anche l'indirizzo IPv6 che si utilizza per fare il tunnel. In questo modo non si utilizza soltanto l'indirizzo sorgente del pacchetto, ma anche l'indirizzo IPv6 che sta fuori dall'header IPv4. La soluzione funziona ma non è eccezionale, poiché il LSN necessita già di opportune caratteristiche per poter funzionare e bisognerebbe chiedere a questo software già complesso di fare ulteriori controlli. Un ulteriore problema è che il NAT non è sotto controllo dell'utente. Un altro problema è legato al port-forwarding: non è possibile installare web server nella rete di casa. Tutti questi problemi vengono risolti con l'A+P.

A+P

to each customer

■ Only ports used by NAT on outside



L'idea è quella di spostare il NAT sul CPE. Così facendo, tutti i problemi citati precedentemente non sono più presenti: si ha controllo su NAT, non più problemi di scalabilità. Così facendo però si introduce un altro problema: per installare il NAT è necessario avere un indirizzo IPv4 da assegnare al CPE. Questo è un problema perché se il provider ha usato IPv6 a causa della mancanza di indirizzi IPv4 pubblici e perciò è necessario utilizzare anche le **porte**.

Posta una scarsità di indirizzi IPv4 ci si ritroverà in una situazione in cui diversi CPE avranno lo stesso indirizzo IP pubblico, poiché i pochi indirizzi IPv4 residui vengono condivisi tra tutti i CPE. Questi ultimi creeranno i tunnel verso l'AFTR, per poi uscire sulla IPv4 pubblica. Quando però il pacchetto torna indietro dalla rete pubblica verso l'AFTR si verificherà un problema di **ambiguità** poiché l'indirizzo IP apparterrà a più NAT. È proprio qui che intervengono le **porte**, assegnando ai vari CPE un **range di porte** diverso da utilizzare sui NAT.

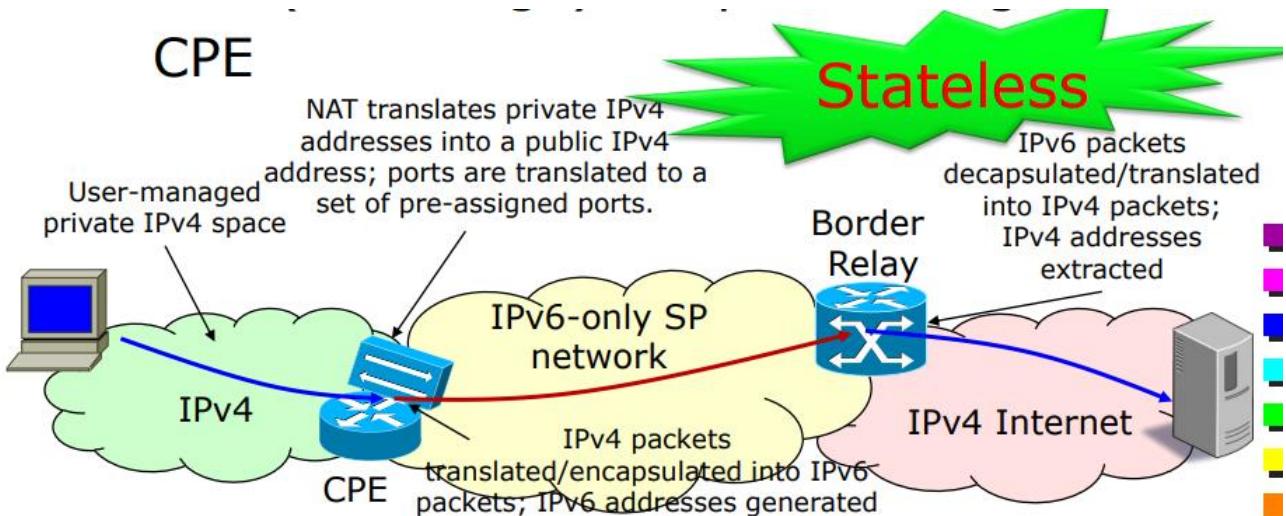
Quando una rete IPv4 vuole comunicare, invia il pacchetto al proprio CPE ed il NAT, oltre a cambiare l'indirizzo sorgente, cambierà anche la porta sorgente e ne sceglierà una in un range ben specifico che è stato assegnato a quel CPE. L'AFTR, per decidere a chi mandare il pacchetto, guarderà la **porta contenuta all'interno del pacchetto** che arriva dalla rete internet Ipv4 e dunque sceglierà a quale CPE consegnare il pacchetto.

Per sapere quante porte sono necessarie per gestire una certa rete esiste un **protocollo** che permette all'ISP di dialogare con i CPE (le soluzioni di questo tipo sono spesso proprietarie).

Di AFTR ne esiste più di uno fisicamente (magari anche distribuiti geograficamente) con diverse connessioni alla rete IPv4 pubblica. Si utilizza però l'**anycast**, cioè un modo per utilizzare lo stesso indirizzo IP su diverse entità e dunque permette alla rete di vedere un unico AFTR (composto da più AFTR dislocati).

MAP

Con A+P ci si è soffermati sul problema dell'indirizzo IPv4 che è lo stesso sui vari CPE e dunque è necessario il range di porte per individuare il CPE corretto. In tutto ciò si è trascurato che il NAT ha un indirizzo IPv4 utilizzato da più CPE, ma è un indirizzo da usare sul NAT e **non** è un indirizzo da usare sull'interfaccia del router, perché la rete sarà IPv6 e quindi l'interfaccia avrà un indirizzo IPv6. Si fa NAT e dunque si crea il tunnel verso l'AFTR. Quando il pacchetto torna indietro l'AFTR riuscirà a capire qual è il CPE corretto grazie al numero di porta però all'interno della rete IPv6 questa informazione non è utile, ma bisogna dunque generare l'**indirizzo IPv6** del CPE a cui consegnare il pacchetto. Questo concetto nella soluzione A+P (ma anche nel DS-Lite) è mantenuto dall'AFTR, che rimane comunque abbastanza complesso poiché deve mappare che il socket indirizzo:porta si raggiunge tramite un certo indirizzo IPv6.

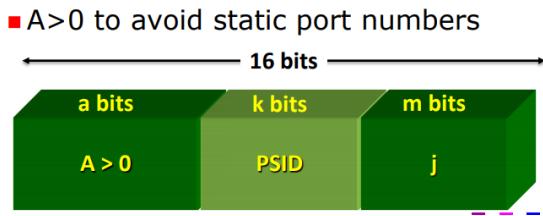


Con il MAP si vuole **ridurre ulteriormente il carico sull'AFTR**, che nel contesto del MAP si chiama **Border Relay**. MAP **non** utilizza un range di porte ma un **set di porte**. Questo è dovuto al fatto che spesso i meccanismi di load balancing si basano sulle porte ed usando un range tutte le porte di una certa rete finirebbero verso un certo server. Invece di usare un range si utilizza un Set in modo da permettere a una rete di comunicare con più server.

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 sull'indirizzo IPv6 del CPE partendo dalle informazioni che ha (e che gli vengono fornite dal pacchetto IPv4 che arriva dalla rete internet).

Questa soluzione nella rete IPv6 può adottare due strategie: **tunneling e translation**. Rispettivamente si ha il **MAP-E (Map with Encapsulation)** e il **MAP-T (MAP with Translation)**.

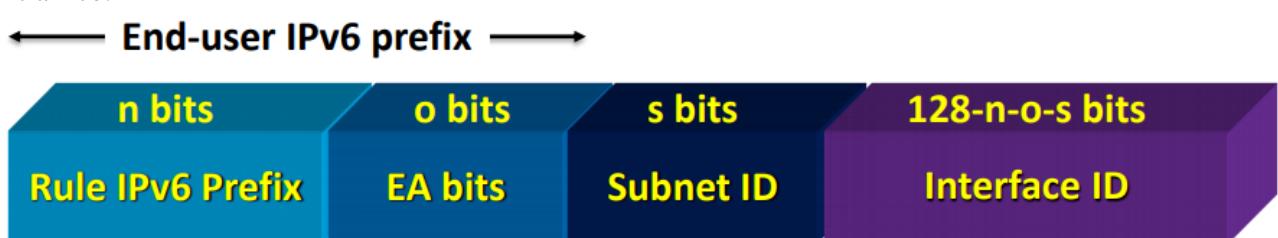
Port Set



Per creare un set di porte si analizzano i 16 bit delle porte di livello 4. Per effettuare un range si tengono costanti i primi bit (es. si assegnano a un CPE tutte le porte che iniziano con 1100). Per creare invece dei set ci si sposta a metà dei 16 bit e si mantengono costanti i bit centrali (es. assegnano a un CPE tutte le porte che nel centro hanno i bit 1100). I bit centrali possono essere di lunghezza variabile e possono iniziare in un punto variabile dei 16 bit.

CPE IPv6 Address

Tutto si basa sul fatto che l'indirizzo IPv6 del CPE contiene delle informazioni sul CPE e legate in qualche modo all'host IPv4 che si vuole raggiungere sulla rete IPv4 in modo tale che il Border Relay possa automaticamente leggendosi le informazioni che ha sul pacchetto IPv4 ricostruirsi l'indirizzo IPv6 a cui inviare il traffico.



- **Rule IPv6 Prefix:** prefisso utilizzato nella particolare istanza di MAP e rimane costante indipendentemente dall'CPE.
- **EA bits:** contengono parte dell'indirizzo IPv4 pubblico assegnato al NAT e una porzione del Port set.
- **Subnet ID:** compatibilità con gli indirizzi IPv6 (sostanzialmente sempre messo a 0).
- **Interface ID:** informazioni dell'indirizzo IPv4

Mapping Rule

Il tutto si basa sulla definizione di una mapping rule assegnata dal Provider. È conosciuta sul CPE e sul Border Relay. Nella mapping rule è presente:

- **Rule IPv6 prefix**
- **Rule IPv4 prefix:** prefisso IPv4 dell'indirizzo IPv4 che viene utilizzato su quel particolare CPE
- **EA bits length**

In più è presente anche il **PSID Offset**: valore di a (indicato nella figura del Port Set con $A>0$) comune a tutto il MAP domain e non è necessario tenerne traccia.

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

Esempio costruzione indirizzo IPv6

- Rule IPv6 prefix: 2001:1:1100::/40
 - Rule IPv4 prefix: 195.2.2.0/24
 - EA bit length: 16
 - PSID offset: 4
 - PSID: 0x33
 - Address used by CPE: 195.2.2.4
-
- | | | | |
|----------------------|------------------|--------------|--------------------------------|
| 40 bits
2001:1:11 | 16 bits
04:33 | 8 bits
00 | 64 bits
0000:C302:0204:0033 |
|----------------------|------------------|--------------|--------------------------------|

Le prime 3 sono le informazioni che si trovano anche sul Border Relay.

Al CPE bisogna dire anche qual è il PSID offset (costante per tutta la rete e per l'istanza di MAP)

Bisogna comunicare anche il PSID per definire il set di porte

Al CPE bisogna dare l'indirizzo IPv4 da utilizzare sul NAT e deve essere all'interno dell'indirizzo presente in Rule Ipv4 prefix.

La costruzione dell'indirizzo Ipv6 che viene utilizzato sull'interfaccia IPv6 del CPE avviene come segue:

- Essendo il prefisso IPv6 su 40 bit, i primi 40 bits vengono utilizzati per inserire il Rule IPv6 prefix.
- L'informazione su quanto deve essere lungo EA bit è presente (16) e dunque in questa porzione viene inserita la parte di **host** dell'indirizzo IPv4 (04) e poi il **PSID** (33). Se EA bit length è fissata a 16 bit bisogna fare in modo che poi inserendo la parte di host e il PSID il tutto "torni", cioè che il totale sia 16 bit. Facendo *EA bit length-PSID offset* posso sapere quanto è lungo il PSID.
- Nella parte di subnet ci sono tutti 0
- Nella parte di **interface ID** vanno inseriti degli 0 iniziali e poi si scrive l'**indirizzo IPv4** dato al CPE seguito da altri 0 e poi di nuovo il PSID.

Così facendo si riesce a risalire all'indirizzo mantenendo solo poche informazioni provenienti dal mapping rule comunicato al Border Relay e al CPE.

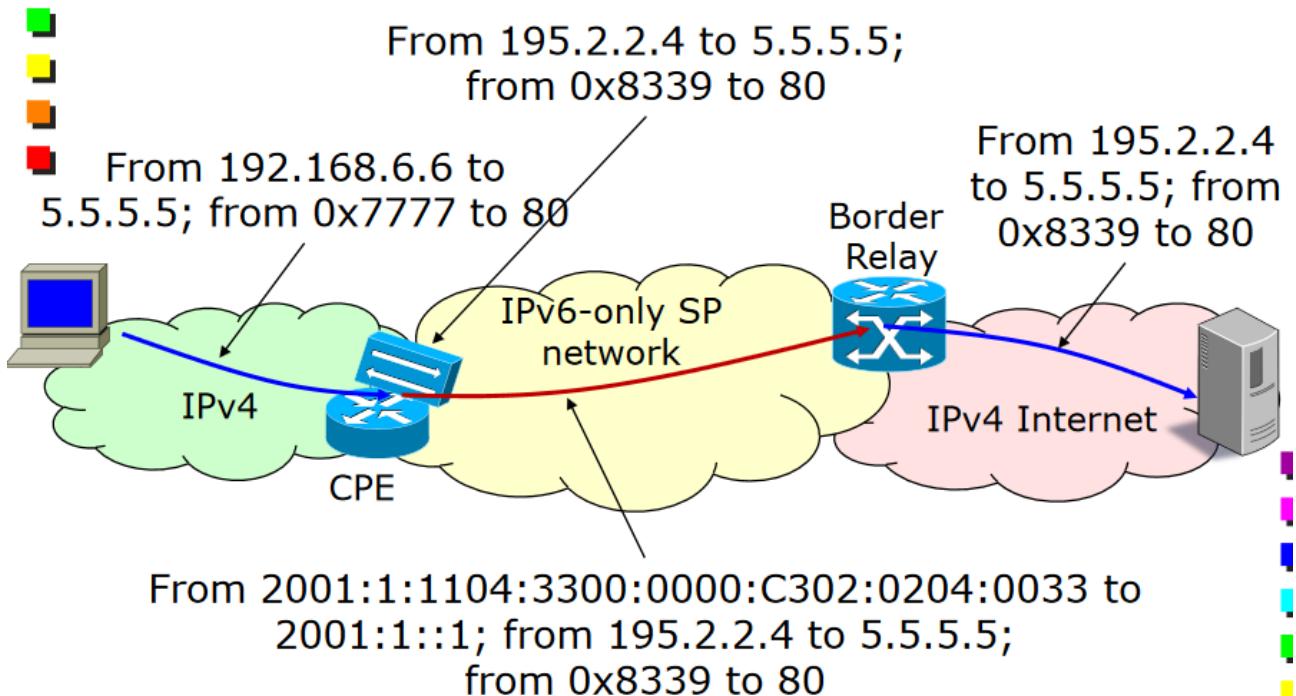
L'univocità dell'indirizzo è data dalla combinazione dell'*indirizzo IPv4+Mapping Rule*.

Border Relay

Il tunnel creato al CPE per inviare il pacchetto alla rete internet IPv4 viene inviato verso il Border Relay, un **indirizzo conosciuto** dal CPE. Spesso viene utilizzato anche in questo caso l'**anycast**.

Successivamente bisogna capire quale delle due metodologie si vuole utilizzare per inviare il pacchetto: se si utilizzare il Tunnel con il **MAP-E** bisogna inviare il tunnel all'indirizzo del Border Relay; viceversa, se si vuole utilizzare il **MAP-T** bisogna trovare un modo per cambiare l'header da IPv4 a IPv6 nel modo opportuno.

MAP-E



Osservando lo schema in figura si nota quanto segue:

- Un nodo privato *192.168.6.6* vuole inviare del traffico a *5.5.5.5* che si trova all'altro capo. Il nodo utilizza come porta sorgente la porta *0x777* che va verso la porta 80 (webserver).
- Il CPE effettua **NAT** trovando come sorgente l'indirizzo IP pubblico *195.2.2.4*, l'indirizzo destinazione e la porta rimangono le stesse ma viene **cambiata anche** la porta che diventa *0x8339* (si nota 33 che corrisponde al PSID).
- Visto che si sta utilizzando il MAP-E si effettua **tunneling**, perciò bisogna creare un header IPv6 da appendere all'header IPv4. L'indirizzo IP sorgente si costruisce esattamente come mostrato prima e lo simanda all'indirizzo IP del **border relay** (in questo caso *2001:1::1*). Il Border Relay rimuove l'header IPv6 e può inoltrare il pacchetto IPv4 sulla rete IPv4 Internet.

Quando invece il webserver risponde al nodo, si procede come già spiegato, cioè il Border Relay leggerà le informazioni nel pacchetto e nelle Mapping Rule per ricreare il tunnel al contrario.

MAP-T

- IPv4-embedded IPv6 address
- RFC 6052



■ BR prefix: *2001:1::/64*

■ Outside destination: *5.5.5.5*



■ *2001:1::5.505:500:0*

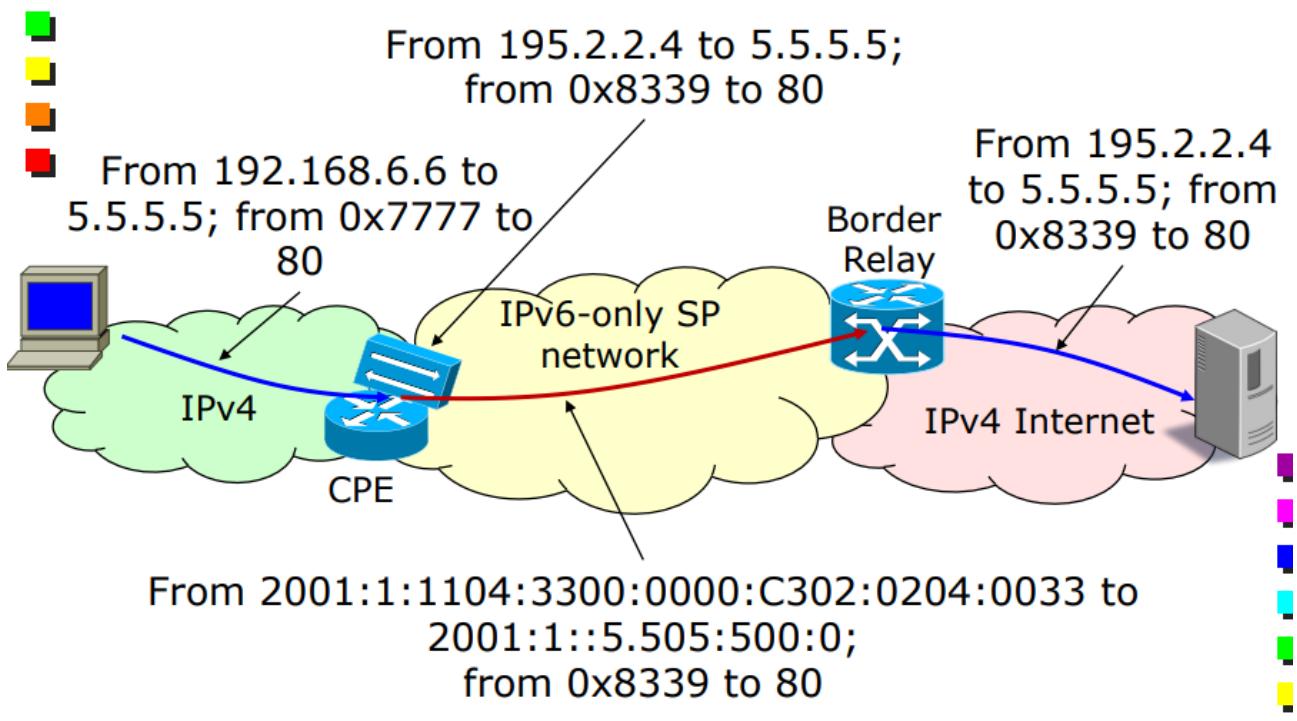
Le cose si complicano nel caso in cui si voglia utilizzare il MAP Translation, poiché l'informazione "da *195.2.2.4* a *5.5.5.5*; da *0x8339* a *80*" non è più presente. È necessario quindi ricostruire l'indirizzo IPv4 corretto. Per risolvere il problema, si utilizza anche per il Border Relay degli indirizzi particolari effettuando **embedding**. Si fa ciò per inserire dentro l'indirizzo IPv6 destinazione (cioè quello del Border Relay) anche l'indirizzo IPv4, altrimenti andrebbe persa. La porta destinazione

non si perde perché è dentro il livello 4. Inoltre, non si perde neanche l'indirizzo IPv4 sorgente perché la costruzione dell'indirizzo del CPE rimane identica a quanto detto prima.

Quando bisogna inviare il pacchetto verso la rete IPv4 pubblica non si ha più come prima un unico indirizzo IP come prima (a.e. $2001::1$) ma si utilizza un indirizzo generato come segue:

- Si prende il prefisso scelto per il Border Relay (esempio come prima $2001:1::/64$)
- Nella parte di Host nell'Interface ID si inseriscono 8 bit a 0 e poi si inserisce l'indirizzo IPv4 nei 32 bit successivi. (inserisco 5.5.5.5 dentro il prefisso IPv6 per ottenere $2001:1::5.505:500:0$)
- A seguire, altri 24 bits a zero.

È l'ennesimo modo per mappare un indirizzo IPv4 in indirizzo IPv6. Se il traffico viene inviato all'indirizzo generato, si ottiene l'informazione sulla destinazione IPv6 e il Border Relay è in grado di ricostruire il pacchetto.



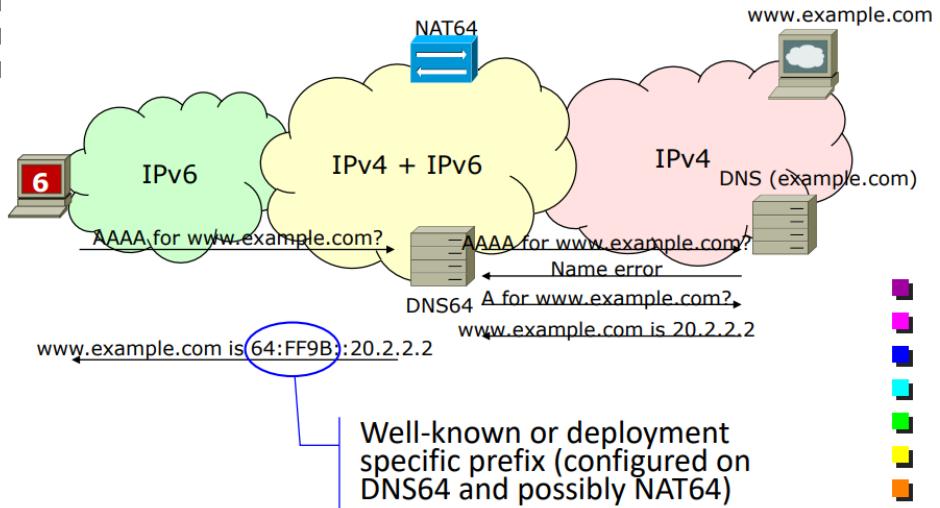
Rimane il problema dei numerosi ECP presenti sulla rete IPv6. Tutti i server pubblici generano un indirizzo per il Border Relay diverso. Il Border Relay per risolvere il problema effettua l'**advertisement** del suo prefisso sul suo backbone (cioè sulla rete IPv6). Il Border Relay che ha il prefisso dell'esempio ($2001:1::/64$) annuncia tale indirizzo sulla rete. Gli annunci causano che tutta la rete viene vista come raggiungibile attraverso il Border Relay, perciò scriveranno sulle proprie routing table tale entry e invieranno il traffico al Border Relay.

Di conseguenza, il Border Relay saprà di dover leggere all'interno dell'indirizzo IPv6 per leggere l'indirizzo IPv4 da consegnare.

NAT64 + DNS64

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 tutto parte con una classica risoluzione di nomi. Quando l'host IPv6 vuole raggiungere il nome `www.example.com` farà una query DNS di tipo IPv6 (AAAA) e la farà al DNS64 che inoltrerà verso il DNS autoritativo di quel dominio la richiesta. Il DNS autoritativo, essendo sulla rete IPv4, non può ricevere la richiesta. Per tale motivo il DNS64



ripeterà la richiesta per un indirizzo IPv4 e il DNS inoltrerà la risposta. Per far funzionare ciò è necessario che la rete intermedia sia *IPv4+IPv6*. Ad alcune stazioni su una porzione della rete bisogna garantire la presenza di due tipi di indirizzi perché se bisogna, appunto, inviare una richiesta al DNS IPv4 è necessario effettuarlo utilizzando un indirizzo IPv4.

Il DNS64, ad ogni modo, riceve l'informazione sull'indirizzo IPv4 che risolve il dominio. A questo punto però il DNS64 deve fornire all'utente un indirizzo IPv6 e non IPv4. Per fare ciò viene utilizzato nuovamente l'**embedding** (identico a quello utilizzato prima) con un prefisso di default **64:FF9B**. Il DNS64 manda indietro una risposta del tipo `64:FF9B:20.2.2.2` che l'utente potrà utilizzare per comunicare con l'host destinazione.

Il problema è che l'utente creerà pacchetti IPv6 da inviare alla destinazione IPv4, perciò a questo punto il **NAT64** si occuperà di tradurre tale pacchetto IPv6 in un pacchetto IPv4. Per inviare il pacchetto al NAT64 si utilizza nuovamente un annuncio di routing attraverso cui il NAT64 può avvisare tutti i router nella rete quale prefisso può utilizzare.

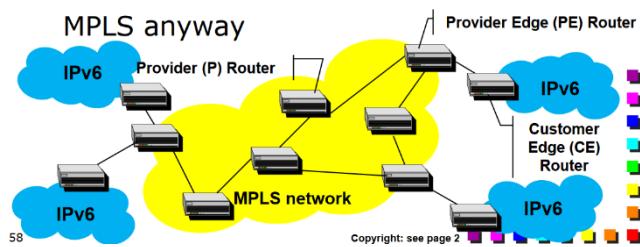
Limite di questa soluzione: funziona soltanto se si utilizzano dei nomi. Esempi:

- Ping `www.example.com` → funziona
- Ping `1.2.3.4` → non funziona

Si potrebbe far funzionare le cose forzando gli host ad effettuare l'embedding in maniera automatica. Inoltre, non è possibile abilitare il DNSSEC (cioè la firma della risposta a un record DNS) perché DNS64 modifica i record.

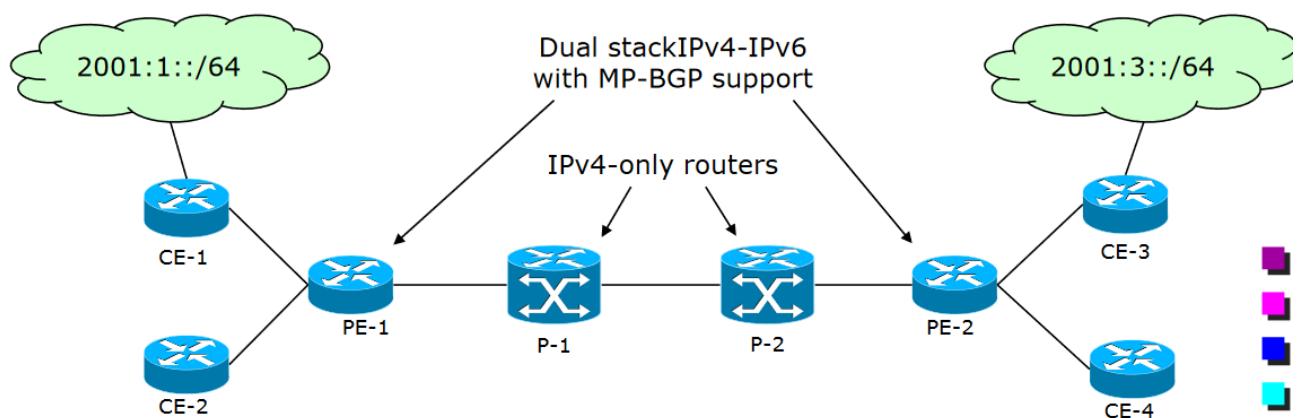
Soluzioni basate su MPLS

Attenzione: fare prima MPLS in fondo agli appunti e poi riprendere da qui dopo le VPN MPLS.

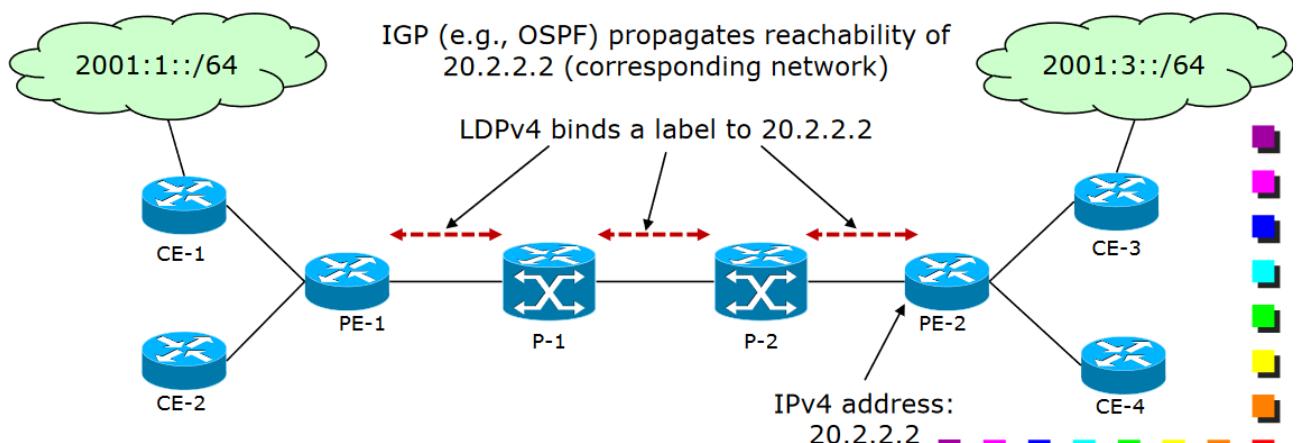


In questo caso vi sono degli utenti che stanno già utilizzando IPv6 e un backbone MPLS IPv4. Si vogliono interconnettere queste isole IPv6 attraverso il backbone IPv4. MPLS non guarda i pacchetti, però il backbone è IPv4 nel piano di controllo, poiché gli indirizzi sono IPv6 mentre i router conoscono solo IPv4. Il problema si risolve allo stesso modo delle VPN MPLS. Si rendono consci dell'esistenza di IPv6 soltanto i PE. Per tale motivo, la soluzione si chiama **6PE**. I router all'interno del backbone continuano ad utilizzare solo IPv4 nel piano di controllo.

Come nel caso basato nelle soluzioni del VPN, l'altro modo è l'equivalente delle VPN MPLS basate sul livello 2 andando a creare a mano le interfacce di accesso dei PE, andando a creare gli LPS per connettere le destinazioni. **IPv6 over Circuit Transport** è l'equivalente di PWE3.

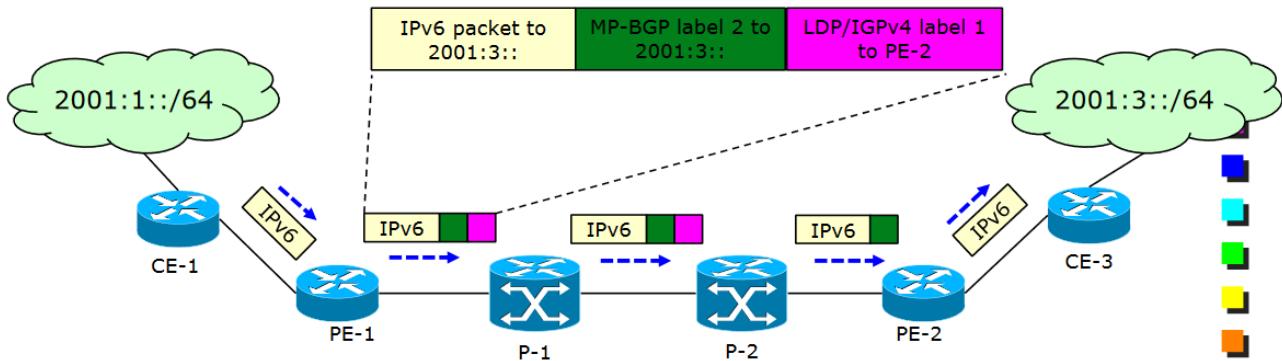


I router del backbone sono solo IPv4, mentre i PE sono dual stack. I PE possono dunque creare delle LSP tra di loro, in questo caso tra PE-1 e PE-2. Gli LSP sono unidirezionali, quindi bisogna crearne 2 per gestire il traffico ambo i lati.



A questo punto i PE usando il BGP (solo BGP) annunciano le destinazioni IPv6 direttamente agli altri PE. Quando poi bisognerà inserire un next hop per PE-2 ad esempio, si farà il mapping dell'indirizzo IPv4 di PE-2 su IPv6. A quel punto si associerà un'etichetta associata alla destinazione (2001:3::/64) e PE-2 la annuncerà anche a PE-1. GBP è un protocollo di routing che permette a router **non adiacenti** di comunicare.

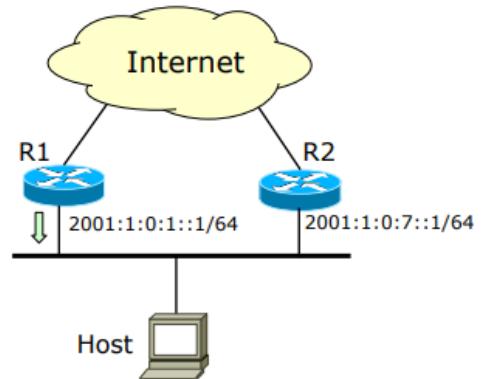
Quando i pacchetti partiranno, questi saranno inviati con due etichette, come in foto.



Dalla stazione $2001:1::/64$ il pacchetto partirà e PE-1 riceverà il pacchetto e andrà a controllare nella tabella quale etichetta assegnare e quale next-hop inviarlo (PE-2) come router IPv6. Quando il PE vede che come next-hop bisogna inviarlo a un indirizzo IPv6 vedrà che quello è un indirizzo IPv4 mappato su IPv6, perciò saprà che ci sarà un indirizzo IPv4 che avrà quella destinazione. Per inviare la trama MPLS a quel next-hop guarderà la tabella MPLS IPv4 e aggiungerà l'etichetta per inviarla a tale nodo. I nodi intermedi guarderanno l'etichetta più esterna e invieranno il pacchetto fino a PE-2. Questo toglierà l'etichetta, toglierà anche l'etichetta relativa all'indirizzo IPv6 e lo inoltrerà sulla rete.

Esercizio 1: Stateless configuration

Given the configuration in the figure, assuming that Router Advertisement is enabled only in R1, which IPv6 addresses are obtained by the host interface with stateless configuration?



Possible answers:

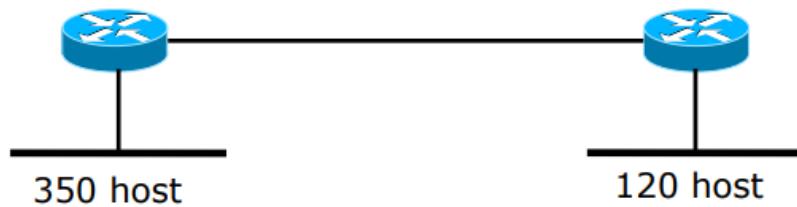
- A) 2001:1:0:1:20b:5dff:fe4c:3a6b, 2001:1:0:7:20b:5dff:fe4c:3a6b, fe80::20b:5dff:fe4c:3a6b
- B) 2001:1:0:1:20b:5dff:fe4c:3a6b, fe80::20b:5dff:fe4c:3a6b
- C) 2001:1:0:7:20b:5dff:fe4c:3a6b, fe80::20b:5dff:fe4c:3a6b
- D) 2001:1:0:1:20b:5dfe:ff4c:3a6b, fe80::20b:5dff:fe4c:3a6b
- E) 2001:1:0:1:20b:5dff:fe4c:3a6b, fe80::20b:5dff:fe4c:3a6b

- A e C sono false perché è presente il prefisso di R2 e ciò non è possibile visto che R2 non invia Advertisement.
- B possiede il prefisso corretto con un certo interface ID e successivamente il **link local** con un certo interface ID. Però il link local **non può** avere 1 (devono essere tutti 0) perciò non è corretto.
- D è falsa perché l'interface ID non è ottenuto correttamente dal MAC address (i due byte FE ed FF sono posizionati in maniera errata). In sostanza non sono presenti i due gruppi da FF che bisogna inserire sul MAC Address da 48 bit per avere i 64 bit (supponendo **stateless configuration**).
- E vera.

Exercise 5: addressing plan

Define an IPv6 addressing plan for the network shown in the figure.

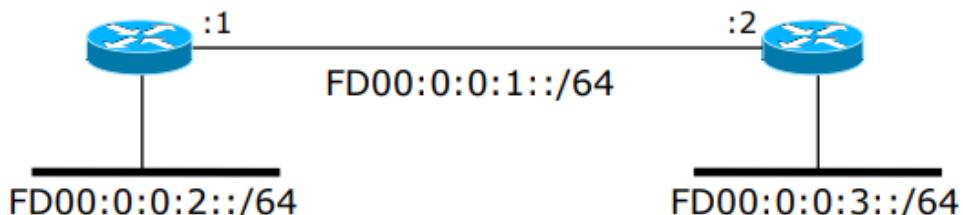
Define also the static routes that should be configured in the two routers, in order to have a correctly working system.



Visto che nella rete non è presente internet, basta utilizzare dei **link local**.

Static Route:
S FD00:0:0:3::/64 FD00:0:0:1::2

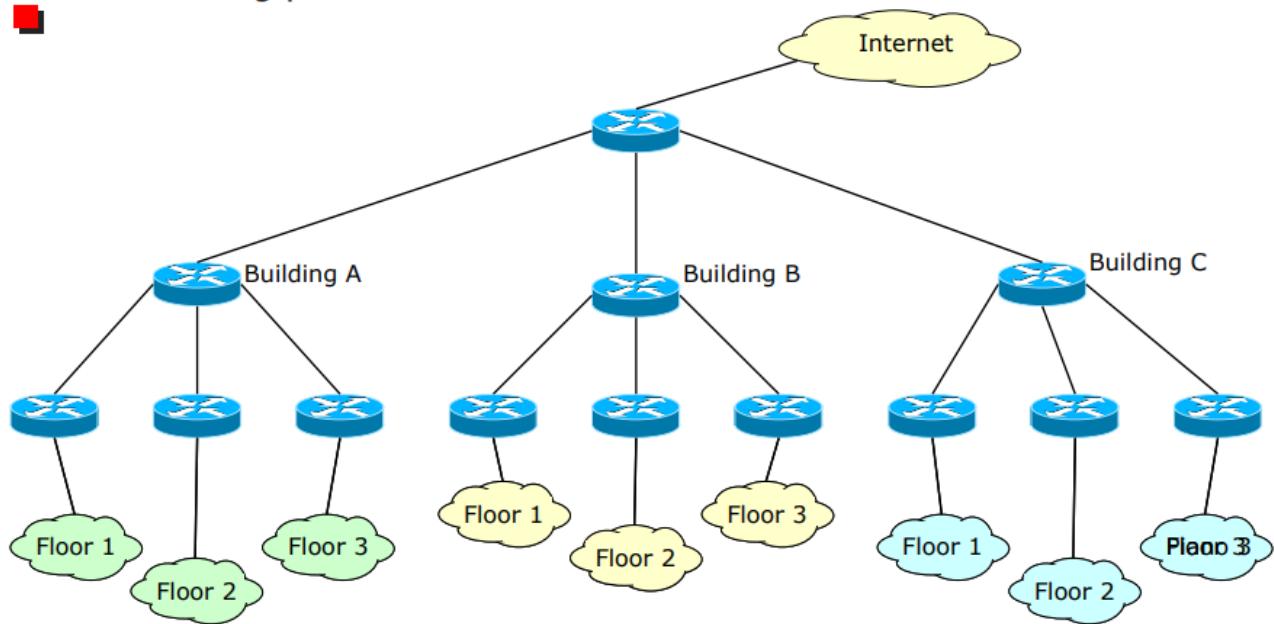
Static Route:
S FD00:0:0:2::/64 FD00:0:0:1::1



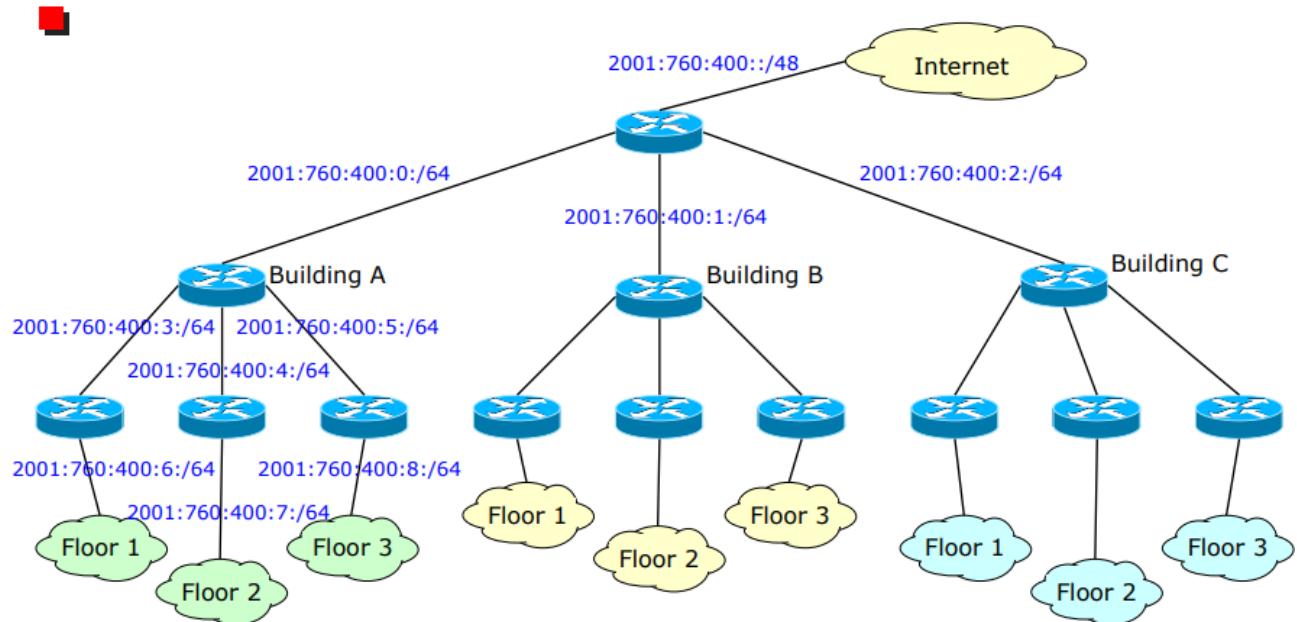
Si utilizzano dappertutto delle reti /64 cambiando semplicemente un prefisso per generare le 3 reti che servono.

Exercise 6: addressing plan

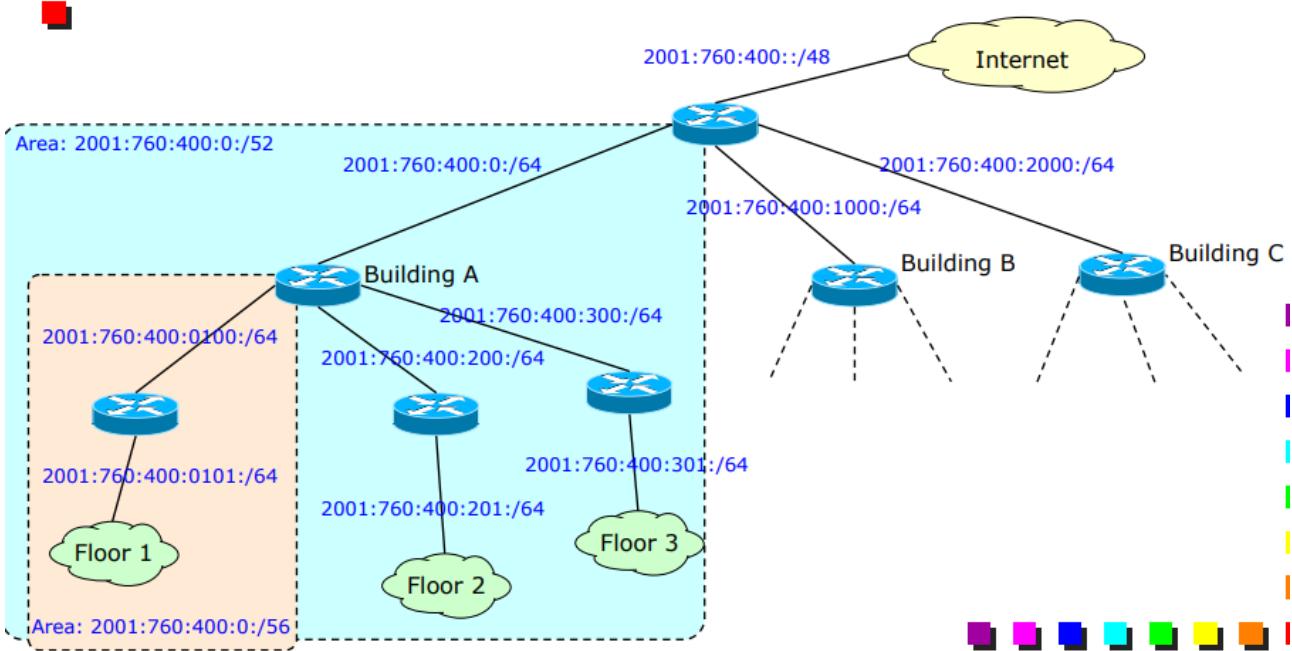
Define an IPv6 addressing plan for the network shown in the figure, and the interfaces where it makes sense to enable the router advertising process.



Per risolvere il problema dell'indirizzamento con reti di questo tipo, la prima soluzione è quella di dire che la rete, non essendo molto grande, non è necessario ottimizzare il routing sulla rete. L'importante è che il routing sia ottimizzato nella rete internet. Quindi la soluzione banale è quella di assegnare le reti **a piacere**.



È possibile però realizzare un routing leggermente più complicato per ottimizzare il routing sul router che si collega direttamente a internet.

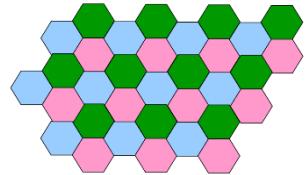


Esattamente come fatto con IPv4 è possibile creare delle aree partendo dall'address range in /48, cercando degli address range più piccoli da utilizzare in tutta l'area evidenziata.

- Partendo dal /48 si è definita una rete /52 per gestire tutta l'area azzurra ($2001:760:400:0:/52$)
- La stessa cosa sarà necessaria anche su Building B e C (cambiando anche solo 1 bit).
- A questo punto, vista la topologia della rete, si può tentare di semplificare la vita anche a Building A creando nuovamente un'altra area /56 (*$2001:760:400:100:/56$ sulla slide l'area è errata*) per la sottorete evidenziata.
- Successivamente si creano delle reti da assegnare presi dall'address range.
- Si è scelto /52 perché ogni cifra esadecimale solo 4 bit, dunque il prossimo numero esadecimale dopo 48 è 52.

Reti cellulari

Una rete cellulare è una rete wireless divisa in **celle** all'interno delle quali sono presenti delle antenne che trasmettono il segnale ai vari dispositivi. Una cosa importante che questi dispositivi devono supportare è il **movimento tra le celle** (non solo quando il telefono è in standby ma anche mentre l'utente sta utilizzando la rete). Questo sistema viene definito **hand over**. Le celle sono organizzate come in foto.



Le possibilità per realizzare questo tipo di rete sono due:

- Si posizionano le antenne al centro dell'esagono facendole irradiare in tutte le direzioni
- Si posizionano le antenne ai confini tra le varie celle andando a irradiare non in tutte le direzioni ma solo in una porzione di circa 120°. Per evitare problemi si utilizzano frequenze diverse.

In realtà però la forma delle celle è ben lontana dall'essere esagonale, a causa di svariati motivi:

- **Potenza di emissione:** l'operatore di rete può scegliere di non definire celle tutte della stessa dimensione.
- **Altezza antenna:** più l'antenna è alta, più è possibile andare distante
- **Guadagno di antenna:** parametro dell'antenna che tiene in considerazione la direzionalità dell'antenna (non usa la stessa potenza in tutte le direzioni)
- **Morfologia dell'area geografica:** in caso di ostacoli il segnale non si propaga linearmente come farebbe in pianura
- **Fading:** quando un segnale rimbalza esso può arrivare comunque al ricevitore che si vede più copie dello stesso segnale
- **Macrocelle/microcelle:** la dimensione delle celle può variare.

All'interno delle celle sono presenti i terminali mobili che devono comunicare con l'antenna. Nel momento in cui sono presenti molti utenti che condividono le frequenze della cella si crea il problema dell'**accesso multiplo**. Se due utenti trasmettono insieme si hanno le **collisioni**. Nelle reti cellulari non si utilizzano gli approcci utilizzati nel WiFi/Ethernet ma altri, tra cui:

- FDMA – *Frequency Division Multiple Access*
- TDMA – *Time Division Multiple Access*
- CDMA – *Code Division Multiple Access*
- SDMA – *Space Division Multiple Access*

La soluzione del CDMA è quella utilizzata per il 3G ma poi abbandonata con il 4G. La tecnica di base si basava sull'applicazione di un **codice** assegnato dall'antenna al segnale.

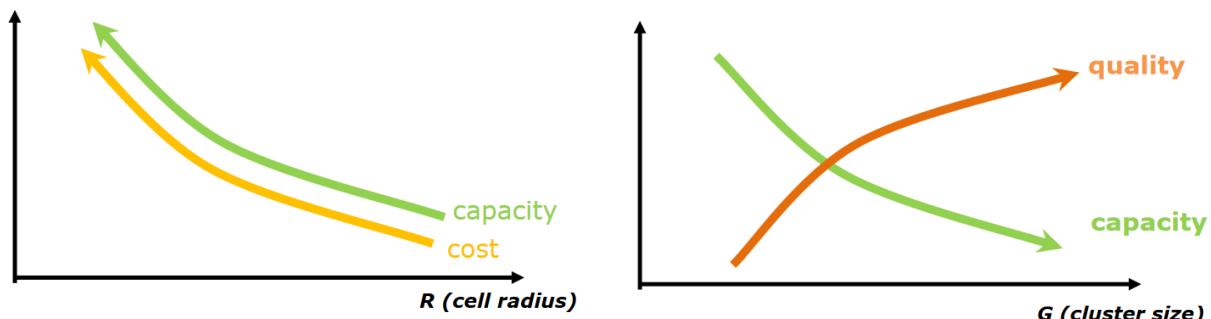
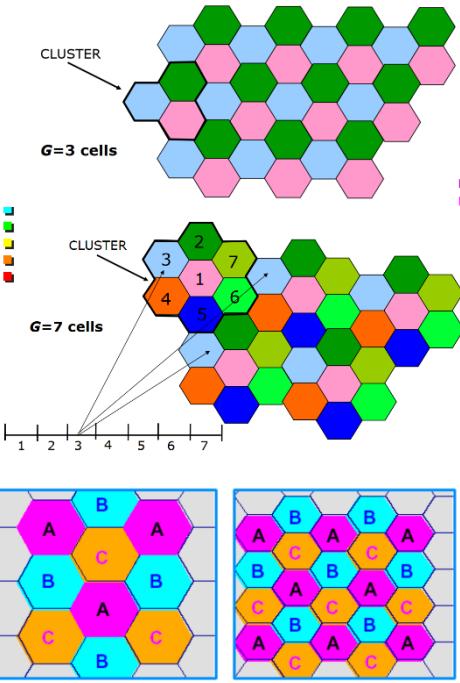
Definendo s_k come il segnale da inviare (ad esempio la voce) e con c_k il codice assegnato, è possibile ottenere $S_k = s_k \cdot c_k$. Questo segnale generato viene inviato da ogni dispositivo all'antenna, la quale li riceverà come $\sum S_i = \sum s_i \cdot c_i = A$. Essendo c_i un segnale ortogonale, è possibile riottenere s_k moltiplicando $A \cdot c_k$ (annullando così la moltiplicazione iniziale per c_k). Sembra una buona idea ma è molto complicato, perciò è stato abbandonato.

Nella versione più semplice possibile, la rete cellulare può essere vista come una rete **FDMA** con riutilizzo delle frequenze in locazioni diverse. Per fare ciò si definiscono dei **cluster G** di celle adiacenti e all'interno di questi cluster si utilizzano **tutte le frequenze possibili**. Ad esempio, se si ha un cluster di 4 celle e il provider possiede 16 frequenze, si andrà a dividere il 16 fra le celle del cluster e dunque ogni cella avrà 4 frequenze a disposizione. Nella foto è presente un cluster a 3 celle. Le celle che riutilizzano le stesse frequenze (ma sufficientemente distanti tra loro) vengono definite **co-channel cells**. Nella seconda foto è presente un cluster a 7 celle.

Combinando insieme la presenza dei cluster con un raggio opportuno è possibile ottimizzare la copertura sulla base del numero di persone da gestire. Se si utilizzano delle celle molto grandi (raggio R molto grande) la capacità della cella diminuisce. Più il raggio è grande, più la capacità diminuisce, perché si hanno meno frequenze.

Accorciando il raggio R delle celle si guadagna in capacità ma aumentano anche i **costi**. Se si va a diminuire 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**. Però, aumentando la grandezza di G aumenta la distanza tra le co-channel cells e dunque le interferenze sono minori e aumenta la qualità.



Il problema del riuso delle frequenze è quello delle **interferenze** come citato più volte. Dovendo ottimizzare l'utilizzo delle frequenze si cerca sempre di avvicinare il più possibile le frequenze riusate. Per tale motivo sono state definite delle tecniche per limitare le interferenze.

- **Splitting:** utilizzare in maniera opportuna una combinazione di microcelle e macrocelle
- **Sectoring:** limitare l'apertura di irradiazione delle antenne
- **Tilting:** si cerca di variare l'angolo con cui si va a irradiare
- **Creazione di femtocelle:** celle molto piccole, portatili (utilizzate magari agli eventi tipo stadi).

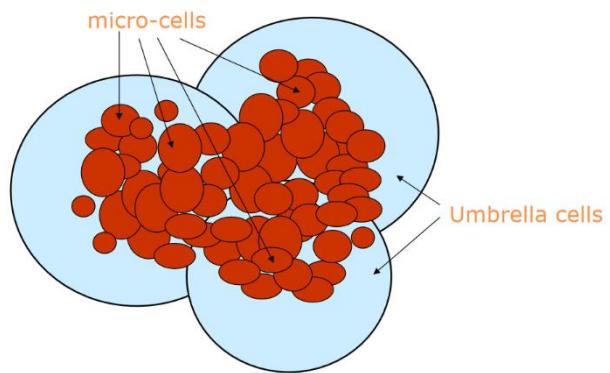
Splitting

Ottimizzazione nell'utilizzo combinato di macrocelle e microcelle. Se si ha un'area altamente popolata è necessario avere un'alta capacità, dunque è meglio creare celle più piccole per soddisfare tutte le esigenze di quella particolare area. Infatti, la potenza di trasmissione delle microcelle è di circa 3W, contro i 20-60W delle macrocelle utilizzate nelle aree rurali.

Cell shaping

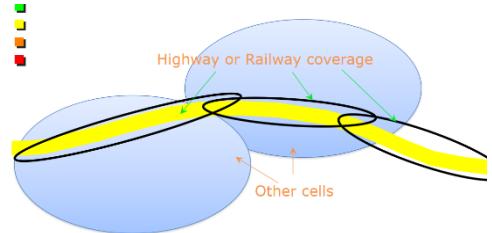
Serve per tentare di dare una forma alle celle che meglio si addice alle esigenze. Una prima forma di cell shaping può essere quella di alternare microcelle e macrocelle.

Dalla foto si nota che sono presenti delle grosse macrocelle dette **umbrella cells** che coprono un'ampia zona di territorio. Al loro interno vengono comunque definite delle microcelle per cercare di ottimizzare l'allocazione degli utenti su celle che possono soddisfare meglio le proprie esigenze. Se un utente si muove verso una certa direzione lungo le microcelle è più consigliato utilizzare le macrocelle per tale utente, poiché nel caso in cui si utilizzassero le microcelle bisognerebbe gestire più volte il passaggio da una cella a un'altra.



Viceversa, se all'interno delle microcelle è presente un utente che è fermo non conviene utilizzare una frequenza nelle macrocelle ma è meglio utilizzare una microcella.

Un'altra cosa interessante che si fa è quella di definire delle microcelle con forme particolari. La forma allungata è quella che conviene utilizzare lungo una strada o una ferrovia.



Power Control

L'idea è semplice: se si ha un cellulare che deve trasmettere all'antenna è logico regolare la potenza di trasmissione del cellulare sulla base della distanza e della qualità presente in quel momento tra cellulare e antenna. Per cercare di controllare la potenza di trasmissione si utilizzano delle tecniche di controllo/misura della qualità istantanea della trasmissione. Sulla base del risultato della misura si regola la potenza di trasmissione. Esistono vari modi per effettuare il power control, uno tra tutti l'**open loop**.

Open loop

L'idea è quella di dire che invece di misurare la qualità con cui si ha inviato un qualcosa, si misura la qualità su ciò che si riceve. Non è la tecnica migliore poiché il canale down-link è diverso da quello up-link quindi la qualità potrebbe non essere la stessa per i due canali e dunque la misura potrebbe non essere valida in up-link.

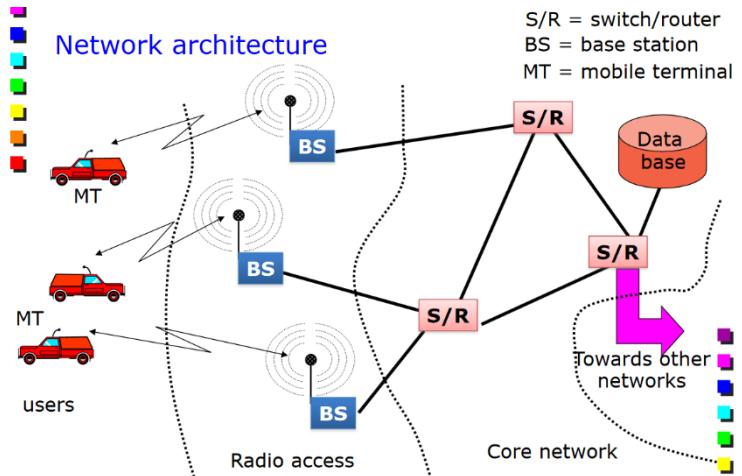
Allocazione della frequenza

Negli esempi si è mostrato il numero di cluster e la divisione delle frequenze nei vari cluster. Questo non è sempre ciò che si fa, ma è possibile invece allocare più frequenze a una certa cella perché essa magari potrebbe avere più utenti da coprire. Più in generale, oltre a una differenza in termini numerici tra una cella e l'altra sono presenti proprio diverse politiche di allocazione di frequenza:

- *Fixed Channel Allocation (FCA)*: si divide la frequenza tra le celle e rimane quella
- *Dynamic Channel Allocation (DCA)*: le frequenze vengono distribuite alle varie celle anche in maniera non equa ma che varia nel tempo.
- *Hybrid Channel allocation Scheme (HCS)*: una porzione di frequenze rimane fissa mentre l'altra viene allocata dinamicamente.

Architettura di rete

È presente una prima parte di accesso dove sono presenti gli utenti chiamati **mobile terminal (MT)** seguiti dall'**accesso radio** ovvero le trasmissioni wireless verso l'antenna che è connessa alla **base station**. La base station è il punto di accesso per gli utenti alla rete cellulare. Le varie base station sono spesso connesso tramite canali wired alla **core network**, dove sono presenti i commutatori di rete che possono essere degli switch a circuito oppure dei router in base alla generazione di rete cellulare. I link **non** sempre sono in fibra, ma possono anche essere dei link wireless (**ponti radio**). Le base station tra loro **non** parlano. Un altro componente importante nelle reti cellulari è il **database**, necessario per memorizzare le informazioni degli utenti. Inoltre, i router della core network hanno anche il compito di connettere la rete cellulare verso altre reti (sia altre reti cellulari che cablate).



Procedure base

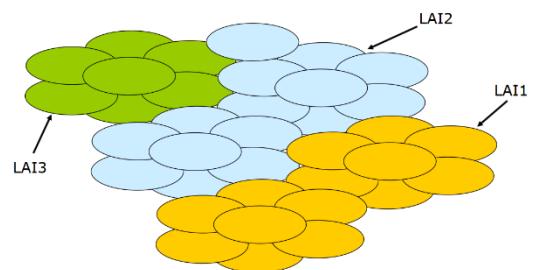
Registrazione

La registrazione permette a un terminale mobile di connettersi alla rete tramite una autenticazione. La registrazione viene effettuata quando si accede alla rete ma anche prima di accedere a un servizio (ad esempio quando si vuole effettuare una chiamata). Ciò che avviene nelle reti mobili è che la registrazione viene effettuata **periodicamente** per aggiornare le informazioni nel database.

Mobility management

Il mobility management è basato su 4 procedure più semplici:

- **Roaming:** si intende la capacità della rete di tracciare il terminale utente quando si sposta. Il roaming è basato sulla presenza di **location areas (LA)**. Si vuole capire in quale location area si trova l'utente, e deve essere garantito. In generale le location area sono formate da più celle e vengono identificate da un **Location Area Identifier (LAI)**.
- **Location updating:** per supportare il roaming è necessario capire quando l'utente ha cambiato location area. La location updating fa sì che l'utente possa comunicare alla rete il proprio LAI quando questo cambia. Sostanzialmente viene inviato in broadcast su tutte le celle della location area, periodicamente, un **control channel** in cui viene scritto il LAI.
- **Paging:** quando un utente deve essere raggiunto (ad esempio chiamata per un utente) la rete ha l'informazione sulla location area in cui si trova l'utente ma non sa in quale delle varie celle si trova l'utente. Dunque, viene inviato un **paging message** di nuovo in broadcast su tutta la location area in cui si trova l'utente. L'utente riconosce che il messaggio è per sé stesso e dunque reagisce di conseguenza. Da quel punto in poi per la rete è più semplice contattare l'utente.
- **Handover:** permette all'utente di spostarsi continuando a utilizzare i servizi che sta già utilizzando. Esistono vari tipi di handover.



- **Intra/inter cella**, il primo molto poco utilizzato poiché si cambia solo la frequenza di utilizzo della cella, mentre il secondo viene spesso utilizzato.
- **Soft/hard**: quello hard è più complesso dal punto di vista di gestione della qualità poiché non si ha la possibilità di mantenere la connessione con la vecchia cella mentre ci si collega con quella nuova (mentre nel soft handover è possibile).
- **Iniziati da MT/BS**: si distingue il tipo di handover sulla base di chi si è accorgo che è necessario effettuare l'handover.
- **Backward/forward**: chi gestisce l'handover, cioè se la BS di partenza o quella di arrivo della nuova cella.

1G: First Generation

La prima generazione era una tecnologia completamente analogica detta **TACS** basata su FDMA. Nel 2005 TIM ha completamente stoppato il servizio e le frequenze sono state date al GSM.

2G: Second generation

GSM non vuol dire 2G. Il GSM è lo standard Europeo di seconda generazione. Ad esempio, negli Stati Uniti lo standard 2G è l'IS-95 (ma ciò non vuol dire che negli USA non esista il GSM). Inoltre, GSM e IS-95 non sono due tecnologie compatibili poiché il GSM usa una tecnica FDMA/TDMA mentre l'IS-95 usa la tecnica CDMA. Il più grande cambiamento rispetto alla prima generazione è che si è passati a una trasmissione **digitale** e in più si sono aggiunti gli SMS.

2.5G: Extended second generation

Caratterizzato dal GPRS/EDGE in Europa (IS-95B negli USA) ha introdotto il servizio dati (comunicazione a pacchetto) con basso data rate.

3G: Third generation

Lo standard della terza generazione è l'**UMTS** in Europa e in Giappone, mentre negli Stati Uniti è l'**IS-95C**. Il grosso cambiamento (a livello europeo) è stata l'introduzione del CDMA, inclusa una serie di migliorie tecnologiche a livello fisico ed anche l'introduzione di **servizi multimediali** a pacchetto (supporto a internet).

3.5G: Extended third generation

Standard del 3.5G è l'**HSPA/HSPA+** con migliorie esclusivamente a livello fisico che hanno portato a un aumento delle prestazioni col trasferimento dei dati.

4G: Fourth generation

Conosciuto anche come **LTE** (Long Term Evolution) ha portato a un grosso incremento del data rate. Tutto ciò è ottenuto principalmente grazie a innovazioni a livello fisico, ma si basa anche per la prima volta su **IP** (dunque non più a circuito). Inizialmente era pensato **solo per i dati**, ma successivamente tramite la definizione dello standard **VoLTE** è stata introdotta la possibilità di utilizzare la voce. Lo standard 4G aveva dei requisiti che andavano rispettati, ma l'LTE standard non li rispettava tutti. Per tale motivo viene definito uno standard 3.9G. Soltanto nel 2011 con l'LTE-Advanced si è riusciti a rispettare tutti i requisiti.

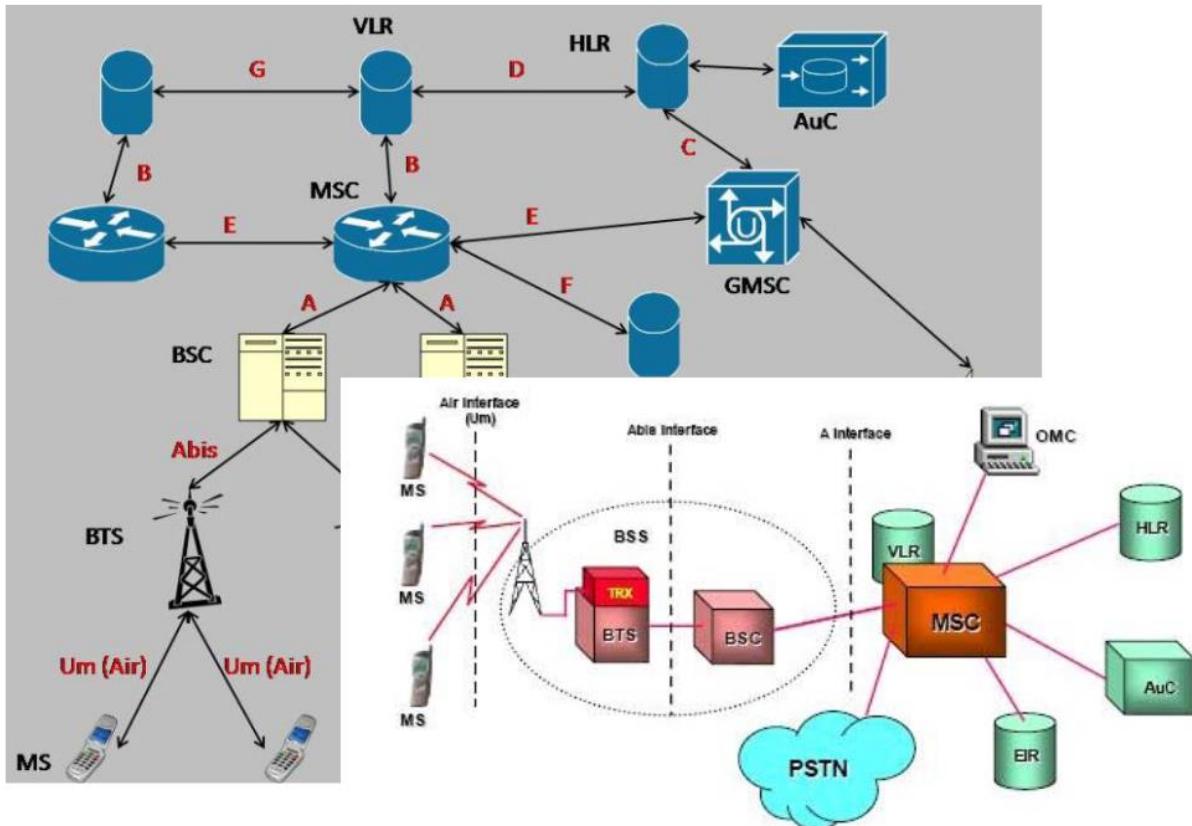
5G: Fifth generation

L'idea del 5G è quella di integrare tutte le reti wireless. Dunque, si tratta di un'evoluzione dell'LTE-A, del WiFi e delle comunicazioni mmWave). Tutto l'edge della rete si vuole uniformare al 5G. Nel core della rete si utilizzano due tecnologie:

- *Software defined Networking (SDN)* per il controllo della rete
- *Network Function Virtualization (NFV)* per l'implementazione dei servizi. L'obiettivo di queste tecnologie è quello di offrire flessibilità ma soprattutto **slicing** agli utenti.

GSM – Global System for Mobile Communications

GSM era la seconda generazione ed offriva servizi molto basici (voce sostanzialmente) a 13kbit/s in full rate oppure a 6.5 kbit/s in half rate, tra cui anche servizi aggiuntivi come SMS, richiamata su occupato e telefax.



L'architettura del GSM non è molto diversa da quella vista prima. La **mobile station** (MS) sarebbe il dispositivo utente (MT) + la SIM. I MS si collegano in maniera wireless alla **BSS** che è formata dalla **BTS** (Base Transceiver Station) e dalla **BSC** (Controller). La BTS è tutta la componentistica hardware che si occupa dell'invio dei segnali, mentre il **BSC** è l'intelligenza che comanda la **BTS**. A seguire è presente la **core network** dove sono presenti tutti gli elementi tra cui l'**MSC** (switch a circuito), l'**HLR** e il **VLR** (entrambi database utente), l'**AuC** per l'autenticazione e l'**EIR** che tiene traccia dei dispositivi (in particolare dei dispositivi rubati). Inoltre l'**MSC** può agganciarsi anche alla **PSTN** (telefonia fissa).

- **Mobile Station (MS):** dispositivo utente che si collega alla rete formato da MT+SIM.
- **Subscriber Identity Module (SIM):** scheda formata da un piccolo processore con un po' di memoria che serve per avere accesso alla rete.
- **Base Station Subsystem (BSS):** si tratta in sostanza di una Base Station (BS) che si divide però in *Base Transceiver Station (BTS)* e *Base Station Controller (BSC)*.
- **Base Transceiver Station (BTS):** La BTS è l'insieme di tutta la componentistica che permette di inviare e ricevere segnali da mobile terminal. Le BTS inviano un segnale a una potenza che è due ordini di grandezza rispetto a quella per il broadcasting in TV. La Base Station, inoltre, trasmette solo se c'è qualcuno che sta utilizzando il servizio. La BTS gestisce verso gli utenti 32 canali FDM (sostanzialmente 32 frequenze per BTS). Metà di queste sono per l'up-link, l'altra metà per il down-link. Queste 16 frequenze per link sono divise in slot. Ogni canale FDM include 8 canali TDM. In altre parole, su ogni frequenza è possibile far convivere 8 utenti. Quando uno slot viene assegnato (ad esempio 0) si trasmette solo in corrispondenza del time-frame 0, raggiungendo 13kbps in full-rate. Se invece si trasmette in half-rate si trasmette ogni due time-frame 0.

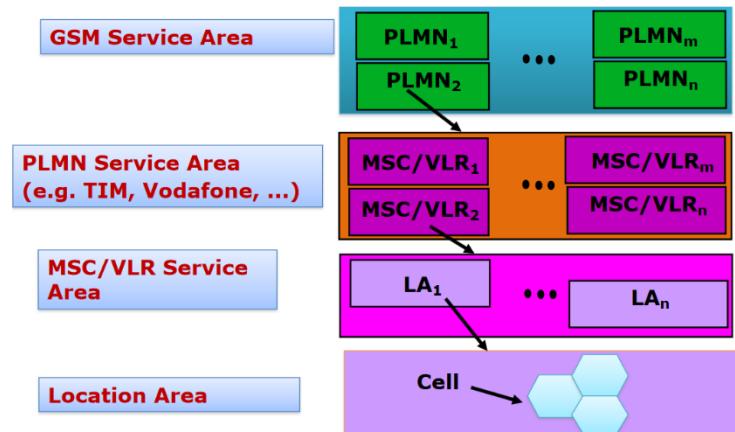
- **Base Station Controller (BSC):** non si occupa in prima persona di inviare il segnale (per quello c'è la BTS) ma è il controller di varie BTS (da 10 a 100 BTS possono essere gestite dallo stesso controller). Di solito la BTS e la BSC comunicano lungo un link wired. La BSC, al di là di gestire (paging, risorse radio, misura qualità del segnale, handover tra BTS controllati dallo stesso BSC), effettua anche il **voice transcoding** da 13kb/s a 64 kb/s per permettere di uniformare la tecnologia con la modulazione che viene utilizzata negli switch a circuito tradizionali utilizzati nella telefonia.
- **Network and Switching Subsystem (NSS):** all'interno dell'NSS sono presenti tutti i componenti già elencati e si occupa della gestione della chiamata (apertura del circuito) ed anche il supporto alla mobilità (roaming, location updating, etc..).
- **Mobile Switching Center (MSC):** switch PCM che si occupano di gestire l'allocazione delle risorse (creazione di un circuito end-to-end) e dunque si occupano di effettuare il routing di una chiamata tra due MT oppure tra MT e rete fissa. Il routing in questo caso è basato sul LAI, poiché il numero di telefono in questo caso si sposta. Se bisogna andare verso la rete di un altro operatore è necessario utilizzare il *Gateway Mobile Switching Center (GMSC)* che è in grado di interfacciarsi verso reti esterne rispetto a quella dell'operatore.
- **Home Location Register (HLR):** effettua lo store di informazioni permanenti dell'utente tra cui l'id, i servizi a cui può accedere un utente e vari parametri di sicurezza, ma anche di informazioni dinamiche tra cui il *VLR Identifier* (a chi chiedere per avere informazioni specifiche per quell'utente).
- **Visitor Location Register (VLR):** serve per tenere traccia degli spostamenti. Si tratta di una copia dell'HLR ma con informazioni leggermente diverse (ad esempio il LAI).
- **Equipment Identity Register (EIR):** database di dispositivi rubati.
- **Authentication Center (AuC):** necessario per definire dei canali sicuri con cui l'utente può comunicare con la rete.

Identificazione del MT

- **IMSI e TMSI.** L'**IMSI** (*International Mobile Subscriber Identity*) è un identificatore univoco della SIM ed è ciò che viene memorizzato nell'HLR. Nel VLR viene invece memorizzato il **TMSI**, un codice temporaneo che rappresenta il mobile terminal. L'**IMSI**, essendo un dato sensibile, viene memorizzato sul canale solo quando strettamente necessario e ove possibile viene utilizzato il **TMSI**. Il **TMSI** (*Temporary Mobile Subscriber Identity*) serve per la stessa cosa dell'**IMSI**, viene memorizzato nel VLR e tutte le volte che viene utilizzato viene cambiato. Inoltre, ne viene comunicato un altro immediatamente all'utente attraverso un canale sicuro.
- **MSISDN** (numero telefonico) e **MSRN** (*Mobile Station Roaming Number*). L'**MSRN** è il numero non da HLR (perché non è un'informazione stabile e permanente) ma da VLR, poiché varia ogni volta che l'utente si sposta. L'**MSRN** viene utilizzato per effettuare il **routing della chiamata**.
- **IMEI** (*International Mobile Equipment Identity*) è un codice univoco che identifica il dispositivo. Viene utilizzato nell'EIR per tenere traccia dei dispositivi rubati.

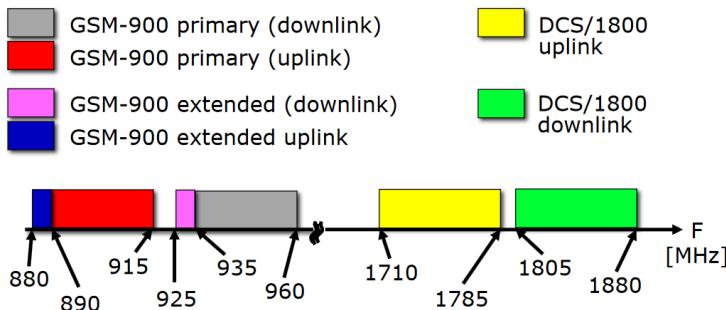
Aree GSM

L'immagine a destra riepiloga quanto segue: più celle formano una **location area**. Più location area sono gestite dallo stesso gruppo **MSC/VLR**. Tanti di questi gruppi formano la rete dell'operatore detta **PLMN Service Area**. Infine, le reti di tanti operatori vanno a generare la rete globale detta **GSM Service Area**.



- Ogni cella viene identificata da un identificativo detto **Cell Global Identifier (CGI)** che viene irradiato all'interno della BTS in modo da permettere ai terminali di ricevere questa informazione esattamente come viene fatto con il LAI.
- Un identificatore di Location Area viene inviato in broadcast per la location updating, ma è anche possibile per il mobile terminal di sapere in quale cella si trova.

Frequenze GSM in Europa



Il GSM utilizza il **FDD (Frequency Division Duplex)** system, ovvero utilizza frequenze diverse in up-link e in down-link.

La figura mostra la banda dei 900 e dei 1800 MHz.

Canali fisici GSM

Un canale fisico è l'insieme di risorse che permette di effettuare una comunicazione. In questo caso del GSM, i canali fisici vengono realizzati tramite un approccio **FDMA/TDMA** che permettono di dividere lo spettro sia in frequenza che nel tempo. L'FDM genera una serie di canali in frequenza ciascuno ampio 200 kHz. Ogni singolo canale FDM è diviso in TDM frames, ognuno dei quali viene diviso in 8 slots.

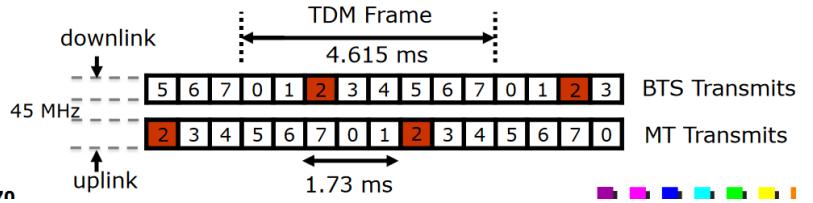
$$\text{Frequenza} + \text{slot di tempo} = \text{canale fisico}$$

Si nota che all'interno dei canali fisici le trasmissioni vengono organizzate in **bursts** (all'interno di un blocco di tempo si invia un burst che somiglia a un pacchetto – ma non lo è, poiché siamo in comunicazione a circuito - poiché è una serie di bit).

Utilizzando tutto ciò è possibile trasmettere complessivamente (su tutte le frequenze) sul mezzo 271 kbit/s.

Frame GSM

Ipotizzando di utilizzare un full rate, andare a segnare uno slot per frame in uplink ed uno in downlink genera dei canali a 13 kb/s. Tra l'altro, se si divide il 271 kbit tra tutti i canali disponibili salta un numero leggermente maggiore di 13, questo perché ai 13kbit/s bisogna aggiungere dei bit per la **codifica di canale** per il controllo degli errori.



Se il sistema decide di assegnare lo slot 2 ad un certo utente, lo slot in uplink non è perfettamente allineato con quello in downlink (ma solo la numerazione è shiftata, perché il sincronismo è mantenuto). Questo permette di utilizzare il transeaver, trasmettere in uplink, avere tempo per spegnere il trasmettore e accendere il ricevitore (downlink). Questo permette di utilizzare un solo transeaver attivo per volta, questo per migliorare gli aspetti di costruzione che di carica della batteria.

Frequency Hopping (FH)

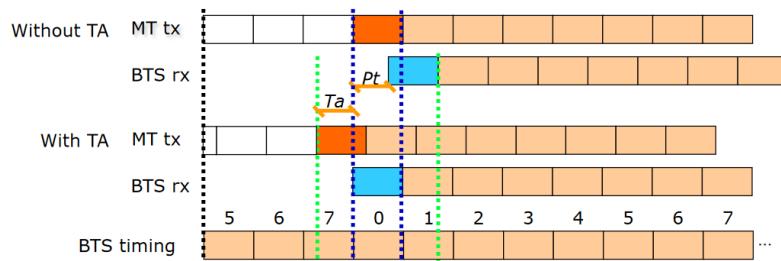
Un'altra procedura attuata nella creazione di canali fisici è quella di riservare più di un canale per utente. Questo implica che ogni singolo utente ha uno slot riservato su diverse frequenze. Questo fa sì che di volta in volta l'utente possa **saltare in frequenze diverse**. Questo riduce il fading, ma riduce anche la capacità.

Ostacoli tecnologici

Sono presenti due ostacoli da affrontare:

- **Tempi di propagazione non nulli:** si risolve con la tecnica del *timing advance*
- **Accensione/spegnimento amplificatori:** si risolve con l'introduzione di *bande di guardia* tra i time slot

Timing advance

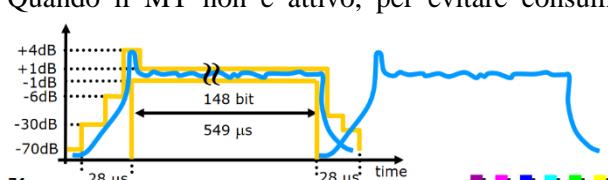


Il segnale di sincronismo è dato dalla BTS. Sul primo terminale senza timing advance accade che se il timeslot 0 viene assegnato, il terminale invia il burst di pacchetti nello slot (in foto quello arancione). Però, il terminale mobile non è vicino alla BTS ma potrebbe essere distante, dunque il ritardo di propagazione non è nullo. Il burst inizia a viaggiare nella rete e arriva sulla BTS andando a sporcare il timeslot successivo, a causa di ritardi di programmazione (in figura è blu). Dunque, se qualcuno sta trasmettendo nel timeslot successivo, si possono avere delle collisioni.

Il problema si risolve tenendo presente che esiste un ritardo di propagazione non nullo, lo si misura tramite alcune tecniche e si anticipa la trasmissione sul mobile terminal in modo tale che quando il primo bit del burst arriva sulla BTS non si va più a sporcare il timeslot successivo. Questa procedura è detta **timing advance**.

Ramp up/down

Quando il MT non è attivo, per evitare consumi di batteria eccessivi (e per ridurre le interferenze) il trasmettitore viene spento. Questi tempi di accensione/spegnimento non sono però nulli. Per risolvere questo problema sono state inserite delle **bande di guardia** e ciò riduce la durata del timeslot.



Tipi di burst nella trasmissione GSM

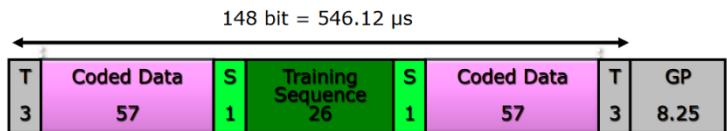
Esistono cinque tipi di bursts nel GSM:

- **Regular:** si inserisce **traffico** dati oppure **informazioni di segnalazione**. Le informazioni inviate dentro gli slot si chiamano **channel**.
- **Access:** utilizzato quando il terminale mobile non ha ancora un canale dedicato da una certa BTS. Si utilizza per accedere ai servizi della rete (es. quando bisogna effettuare una chiamata)
- **Synchronization:** utilizzato dalla BTS per inviare verso i MT delle informazioni di sincronizzazione
- **Frequency correction:** inviato dalla BTS per far sì che i MT si possano agganciare correttamente alle frequenze disponibili all'interno della cella
- **Dummy:** inviato sulla frequenza principale detta **main carrier**, la prima frequenza della cella (sia in up che in down) viene chiamata C0. Infatti, delle 16 frequenze che ci sono, in realtà sono 15 perché la prima non può essere usata dagli utenti. Se nella C0 non ci sono altre informazioni da trasmettere, viene inviato un dummy burst. In tutti gli slot vuoi viene inviato il dummy burst, utilizzato ad esempio per misurare il livello di potenza offerto dalla BTS (utile per gli handover, per capire quale cella può offrire il servizio migliore).

Struttura del “Regular” burst

La struttura è quella in figura.

- **GP (Gap Period)** è la banda di guardia.
- I **T** bit sono un'ulteriore banda di guardia.
- **Coded data:** sono presenti due gruppi da 57 bit (per un totale di 114 bit di dati) e sono i veri bit di dati. A causa della codifica di canale saranno un po' meno di 114. Per la voce si arriva a 13kbps (aggiungendo la codifica di canale), mentre per i dati (solo dal GPRS) si arriva fino a 9.6kbps (**SOLO GPRS**)
- **S** sono gli stealing bits dicono se i gruppi di coded data sono utilizzati per la voce o per il trasporto di informazioni di controllo

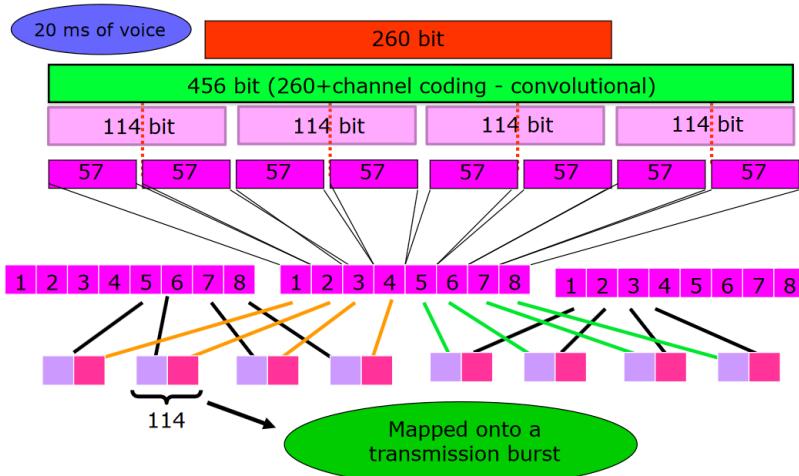


Struttura dell'Access burst

La struttura è quella in figura.

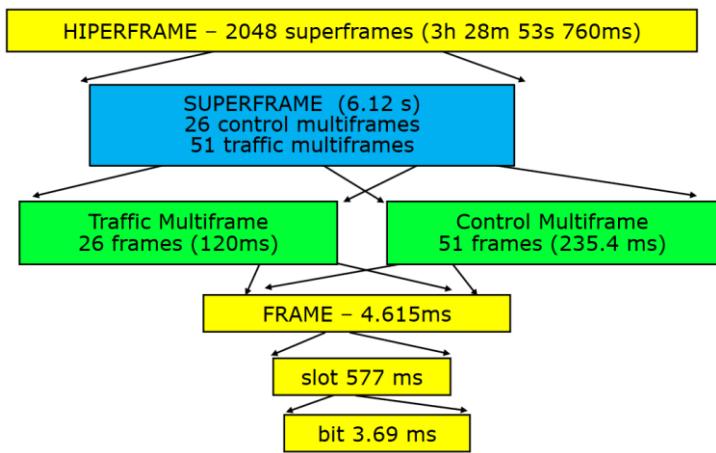
- Sono di nuovo presenti bande di guardia dette **Extended T-bits**. Esiste una sequenza specifica (11001010) poiché essendo la prima informazione che il MT manda verso la BTS è necessario che i due siano sincronizzati.
- **Sync** bits servono per fare la valutazione del timing advance.
- **Coded data** sono eventuali comunicazioni da inviare alla BTS
- **T e Ext GP** formano di nuovo una banda di guardia molto grande. Questo perché il timing advance non è stato ancora calcolato perciò ci si pone nel caso peggiore.

Interleaving



lo shift dei gruppi (spostandoli di ordine) per poi ricostruire i gruppi da 114 bit dove l'ordine dei bit non è lo stesso. Per fare l'interleaving bisogna dunque attendere 20ms di voce. I 114 bit andranno poi inseriti nel burst. Questa tecnica si utilizza per **mitigare gli errori**. La codifica di canale funziona bene se gli errori sono pochi (e non contigui).

GSM framing



solo dopo 3 ore, utile ad esempio per le procedure di **encryption**.

Nella slide è presente anche la durata di un singolo bit, seguito dallo slot e dal frame. Più frame vengono raccolti all'interno di un **traffic/control multiframe**. Più multiframe formano poi un **superframe** e infine 2048 superframe formano l'**hiperframe**.

Tutto ciò viene fatto perché sulla base di questa classificazione viene definito un numero detto **frame number (FN)** che identifica il singolo frame all'interno di tutta la gerarchia di frame della durata di 3h. Il frame number si ripeterà

The Frame Number (FN) module is:



Canali logici GSM

Nella rete GSM (e in generale con le reti cellulari) cosa viene trasmesso all'interno dei vari slot viene chiamato **logical channel**. Dunque, con canale logico si specifica **cosa** si va a trasmettere. Esistono due tipi di canali logici:

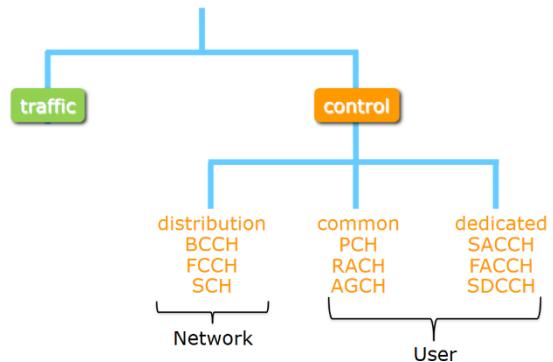
- **Control channels**: si trasportano informazioni di controllo e segnalazione
- **Traffic channels**: si trasferiscono i dati (nel caso del GSM la voce)

Control Channels

I control channels si dividono a loro volta in due tipi:

- **Network signaling control channels**: servono alla rete per segnalare qualcosa all'utente (parametri cella, info di sincronizzazione)
- **User signaling control channels**: definisce tutto ciò che va ad essere legato al controllo lato utente (controllo di chiamata, segnalazione livello di qualità durante la chiamata)

I network control channel sono BCCH, FCCH, SCH mentre gli user control channel sono a loro volta suddivisi in **common** e **dedicated** e sono tutti quelli mostrati nella figura a lato.



Network signaling

Vengono anche chiamati **distribution** channels. Sono canali in cui viene effettuata la trasmissione di informazioni di interesse generali. In particolare sono presenti:

- **Frequency Correction Channel**, FCCH: si indicano alcune informazioni che permettono agli utenti di **correggere** la frequenza e rimanere allineati al carrier canale. Si tratta di un canale unidirezionale.
- **Synchronization Channel**, SCH: all'interno viene inviato il Base Station Identity Code (BSIC) che è uno dei codici identificativi della cella. In questo BSIC è presente anche il codice dell'operatore, dunque l'utente può leggere di chi è quella cella su cui lui sta ricevendo l'SCH. All'interno del canale è presente anche il Frame Number.
- **Broadcast Control Channel**, BCCH: la cella effettua il broadcasting di una serie di informazioni utili a tutti, ad esempio il LAI viene inviato qui.

Uso del broadcast control channel

- Il MT viene acceso
- Il MT fa lo scan di tutte le frequenze di GSM che sente (o di quelle memorizzate nella SIM)
- Si aggancia alla C0, dove arrivano i tre canali citati prima. Dentro l'SCH l'utente può trovare il codice dell'operatore, per agganciarsi o meno se l'operatore corrisponde a quello della SIM. Una volta fatto l'aggancio, dal BCCH arrivano alcune informazioni che servono per poter entrare a far parte della rete dell'operatore.

Tutte le informazioni ricevute sono **per cella**. Quando il MT entra in una nuova cella, il processo ricomincia.

User signaling

Esistono due tipi di user signaling control channel:

- **Common Control Channels (CCCH)**: utilizzati quando l'utente non ha ancora a disposizione un canale dedicato (di utilizzo occasionale). I common control channels sono tutti **unidirezionali**.
- **Dedicated Control Channels (DCCH)**: utilizzato dall'utente come canale dedicato, utilizzato in modo periodico

Common control channels (CCCH)

Sono i seguenti:

- **Paging channel (PCH)**: quando la base station deve trovare l'utente, deve inviare in broadcast una richiesta (es. "telefonata per utente, chi è?") e ciò viene fatto utilizzando proprio questo canale. Il broadcast viene fatto su **tutte le celle**, cioè su tutta la location area.
- **Random Access Channel (RACH)**: serve all'utente per accedere alla rete. Serve per **richiedere l'allocazione di un circuito**. Unico canale soggetto a **collisione** perché i MT possono essere molti e non avendo ancora richiesto nulla potrebbero parlare nello stesso momento. Il RACH si utilizza: **per ricevere telefonate** e l'utente risponde al paging dicendo di essere disponibili per la telefonata; **per effettuare una telefonata**; **per far partire la location update** quando si accorge di essere uscito da una location area e si è entrati in un'altra.
- **Access Grant Channel (AGCH)**: utilizzato dalla BTS per rispondere al RACH. È una sorta di ACK. La BTS oltre a rispondere mette in piedi delle risorse dedicate per l'utente (no canale dati), un canale di controllo detto *Stand-alone Dedicated Control Channel* (SDCH). Questo è il primo canale dedicato a un utente specifico.

Quando il mobile terminal invia la richiesta sul RACH ci aggiunge il corrispondente **frame number** (FN). Succede dunque che nella AGCH, la BTS ci aggiunge l'FN (frame number) letto nel RACH. Ciò vuol dire che se si vede tornare indietro una risposta sull'AGCH con tale FN è perché la BTS ha ricevuto il messaggio e lo ha capito senza problemi di collisione. Questo si usa perché è impossibile che due persone, utilizzando due slot diversi, trasmettano nello stesso slot. Se l'FN è diverso, un altro utente ha usato uno slot diverso.

Dedicated Control Channels (DCCH)

I DCCH sono i seguenti:

- **Stand-alone dedicated control channel (SDCH)**: primo canale di controllo dedicato all'utente per mettere in comunicazione, in modo dedicato, il MT con la rete. È possibile, ad esempio, far passare informazioni dedicate per creare un canale sicuro (autenticazione).
- **Slow Associated Control Channel (SACH)**: bidirezionale, invia informazioni utili al terminale durante l'effettuazione di una certa chiamata. È ad esempio possibile inviare informazioni sul timing advance (perché se ci si sposta nella cella bisogna ricalcolarlo), oppure vengono anche inviate le stesse informazioni che viaggiavano sul broadcast control channel (che viaggiano su C0) perché se nel mentre si è ottenuto un canale dati, non avendo più transeaver, non è possibile ascoltare ciò che sta viaggiando su C0. Dunque, le informazioni vengono ripetute in questo canale.
- **Fast Associated Control Channel (FACH)**: viene implementato in maniera diversa rispetto allo slow e viene utilizzato se è necessario informare la controparte in maniera rapida (ad esempio, per effettuare un handover). Per implementarlo, si utilizzano gli **stealing bit** presenti all'interno del regular burst.

Traffic channels

Vengono utilizzati per trasportare voce o traffico dati, ma in realtà in GSM è presente solo voce.

- **Full Rate Traffic Channel:** TCH/F a 13 kb/s (uno slot in ogni frame)
- **Half rate Traffic Channel:** TCH/H a 6.5 kb/s (uno slot ogni due frame)

Un altro servizio presente nella rete GSM è l'**SMS**. Il servizio viene offerto dal **Service Center**, un'entità che gestisce gli SMS. Se il MT è spento, quando c'è un SMS per quel MT l'SMS viene depositato sul Service Center. Quando invece il MT è attivo in "idle", cioè non è ancora attivo il TCH (o SACH), la base station utilizza lo Stand-Alone control channel poiché si assume che l'utente ha già effettuato l'accesso alla rete. Se invece il MT è attivo ed è impegnato in una chiamata, viene utilizzato l'SACH.

Mapping dei canali logici sui canali fisici

TCH e SACCH

Questi due canali vengono mappati sulla stessa frequenza e lo stesso slot (stesse risorse). Questo è possibile perché, partendo dalla velocità del full rate di 13kb/s, vanno aggiunti altri bit della codifica di canale fino ad arrivare a 22.8kb/s. Bisogna però aggiungere dell'altro: in base a cosa vien fuori andando a considerare uno slot ogni 8, con tale banda e tale durata, salta fuori un risultato pari a 24.7 kb/s (più di ciò che serve). Succede, dunque, che ogni multiframe (ogni 26 frame) vengono utilizzati 22 frame per trasferire la voce ma due vengono saltati. Degli altri 2 slot, uno viene utilizzato per **mappare lo SACCH**. Da qui si capisce perché si chiama Slow: bisogna sempre attendere che arrivi quel frame in cui lo slot è idle (cioè non si manda la voce). L'altro slot libero in idle viene utilizzato per **effettuare le misure** di qualità della chiamata, agganciandosi per **un solo frame** alla C0 per **valutare la qualità delle altre celle** (serve per l'handover). Ciò è necessario poiché è presente un solo transeaver.

FACCH

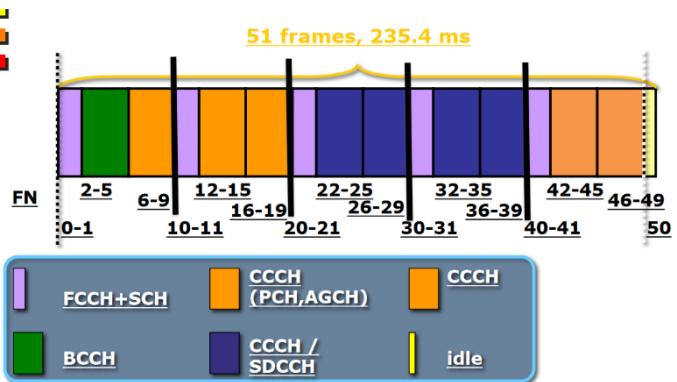
Viene mappato rubando le risorse al TCH tramite lo stealing bit.

Gli altri canali

Tutti gli altri canali di cui si è discusso vengono tutti mappati sulla C0. In particolare, *FCCH* ed *SCH* sono mappati sul timeslot 0 della C0. Invece, *BCCH*, *PCH*, *RACH* (simile a Slotted Aloha), *AGCH* possono utilizzare qualsiasi slot pari della C0. Infine, *SDCCH* utilizza il timeslot 0 o timeslot 1 su C0.

Il TS0 (Timeslot 0) in C0 viene utilizzato in maniera particolare come in figura. Ogni 10 frame viene inviato l'FCCH+SCH. Viene ripetuto così frequentemente perché sono molto importanti visto che viene utilizzato dai MT per potersi agganciare ma anche per effettuare la correzione di frequenza.

Invece, il BCCH viene inviato una volta soltanto. Si tratta di scelte architettoniche in base a quanto sono importanti determinate informazioni.

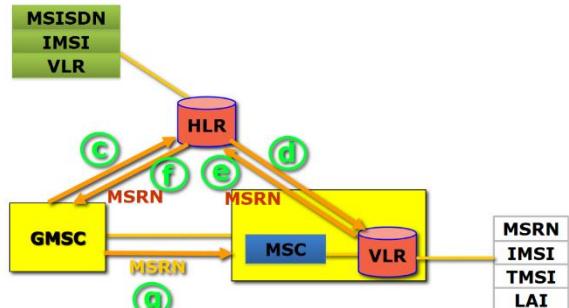


Procedure GSM

- Chiamata diretta al MT
- Handover

Chiamata (da telefono fisso) a MT

- L'utente da telefono fisso digiterà l'MSISDN (Numero di telefono).
- A questo punto, grazie al numero di telefono, la rete fissa può capire qual è l'operatore mobile (operazione non molto semplice da quando esiste la portabilità).
- Effettua dunque il routing della chiamata (creando il circuito) verso il GMSC (switch della rete dell'operatore mobile che si interfaccia con l'esterno).
- Il GMSC interagirà con l'HLR, dentro cui sono presenti i dati riferiti al particolare MT, incluso il numero di telefono.
- L'HLR individuerà l'IMSI del MT ed anche il VLR in cui, in quel momento, si trova il MT.
- L'HLR invia al VLR una **roaming information request**, per sapere l'MSRN (numero aggiuntivo, diverso dall'MSISDN necessario per raggiungere l'utente)
- Il VLR invia l'MSRN all'HLR
- L'HLR invia al GMSC l'MSRN per permettergli di procedere con il routing della chiamata verso l'MSC corretto.
- Dunque, il GMSC effettua il routing verso l'MSC che corrisponde a quel VLR.
- Sempre grazie all'IMSI e al VLR, l'MSC riesce a identificare la location area in cui si trova il MT
- L'MSC invia un messaggio di PAGE (sul PCH) alle BSC con dentro l'identificativo dell'utente. All'interno del PAGE message non verrà inviato l'IMSI, ma il TMSI, per poter permettere il broadcasting in sicurezza.
- L'MT invia un Access Burst sul RACH
- La BTS risponde sull'AGCH e assegna all'MT un SDCCH
- Utilizzando l'SDCCH sarà possibile effettuare numerose operazioni, tra cui
 - Autenticazione
 - Encrypt dei dati
 - Re-allocazione del TMSI
- Una volta completate tutte le operazioni, sarà possibile far partire la chiamata e l'MSC e la BTS assegneranno un TCH al MT.
- L'MT invierà una notifica che il telefono di destinazione sta suonando/se ha risposto
- L'MSC collega la chiamata al TCH e conferma l'inizio della chiamata.



Handover

La prima cosa da discutere in ambito di handover è quello di capire come vengono effettuate le misure per capire se è disponibile una cella migliore di quella attuale. Per fare ciò, si effettua una procedura detta **locating**: procedura utilizzata dal MT per inviare le misure che ha inviato alla BSC.

Locating

- La BSC comunica al MT (periodicamente, dentro al SACCH) l'ID di 6 BTS che dovrebbero essere misurate.
- Il MT misura:
 - L'intensità del segnale ricevuto sul TCH
 - La qualità sul TCH
 - Nello slot idle si va a misurare l'intensità del segnale sulla C0 (una cella alla volta)
- Periodicamente, l'MT invia (sempre nel SACCH ma in uplink) le misure alla BTS, la quale invia alla BSC le misure insieme alle misure della BTS.
- In tutte le misure viene associata una penalità alla cella corrente, per evitare l'effetto ping-pong tra due celle.
- Sulla base di ciò, la BSC decide se è il momento di effettuare l'handover

I motivi per effettuare l'handover possono essere molteplici, tra cui: intensità del segnale entro una certa soglia; distanza tra BTS e MT; bilanciamento del carico; manutenzione.

Tipi di handover

- Intra-cellula
- Tra BTS dello stesso BSC
- Tra BTS di diverse BSC ma stesso MSC/VLR
- Tra BTS di diverse BSC e diverse MSC/VLR

La regola comune tra tutti i tipi deve **sempre avvenire entro 100 ms**.

Handover tra BTS di diverse BSC e diverse MSC

- La BSC per iniziare l'handover effettua una comunicazione al MSC corrente, il quale contatta il nuovo MSC.
 - Il nuovo MSC assegna un numero all'handover e lo notifica al vecchio MSC
 - Il nuovo MSC apre un circuito verso un nuovo BSC, il quale a sua volta crea un circuito verso la nuova BTS e riserva un TCH.
- Quando tutto è pronto, il nuovo MSC notifica quello vecchio
 - La vecchia BSC invia uno *switch command* al MT (sul FACCH): l'MT si collega al nuovo TCH
 - Il vecchio MSC effettua lo switch della chiamata su un nuovo circuito e rilascia quello vecchio

Il processo citato è un **hard handover**.

General Packet Radio Service (GPRS)

L'idea di GPRS è quella di costruire in **parallelo** alla rete a circuito basata su MSC del GSM, una rete di packet switches detta **Serving GPRS Support Node** (SGSN) per la gestione di traffico dati. Tra i SGSN si trova il **Gateway GSN (GGSN)** che è l'equivalente del GMSC, cioè il router che gestisce il collegamento verso l'esterno. Con ciò, gli utenti sono **anche** identificati da un indirizzo IP.

Sull'accesso radio cambia il modo in cui vengono utilizzati gli slot. Nel GSM i dati sono organizzati in bursts (1 slot ciascuno), invece in GPRS si è cercato di migliorare il supporto all'invio di pacchetti. Sono stati definiti i **radio blocks** (4 slot ciascuno) con la possibilità di creare un canale con 2 *radio blocks* avendo 8 slots.

Dunque, è possibile che le chiamate e i pacchetti si vadano a contendere i blocchi. In realtà, però, in GPRS le chiamate **hanno sempre la priorità**. Si tratta, in altre parole, di un optional del GSM.

Enhanced Data rate for Global Evolution (EDGE)

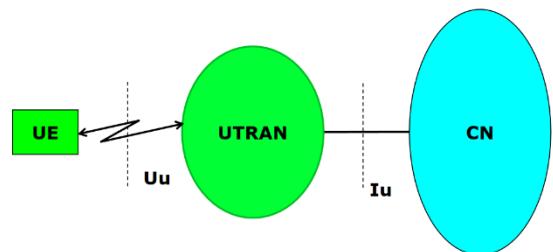
È stato migliorato il livello fisico dal punto di vista della codifica e della modulazione. La struttura rimane identica alla precedente. Con la stessa banda a disposizione si riescono a inviare dati maggiori.

Universal Mobile Telecommunication System (UMTS)

Uno degli obiettivi di questa tecnologia è stato quello di **garantire la retrocompatibilità** con i sistemi 2G (tra cui la possibilità di effettuare un *vertical handover*, cioè un cambio di tecnologia durante una chiamata). Un altro target era quello di supportare i servizi multimediali e di internet.

L'architettura di rete è la stessa definita per il GSM.

I terminali mobili (UE) accedono alla radio access network detta **UTRAN** (*UMTS Terrestrial Radio Access Network*), che si va ad interfacciare con la Core Network.



User Equipment (UE)

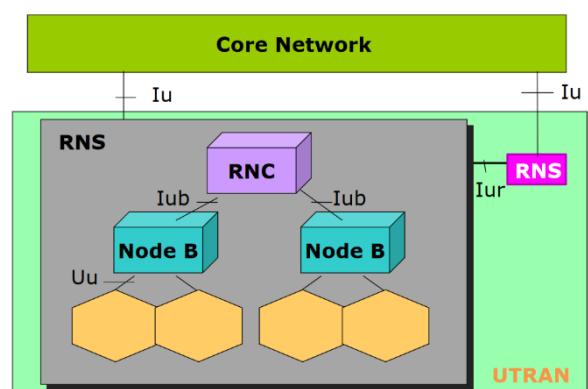
Formato da:

- USIM (Universal Subscriber Identity Module)
- TE (Terminal Equipment): include i protocolli di livello 3/4 (TCP/IP) e anche il supporto al CDMA.
- MT: funzionalità relative alla trasmissione radio

UTRAN

L'UTRAN ha una struttura molto simile a quella del GSM, le BTS non sono più presenti ma sono presenti dei nodi equivalenti detti **Node B**. Il BSC è stato sostituito dall'RNC (Radio Network Controller). Il BSS (che in GSM era BTS+BSC) viene detto **RNS** (Radio Network Subsystem).

La differenza sta nel fatto che uno stesso Node B può gestire più celle (mentre in GSM una BTS gestiva una sola cella).

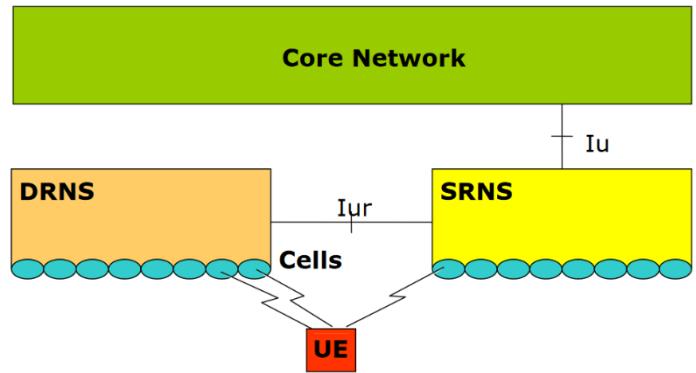


Macrodiversità

Una novità dell'UMTS è che l'utente può collegarsi a più celle (anche gestite da diversi RNS), come in foto. Ciò è possibile perché viene utilizzata la CDMA, e dunque viene utilizzata la stessa frequenza per comunicare con le celle, ma utilizzando il codice appropriato.

Essere connessi a più celle è un vantaggio per la microdiversity. Vuol dire che è possibile sfruttare le celle per raggiungere una destinazione utilizzando più percorsi. Ricevendo il segnale da più celle è possibile utilizzare degli algoritmi per sceglierne alcuni o combinare le copie ricevute.

In up-link però la situazione è più complicata perché inviare più copie dello stesso dato non hanno senso di esistere nella core network. Per gestirla, si sono definite le due entità **DRNS** ed **SRNS** (*Serving RNS*). L'SRNS è l'unico che davvero controlla ciò che l'UE fa. Invece, il DRNS (*Drift RNS*) è di supporto e può gestire le copie che arrivano dall'User Equipment. Il DRNS semplicemente invia il traffico all'SRNS, il quale utilizza gli algoritmi per scegliere quale copia utilizzare.

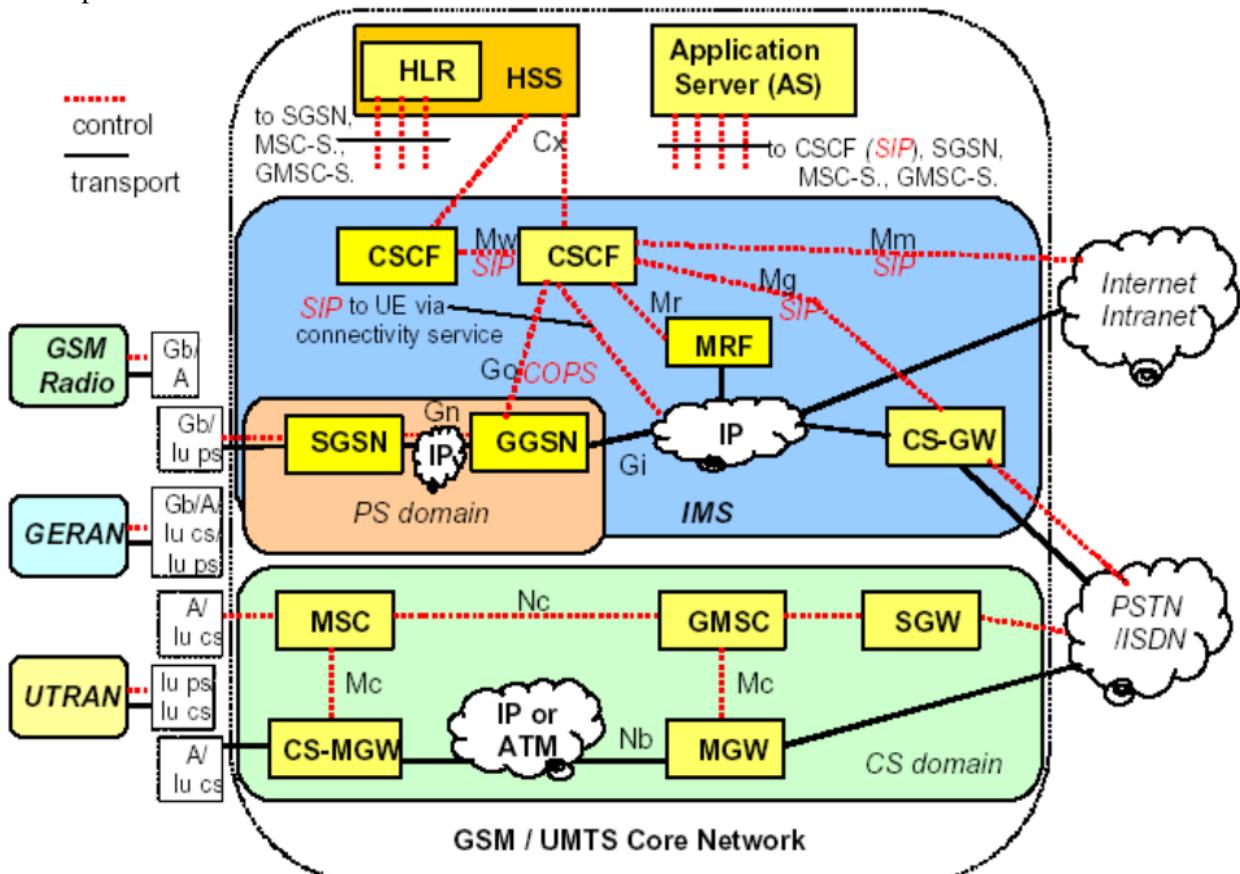


Rispetto a GSM e GPRS

- La RAN è basata su CDMA e non più su FDMA
- Dalla release 5 in avanti si è stabilito che la core network fosse completamente IP (inclusa la voce).
- Al livello applicativo, è stato introdotto l'*IP Multimedia Subsystem* (IMS), che supporta i servizi multimediali basati sul protocollo IP

Core Network

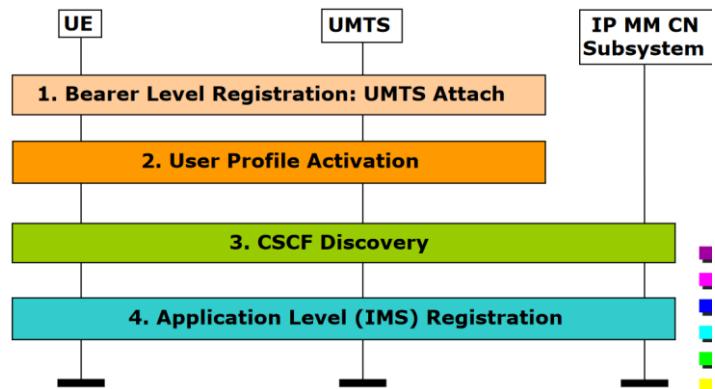
In foto è presente la core network UMTS.



- HSS è un nuovo database detto *Home Subscriber Service* che contiene anche l'*home location register* (HLR).
- L'application server serve per il supporto dell'IMS.
- La Core Network in azzurro (+ blocco arancione che indica i router) rappresenta la parte “a pacchetto” della Core Network UMTS.
- La parte detta **CS domain** è la parte a circuito per la gestione delle chiamate. La rete a circuito dovrebbe servire soltanto per la parte di **segnalazione** per la gestione della chiamata, ma alla partenza della chiamata per riservare le risorse si va a creare un traffico IP che viaggia nella rete a pacchetto. Questa non è una cosa che ha molto senso poiché si crea sia il circuito che la rete a pacchetto, infatti molti operatori continuano ad usare la rete a circuito per le chiamate (quindi l’obbligo di IP per la release 5 non è completamente vero).
- L’IMS (linee tratteggiate rosse) è un framework per la gestione della segnalazione di servizi multimediali basato su **SIP** (standard per la segnalazione VoIP). Veniva utilizzato per fornire servizi di videochiamata o di streaming video non basato su internet. Oggi è sparito a causa dell’utilizzo di Internet che lo ha completamente sostituito.

Procedure per accedere ai servizi UMTS

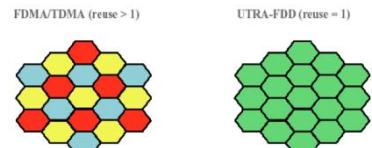
Per accedere ai servizi UMTS, l’UE deve far partire una procedura detta **UMTS Attach** (autenticazione, assegnazione canali) seguita da una procedura di **attivazione del profilo utente** fornendo l’UE di un **indirizzo IP**. Per accedere anche ai servizi di IMS è necessario effettuare una **CSCF Discovery** seguita da una **IMS Registration** (si comunica al server l’indirizzo IP).



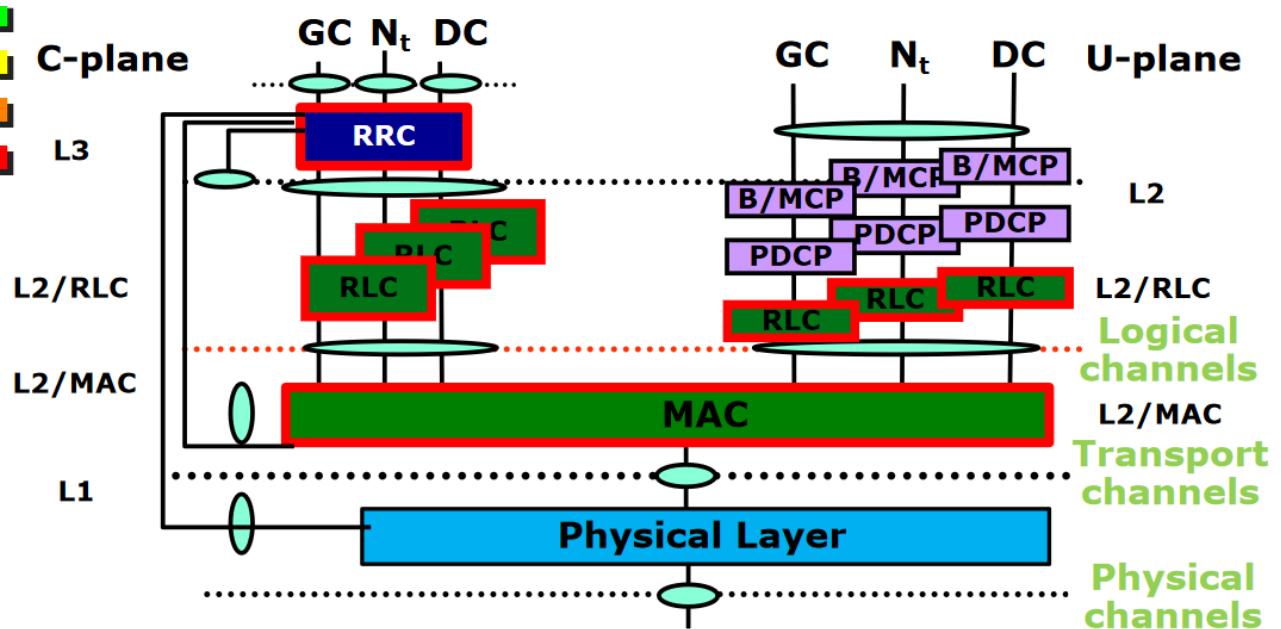
Tecnologie UTRAN

L’UTRAN può essere basato su:

- **FDD+CDMA**: frequenze separate per uplink e downlink per evitare la creazione di interferenze se si usa lo stesso codice. In questo caso non si ha più il riutilizzo di una frequenza in posti diversi, ma la stessa in tutte le celle. L’accesso avviene col CDMA ma al livello fisico (FDD) ogni canale è ancora diviso in slot. Vengono definiti dei **radio frames** che a loro volta vengono divisi in slot. Un frame è lungo 10 ms e include 15 slots. Gli slot sono utili per gestire meglio il mapping logico su canali fisici.
- **TDD+TDMA+CDMA**: si divide nel tempo l’uplink e il downlink unito a una TDMA+CDMA. Questa soluzione viene utilizzata molto poco. Può avere dei vantaggi grazie alla divisione di tempo (per garantire maggiore tempo in downlink rispetto all’uplink) ma richiede molta potenza.



Stack protocollare dell'interfaccia radio



RLC: Radio Link Control

RRC: Radio Resource Control

Lo stack frame è diviso in **Control Plane** (piano di controllo) e **User Plane** (piano dati). Sullo user plane sarà presente al livello 3 IP e al livello 4 TCP/IP. Invece, sul piano di controllo è presente anche il piano **RRC** (Radio resource control). Si nota anche la presenza di canali fisici e logici, ma anche **canali di trasporto**. Partendo dal basso si nota:

- Livello fisico
- Livello 2: MAC
- Livello 2: sublayer detto **RLC** che aggiunge ai servizi offerti dal MAC una serie di servizi, tra cui **l'error recovery** (ARQ, invio e mi aspetto risposta di conferma/errore). Questo perché i servizi da gestire sono diversi, perciò nella gestione dei dati avere tecniche ARQ è utile. (Con la voce anche se manca qualcosa si capisce comunque). L'error recovery è **opzionale** e viene utilizzato solo per l'invio di dati.
- Livello 3: **RRC** è un modulo aggiuntivo implementato nel Control Plane che aggiunge ulteriori funzioni, tra cui: gestione della mobilità, allocazione risorse radio.

Canali UMTS

- *Canali fisici*: dato da frequency channel + codice.
- *Canali logici*
- **Canali di trasporto**: indica come e con che livello di qualità l'informazione deve essere trasferita attraverso l'interfaccia radio. Questi canali sono stati aggiunti perché sono presenti maggiori servizi (voce e dati). Dentro ai transport channels vengono inviati dei pacchetti detti **Transport Blocks (TBs)**. Un Transport Block viene poi inviato verso il livello fisico ogni **Transmission Time Interval (TTI)**. Il TTI equivale a k frame, tipicamente 1 (dunque TTI=10ms). I 10 ms sono il tempo di **interleaving**, per ridurre il fading. I transport channel sono definiti da diverse caratteristiche (TB size, rate di coding e trasmissione, etc..)

UMTS soft handover

In questo caso, essendo possibile rimanere agganciati su più celle, è possibile sfruttare questa cosa per gestire meglio l'handover: se è arrivato il momento di cambiare cella, ci si sgancia da quella vecchia solo dopo essersi agganciati a quella nuova. Concettualmente è più semplice del 2G ma è più semplice da applicare.

Vertical Handover

Questo tipo di handover utilizzato per cambiare generazione di rete durante una chiamata. Sostanzialmente ha due vantaggi:

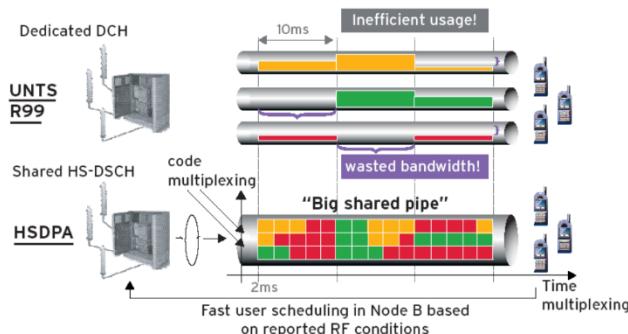
- **Coverage extension:** quando l'UE non riesce ad avere una buona copertura UMTS può fare il cambio alla GPRS e viceversa
- **Load balancing:** fatto dal provider per bilanciare il carico della rete

High Speed Packet Access (HSPA)

La prima tecnologia nata nell'ambito del 3.5G è l'HSPA. Tutte le modifiche sono state effettuate sull'interfaccia radio per migliorare il bit rate.

- Si utilizzano molto di più i canali condivisi per limitare lo spreco di risorse
- Grazie a modulazioni e schemi di codifica più efficienti si è abbassato l'interleaving time, riuscendo ad accorciare il TTI a 2 ms.
- La riduzione del TTI si è trasformato anche in una riduzione della latenza
- **Fast ARQ:** le tecniche mostrate in precedenza vengono spostate al livello fisico per ridurre la latenza

Canali condivisi



Nei canali dedicati usati fino all'UMTS, era molto comune avere dei casi in cui il frame era parzialmente o completamente vuoto (banda sprecata). Per tale motivo, vengono creati in HSPA e in HSDPA (downlink) dei canali condivisi in modo da accomodare nelle pipe in maniera più capillare i dati. Questo si può fare grazie alla riduzione sul TTI e all'utilizzo di codici diversi. Tutto quanto per sfruttare la **multiplazione statistica**.

HSPA+

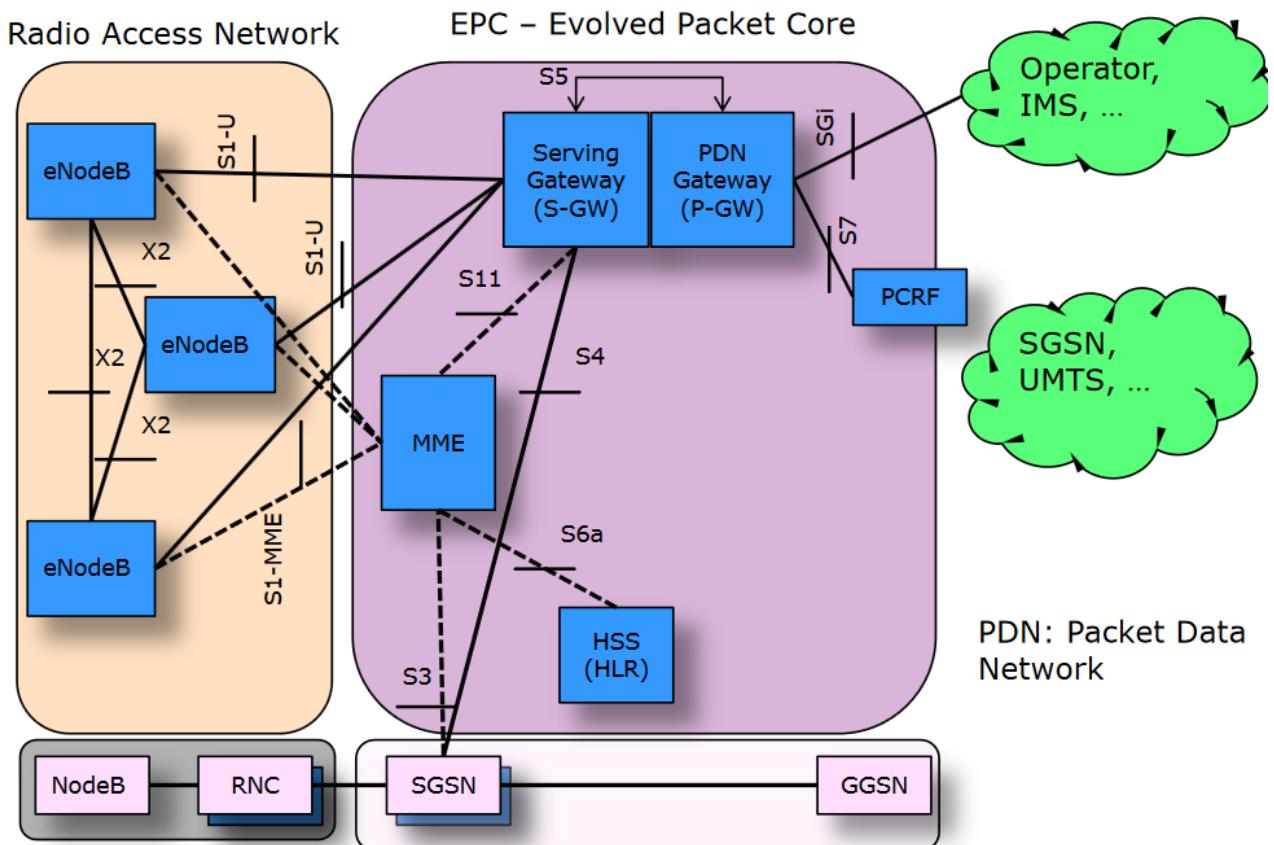
Tramite migliorie ulteriori del livello fisico come l'utilizzo di antenne **MIMO** che limitano il fading e una modulazione a 64 QAM, si è arrivati a elevati bitrate.

LTE

È stato specificato nella release 8 e 9. Le prestazioni sono: 300 Mb/s in DL e 75 Mb/s in UP, con un TTI di 1 ms. Ciò è stato possibile grazie all'utilizzo di una tecnica detta **OFDMA** (che riduce il fading), impiego di sistemi MIMO e una modulazione 64 QAM.

LTE è una rete “**all-IP**” (inclusa la segnalazione) che in principio era stata pensata solo per il traffico dati. Successivamente, tramite l'aggiunta di VoLTE si è inserito anche il traffico voce. Il VoLTE non viene sempre utilizzato (e non sempre si ha una “**all-IP**” network) ma spesso si utilizza il CSFB (*Circuit Switched Fall Back*).

Per la gestione dell'uplink e del downlink sono possibili due modalità: **FDD** (Europa e America) e **TDD** (Asia).



L'architettura LTE è molto più semplice rispetto all'UMTS poiché sono sparite tutte le voci relative all'IMS. Rimane comunque il modulo **PCRF** che permette di interfacciarsi con i servizi UMTS (incluso IMS). Si utilizza questa rete, per quanto riguarda i dati, come una rete di trasporto a pacchetto. L'RNS si è voluto, mettendo insieme le funzionalità di NodeB e di RNC in **eNodeB**. Gli S-GW e il P-GW sono i nodi di routing, mentre l'MME è la gestione del mobility management.

L'UTRAN diventa **E-UTRAN**, sostanzialmente è presente solo l'eNodeB. Inoltre, gli eNodeB possono parlare direttamente tra loro (novità rispetto alle reti precedenti).

La Core Network qui viene chiamata **Evolved Packet Core** (EPC). Il P-GW è l'equivalente del GMSC (che permette di andare dalla rete verso altre reti).

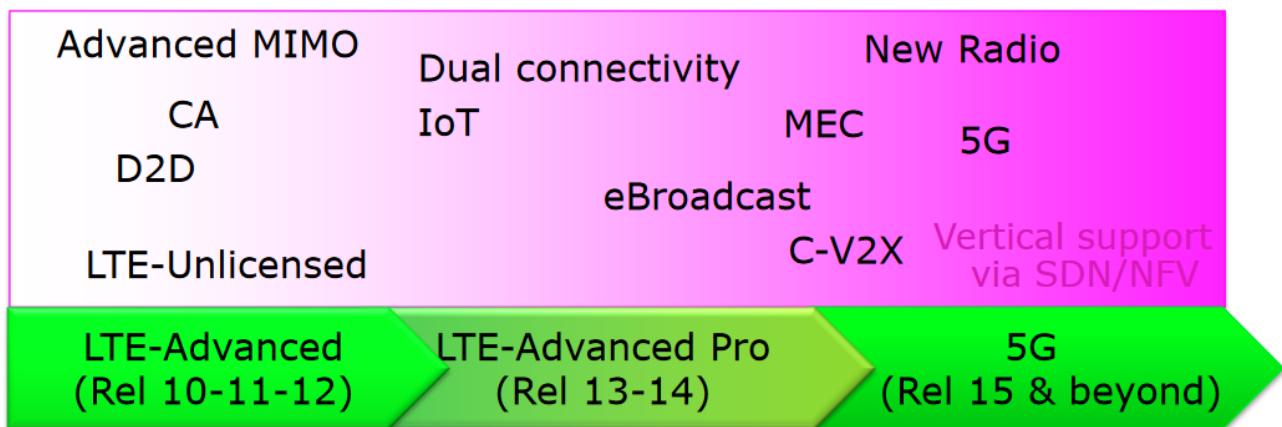
In LTE è presente la stessa struttura presente per l'UMTS se non la riduzione del numero di transport e physical channels per semplificare la struttura. Si favorisce l'utilizzo di canali **shared** e non vengono utilizzati più canali dedicati.

LTE-A

Rispetto all'LTE si va a fare ulteriori migliorie a livello fisico, con sistemi MIMO ancora più efficienti e si riesce a ottenere prestazioni fino a 3 Gbps in DL e 1.5 Gbps in UL.

Fifth Generation (5G)

Nuova generazione di reti cellulari con un target ambizioso: **generare una convergenza tra tutte le tecnologie wireless**; arrivare a 10 Gbps in downlink, migliorare il numero di dispositivi connessi simultaneamente (passare da 100k in 4G a 1M in 5G).



Si è ancora lontani dai target del 5G, ed il WiFi è tutt'ora utilizzato.

Core Network 5G

Si tratta di un'evoluzione della Core Network del 4G ma è più software and cloud based: più focus sull'automazione della rete. Inoltre, la si vuole più orientata ai **servizi**.

Servizi diversi hanno diversi requisiti tramite la definizione di diversi KPI (Key Performance Indicators):

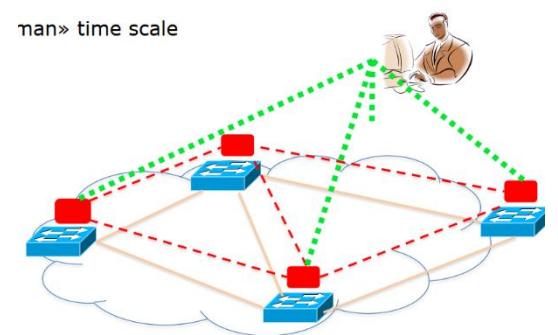
Use case	Data rate	Latency	Mobility	Availability /Reliability	Connection density
Natural disaster	0.1-1 Mbps	not critical	0-120 km/h	99%	at least 10^4 users/km ²
Public safety	10 Mbps	10 ms	0-500 km/h	99.999%	not critical
Automotive	0.5-10 Mbps 25-50 Mbps	1 ms 10 ms	0-500 km/h	99.999%	2000/km ² users
Virtual reality	100-1000 Mbps	< 20 ms	not critical	not critical	not critical

Per supportare tutti questi servizi diversi, l'idea è quella di dividere le risorse disponibili tra i vari servizi in maniera efficiente: **network slicing**. Il network slicing si può ottenere grazie a due tecnologie dette **SDN** e **NFV**.

Software Defined Networking (SDN)

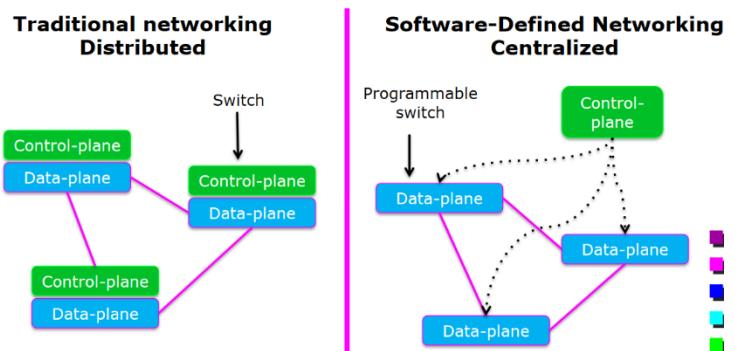
Per SDN si intende un disaccoppiamento tra control e data plane nella rete, con l'utilizzo di un **controller centralizzato logicamente**, che porta a poter gestire un control plane programmabile. Tutto ciò abbinato a una **virtualizzazione** delle risorse, che vanno a definire il **virtual data plane**.

In una rete tradizionale ogni dispositivo implementa un certo numero di servizi a livello di data plane (es. switch che effettuano forwarding, filtering, etc..). Essendo necessario lavorare a livello di pacchetto bisogna essere molto veloci, perciò tutte le funzioni vengono implementate a livello HW. Questi dispositivi posseggono anche un **control plane**, tipicamente implementato in **software** (poiché non è necessario avere prestazioni ottimali). La route computation, ad esempio, viene decisa dal protocollo di routing che “pian piano” grazie agli algoritmi di routing va a riempire le routing table. È la procedura di forwarding, di lettura della routing table, che deve essere veloce (e fa parte del data plane).

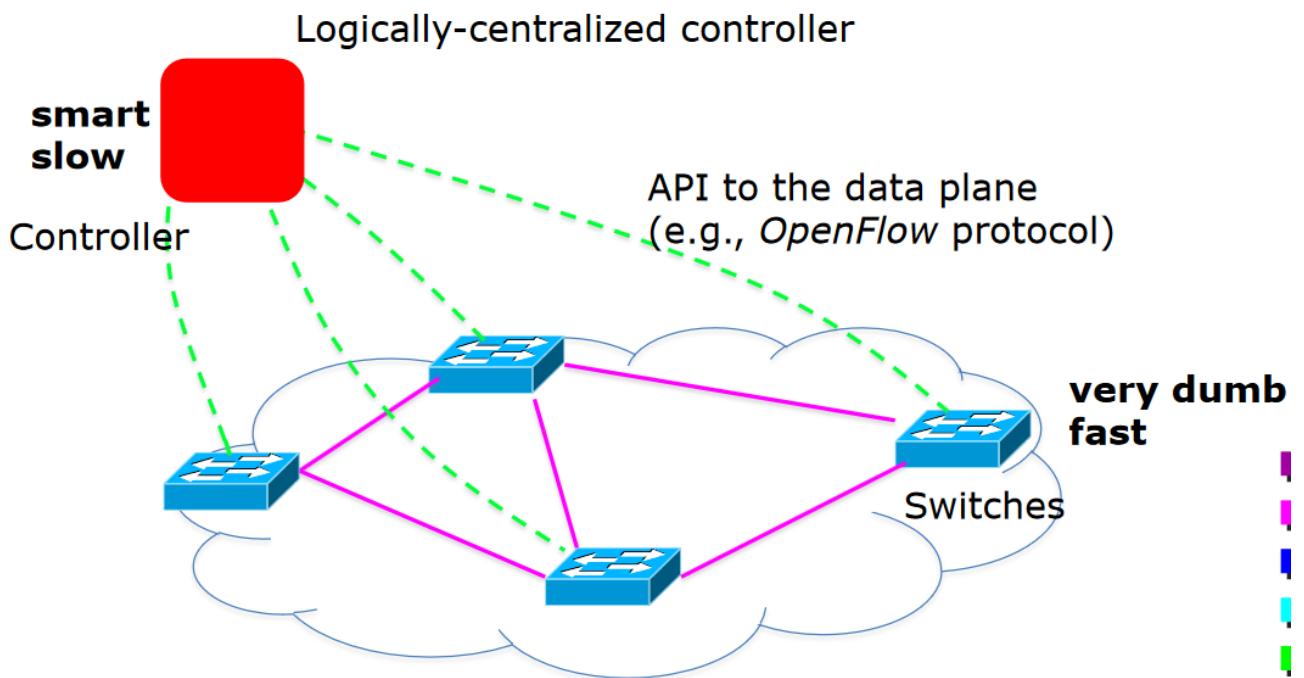


Esiste anche un **management plane**, gestito dall'esterno, che effettua il monitoraggio della rete.

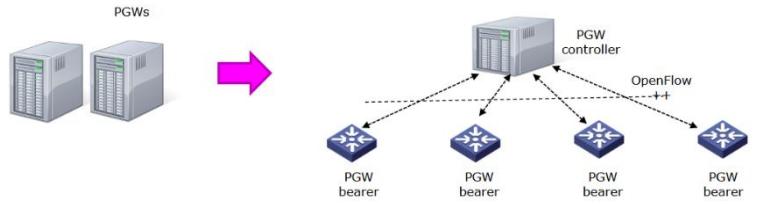
Il nuovo approccio è quello di dire che se il control plane lo si centralizza, è possibile realizzare il data-plane con dell'hardware programmabile, permettendo così di realizzare ciò che si vuole sui vari dispositivi. Diventa dell'hardware generico customizzabile (anche se di vendor diversi).



Grazie a ciò si ottiene un piano di controllo **centralizzato** lento ma molto intelligente e gli switch posseggono solo dell'hardware diventano così molto veloci ma molto “stupidi”. Serve dunque un protocollo per comunicare coi vari switch e si utilizza l'**OpenFlow**.



L'utilizzo di SDN può essere utile per la scalabilità dell'EPC. Un P-GW nell'LTE costa diverse migliaia di dollari. Sostanzialmente, ciò che si può fare per ridurre i costi e aumentare la scalabilità della rete è quello di avere tanti nodi hardware generici, collegati a un controller generalizzato che implementa le funzionalità del P-GW. Questo riduce i costi e aumenta la scalabilità.



Un'altra miglioria dell'SDN è la possibilità di implementare un load balancing efficiente. Se si ha un controller centralizzato con visibilità su tutta la rete, il load balancing globale è molto più efficiente rispetto a procedure che possono essere applicate dai singoli switch.

Vantaggio dell'SDN potrebbe essere nel vertical handover, perché non esiste un protocollo per il controllo del forwarding attraverso diverse tecnologie. È dunque necessario realizzare il vertical handover realizzando vari switch di diversa tecnologia. Un controller centralizzato con possibilità di programmare i nodi, potrebbe essere possibile usare un protocollo comune per minimizzare i tempi di setup per realizzare l'handover.

Network Function Virtualization (NFV)

Capito che il controller generalizzato può gestire diversi nodi hardware, si passa al passo successivo: sui nodi HW è possibile installare le applicazioni necessarie chiamate **Virtual NF**. Si tratta di funzioni che invece di essere realizzate in HW vengono realizzati in maniera SW attraverso procedure di virtualizzazione (es. firewall). Il risultato è un servizio più **flessibile** offerto ai verticals. Le VNF si possono spostare “al volo” in base alle condizioni della rete.

Architettura SDN + NFV (ETSI)

I vertical hanno bisogno di servizi, che vengono specificati attraverso linguaggi opportuni. Il servizio richiesto viene scomposto in funzioni elementari per definire la **service chain** (o graph) in cui l'insieme delle funzioni realizzano il servizio richiesto dal vertical.

A basso livello è presente dell'HW programmabile gestito dall'SDN. Ma grazie all'NFV è possibile installare le funzioni virtuali sulle risorse HW disponibili (virtual storage, virtual computing, virtual network). La decisione su dove piazzare tali funzioni viene gestita dall'**orchestrator**, che è in grado di spostare le funzioni nel tentativo di ottimizzare il servizio offerto ai vertical.

Grazie all'orchestratore il sistema può ottimizzare il posizionamento delle varie funzioni virtuali per cercare di rispettare i requisiti espressi dai vari KPI dei vari vertical. È dunque possibile realizzare lo **slicing**. L'orchestratore di ETSI si chiama MANO.

Routing

Si definisce **routing** il processo che permette di determinare il percorso di rete che devono seguire i pacchetti per arrivare da sorgente a destinazione. Spesso il termine viene intercambiato in modo scorretto col termine di **forwarding**.

Il **forwarding** è l'azione che consiste nell'inviare il pacchetto verso una specifica interfaccia di rete del dispositivo. Si decide, in base a una tabella di forwarding, su quale interfaccia inviare il pacchetto.

Il routing si basa su decisioni eseguite da protocolli di routing.

L'**algoritmo di routing** è l'insieme di interazioni di vari dispositivi di rete nell'ottica di permettere a ogni dispositivo di eseguire il forwarding.

Il routing è un processo **proattivo**, mentre il forwarding è **on-the-fly**.

Il processo di routing è un processo **indipendente** rispetto all'attuale traffico. Il routing ha l'obiettivo di determinare quali destinazioni sono raggiungibili e quali non lo sono. Ha, inoltre, l'obiettivo di calcolare la **best route**, il quale viene deciso in base al tipo di algoritmo applicato (il percorso migliore potrebbe essere il più breve, il più veloce o il più robusto).

Si definisce processo di forwarding la realizzazione di dove deve andare il pacchetto, realizzata di nodo in nodo. Ciò viene fatto basandosi su informazioni locali, in particolare le tabelle di routing/forwarding.

Gli **algoritmi di forwarding** si dividono in tre tipologie:

- **Routing by Network Address** (se ho destinazione IP A, prendo interfaccia 1, IP B interfaccia 2)
- **Label Swapping**: si definisce un identificativo per la **comunicazione tra host**. Tale identificativo utilizza una quantità inferiore di bit (rispetto alla tipologia precedente in cui bisogna usare necessariamente 32 bit per IPv4 e 128 bit per IPv6) e deve essere **univoco**.
- **Source Routing**: si ottiene l'informazione di come instradare il pacchetto direttamente dal pacchetto stesso

Le fasi invece sono le seguenti:

- **Routing (on-the-fly)**
 - Selezione della porta d'uscita
 - Identificazione del next-hop
- **Switching**: trasferimento alla porta di uscita
- **Trasferimento**

Gli algoritmi di routing proattivi sono:

- **Algoritmi non adattivi** (statici): non si adatta in base agli aggiornamenti dello stato della rete
- **Algoritmi adattivi** (dinamici)

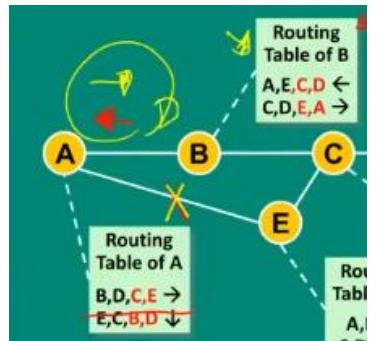
Gli algoritmi non adattivi

Si dividono in:

- **Fixed Directory routing:** tabella di forwarding statiche, configurate manualmente.
- **(Selective) flooding and derivates:** si inoltra il pacchetto in tutte le interfacce di rete senza scegliere una determinata strategia di routing. Ciò può essere fatto a priori per tutto il traffico oppure selettivo per selezionare solo determinato traffico.

I vantaggi di un algoritmo statico sono: semplicità, gestione completa da parte dell'amministratore della rete. Ciò implica che potrebbero sorgere errori e l'amministratore potrebbe non accorgersene. Inoltre, questo tipo di algoritmo **non** riesce ad adattarsi a cambiamenti di topologia.

Negli algoritmi statici viene utilizzata anche una **backup route** da utilizzare in caso di guasti singoli oppure di sovraccarico di un nodo. Inoltre, in caso di errori, i vari router non aggiornano tale informazione, perciò è facile la creazione di **loop**. Un esempio è quello in figura, in cui il link A-E non funziona e B per inviare il pacchetto ad E lo inoltra ad A. A si accorge del guasto e allora invia il pacchetto a B, iniziando così un loop.



Algoritmi adattativi

In base alla diversa tipologia di algoritmo verranno poi realizzati i protocolli. I principali algoritmi adattativi sono:

- Routing centralizzato
- Routing isolato
- Routing distribuito
 - Distance Vector
 - Link State

Routing centralizzato

Per eseguire routing centralizzato è necessario avere un nodo che diventa un **Routing Control Center** che, ricevendo le informazioni dalla rete, calcolerà per ogni dispositivo le forwarding table. Per eseguire il calcolo, tutti i nodi devono dare le informazioni sufficienti affinché i calcoli vengano eseguiti. Il nodo, effettuerà il calcolo applicando delle **ottimizzazioni** sul percorso migliore. Il nodo, avendo la visione globale della rete, può applicare delle tecniche molto più raffinate rispetto al routing distribuito. Un algoritmo centralizzato può inoltre risolvere i problemi di guasti in maniera più rapida rispetto a un algoritmo distribuito.

Di contro, un algoritmo centralizzato ha il grosso svantaggio di essere un **single point of failure**, portando a rischio l'intero processo decisionale. Inoltre, un algoritmo di questo tipo non è consigliabile per reti **altamente dinamiche**. Questo perché ogni volta che la topologia cambia, è necessario aggiornare l'RCC, che deve tenere conto di tutti i cambiamenti e ricalcolare **tutte** le forwarding table.

Routing isolato

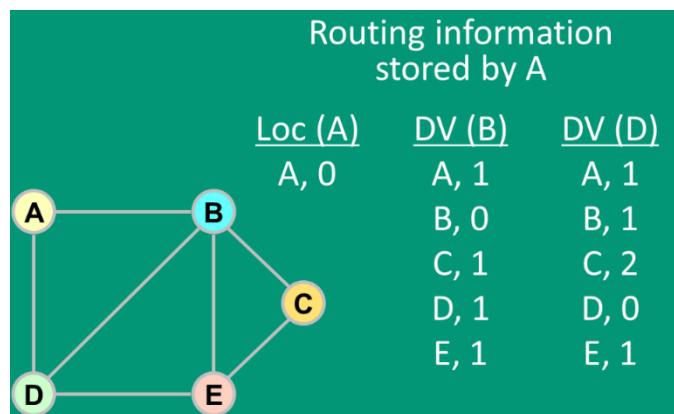
Come per l'algoritmo a flooding, anche quello isolato ha diversi svantaggi. Viene realizzato perché ogni nodo decide in maniera **indipendente, senza scambio di informazioni**. Con determinate caratteristiche (ad esempio una topologia “lineare”) ciò può essere vantaggioso.

Routing distribuito

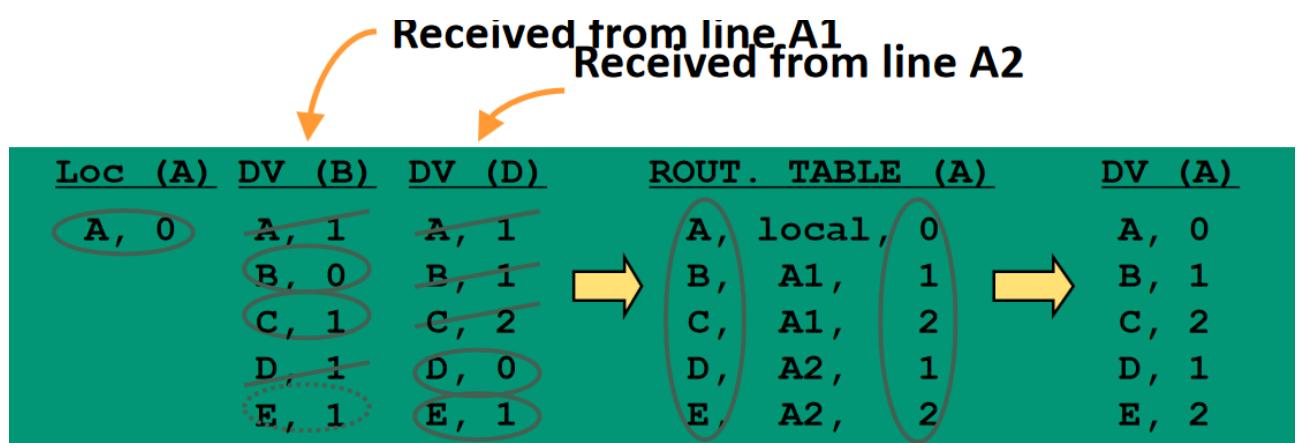
Il routing distribuito combina i vantaggi della decisione presa in autonomia nel routing isolato con lo scambio di informazioni. Con il routing distribuito, infatti, ogni router genera la propria forwarding table in autonomia ma comunicando con i router, evitando però i problemi del routing centralizzato.

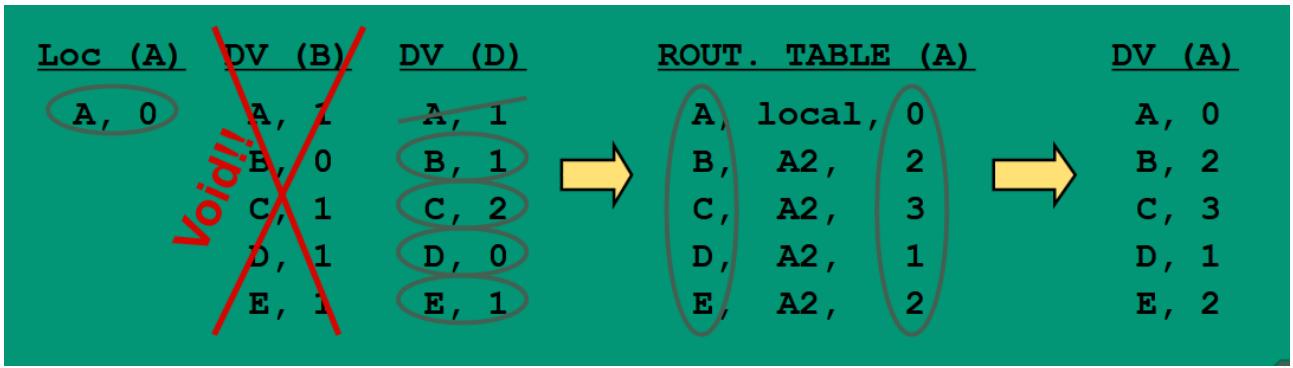
Algoritmo Distance Vector

Gli algoritmi distance vector si basano sul fatto che la lista delle destinazioni raggiungibili viene comunicata a tutti i nodi. Inoltre, viene comunicata la distanza che un singolo router deve impiegare per raggiungere un altro nodo e viene fatto da ogni router e la si manda su tutte le interfacce. In realtà, però, l'informazione rimane soltanto ai vicini. I vicini ricalcoleranno la loro forwarding table ricalcolando il tutto e inviando a loro volta le informazioni ai vicini. Si ha visione solo dei **neighbors** (next-hop). Il problema di questo algoritmo è il tempo che interessa affinché ogni router riesca a calcolare le proprie tabelle definitivamente.



Nello scenario a sinistra, A calcola le proprie forwarding table basandosi sulle informazioni provenienti dai nodi a lui vicini. Così facendo, A genera la propria forwarding table.





Se si ha un cambio di topologia, ad esempio con un guasto nel link A-B, la routing table di B non sarà più valido e A dovrà ricalcolare la propria vector table.

Cold Start

All'avvio dei router, ogni nodo possiede solo la conoscenza relativa al proprio nodo. Supponendo che sia A il primo nodo a inviare il Distance Vector, lo invierà ai propri vicini. Dunque, B e D si accorgeranno che A è raggiungibile con distanza 1 e dunque al prossimo step B e D propagheranno le informazioni ai vicini. A sua volta A aggiornerà le informazioni ma anche C ed E faranno altrettanto. Ciò viene fatto finché il tutto non converge.

<u>RT (A)</u>	<u>RT (B)</u>	<u>RT (C)</u>	<u>RT (D)</u>	<u>RT (E)</u>
A, loc, 0	B, loc, 0	C, loc, 0	D, loc, 0	E, loc, 0

A sends its DV

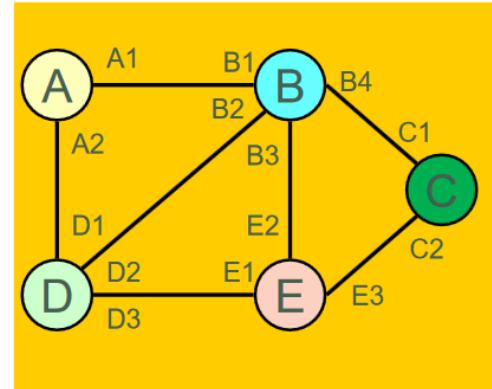
<u>RT (A)</u>	<u>RT (B)</u>	<u>RT (C)</u>	<u>RT (D)</u>	<u>RT (E)</u>
A, loc, 0	A, B1, 1 B, loc, 0	C, loc, 0	A, D1, 1 D, loc, 0	E, loc, 0

B and D send their DVs

<u>RT (A)</u>	<u>RT (B)</u>	<u>RT (C)</u>	<u>RT (D)</u>	<u>RT (E)</u>
A, loc, 0	A, B1, 1 B, loc, 0	A, C1, 2	A, D1, 1 B, D2, 1	A, E2, 2 B, E2, 1

All send their DVs

<u>RT (A)</u>	<u>RT (B)</u>	<u>RT (C)</u>	<u>RT (D)</u>	<u>RT (E)</u>
A, loc, 0	A, B1, 1 B, loc, 0	A, C1, 2 B, C1, 1	A, D1, 1 B, D2, 1 C, D2, 2	A, E2, 2 B, E2, 1 C, E3, 1
B, A1, 1			B, D2, 1	B, E2, 1
C, A1, 2	C, B4, 1	C, loc, 0	C, D2, 2	C, E3, 1
D, A2, 1	D, B2, 1	D, C2, 2	D, loc, 0	D, E1, 1
E, A2, 2	E, B3, 1	E, C2, 1	E, D3, 1	E, loc, 0



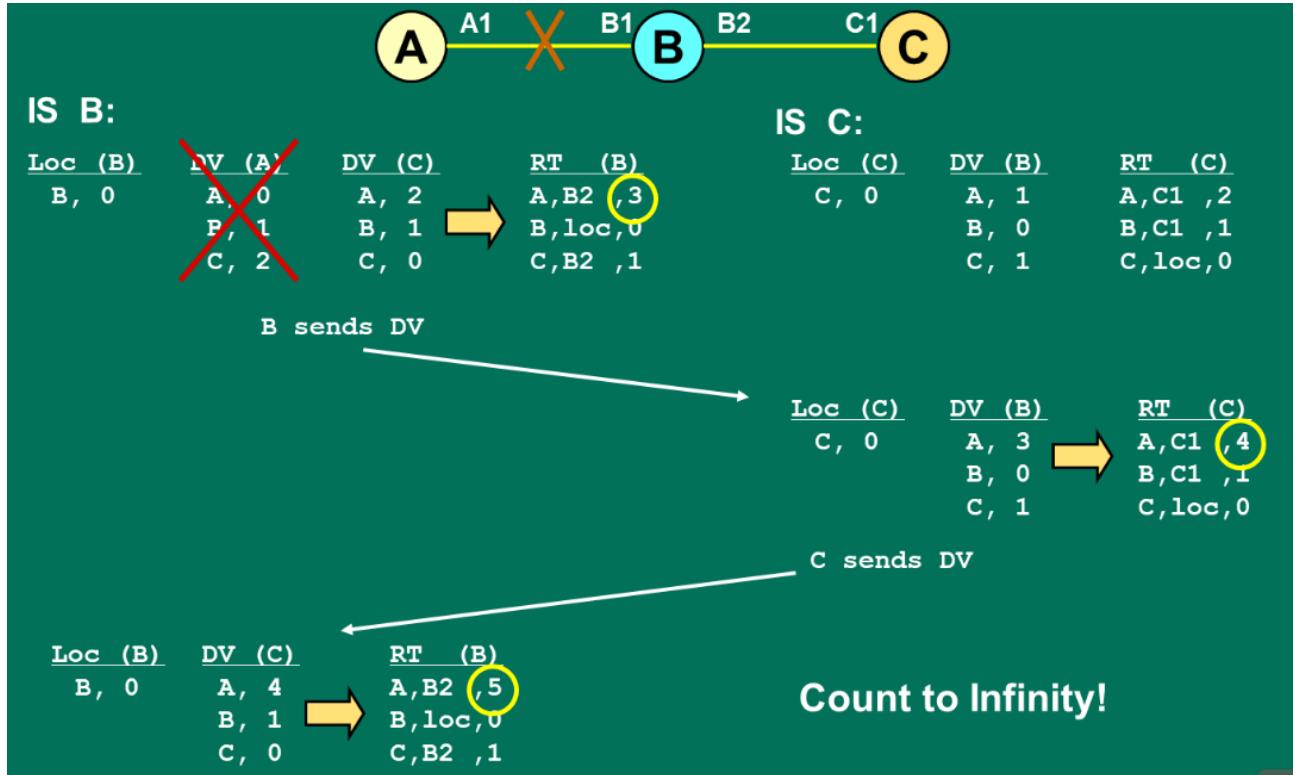
All'inizio, le informazioni non saranno presenti, perciò finché il transitorio non termina non è possibile inoltrare pacchetti.

Problemi dell'algoritmo

- **Black hole**
- **Count to infinity**
- **Bouncing Effect (loop)**

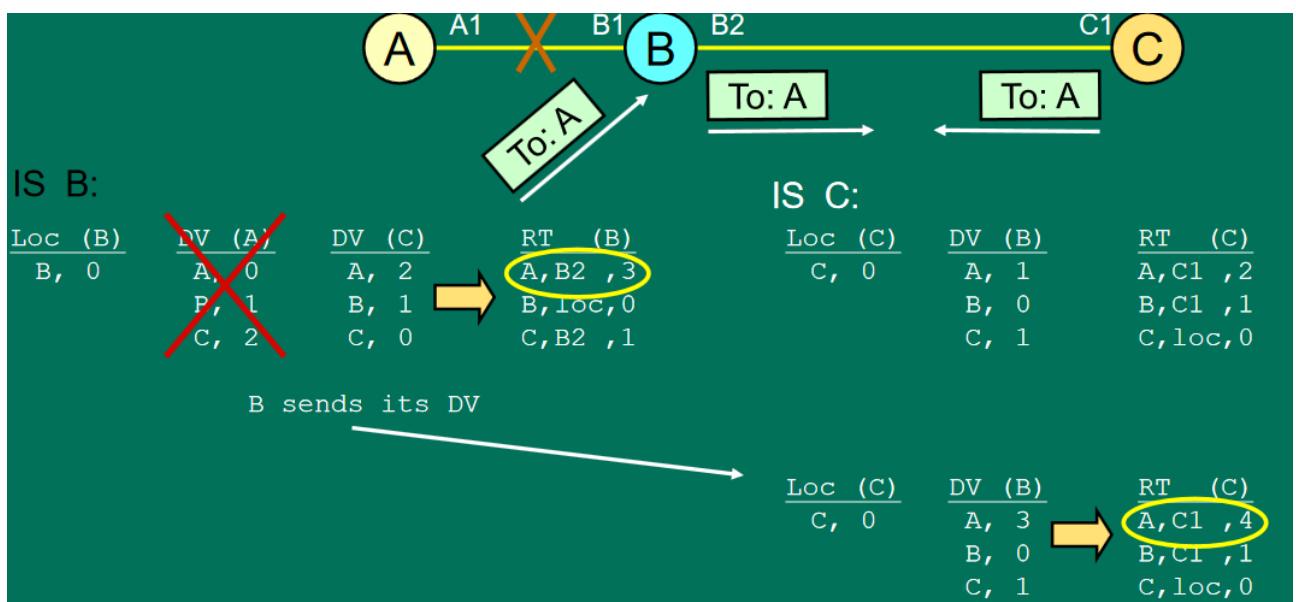
Count to infinity

Sebbene possano esserci topologie semplici, se avviene un guasto tra A e B accade quanto segue: B elimina la distance vector di A e riaggiornerà la propria aggiornando l'informazione relativa ad A. B però non conosce la topologia, perciò non è a conoscenza che A non è più raggiungibile, perciò B dirà che A lo può raggiungere attraverso C. A sua volta, però, C era a conoscenza che A era raggiungibile tramite B, ma non avendo l'informazione sul guasto aggiornerà la propria vector table dicendo che A è raggiungibile attraverso B. Tutto ciò viene effettuato senza fine, provocando un **count to infinity**.



Bouncing Effect

Nel caso in cui arrivasse un pacchetto a B destinato ad A, B lo inoltrerà a C. A sua volta C lo inoltrerà a B.



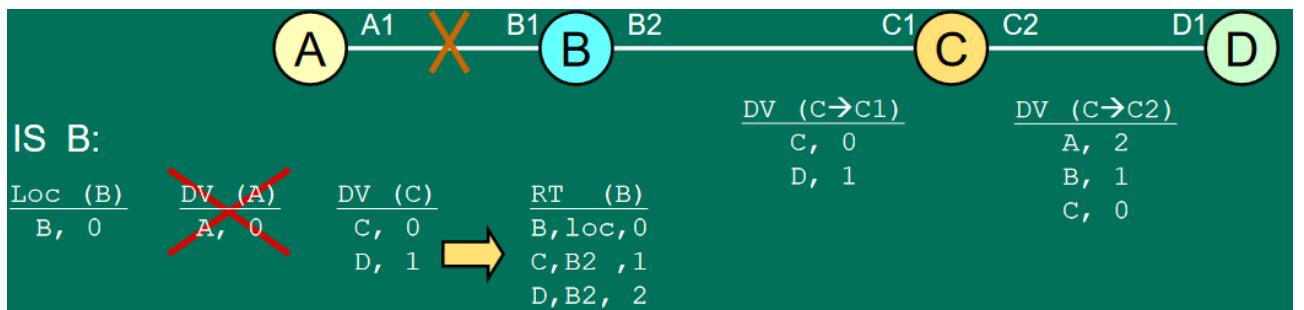
Questo provoca un rimbalzamento ed un aumento del traffico tra B e C.

Viste queste criticità e visto che il Distance Vector non conosce la topologia della rete, esistono delle soluzioni parziali al problema:

- Split Horizon
- Path Hold Down
- Route Poisoning

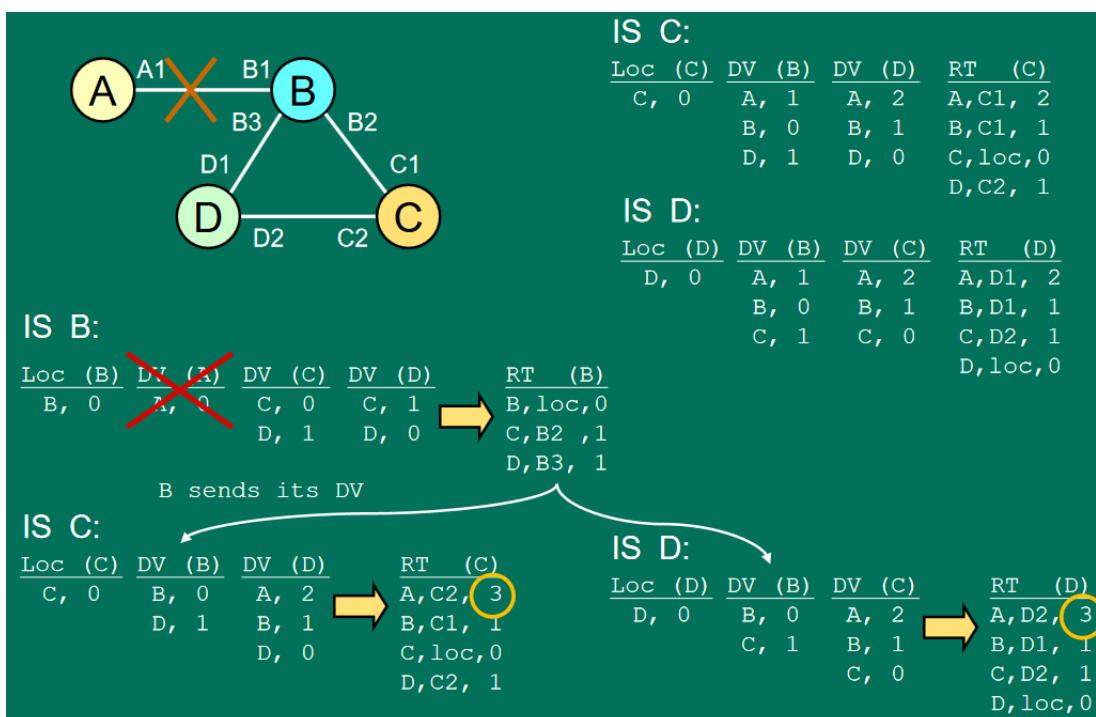
Split Horizon

La variante si basa sul principio: "se C raggiunge A attraverso B, è inutile per B provare a raggiungere A attraverso C". Questo serve per evitare la creazione di loop. Così facendo B non propaga sul distance vector l'informazione di A.



Lo split horizon previene i loop tra due nodi ed aumenta la velocità di convergenza. Questo perché vengono inviate informazioni parziali e si ha una **personalizzazione** del DV verso i vicini. In particolare, la DV di C verso B non contiene destinazioni raggiunte attraverso B. Nelle implementazioni reali, la route ha un **timeout**. In questo modo, se non si raggiungono aggiornamenti entro un tot di tempo, tale route non viene più ritenuta valida.

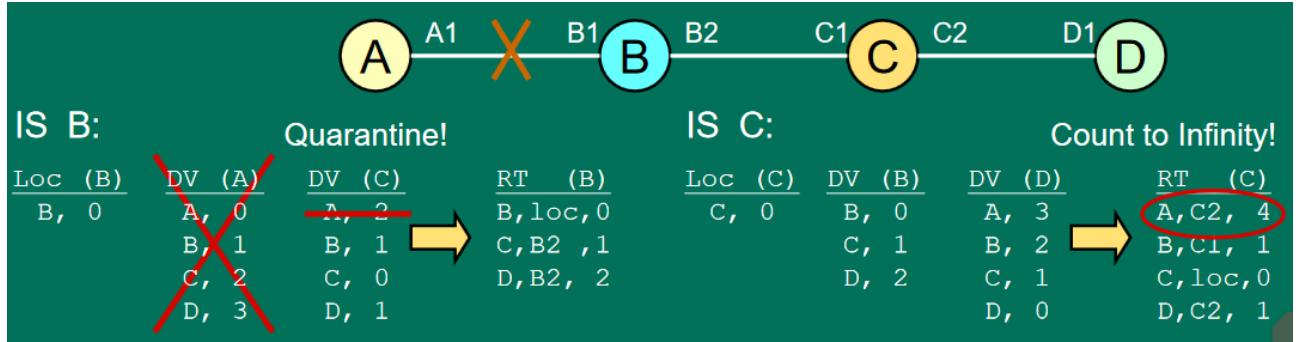
Su una rete completamente connessa, il problema del count to infinity potrebbe comunque verificarsi nel periodo transitorio perché C e D non sono a conoscenza di tutti i path che possono raggiungere A. Il problema di intasamento del link si supera soltanto quando C e D manderanno di nuovo il DV. Però tale DV viene inviata anche a B.



B però non dovrebbe avere informazioni su A, perciò la tabella di B viene “avvelenata”. In questi casi interviene la variante Path Hold Down.

Path Hold Down

“Se un link L fallisce, tutte le destinazioni raggiungibili attraverso il link L sono considerate irraggiungibili per un certo periodo di tempo”. Dunque, nessun percorso di A viene calcolato.

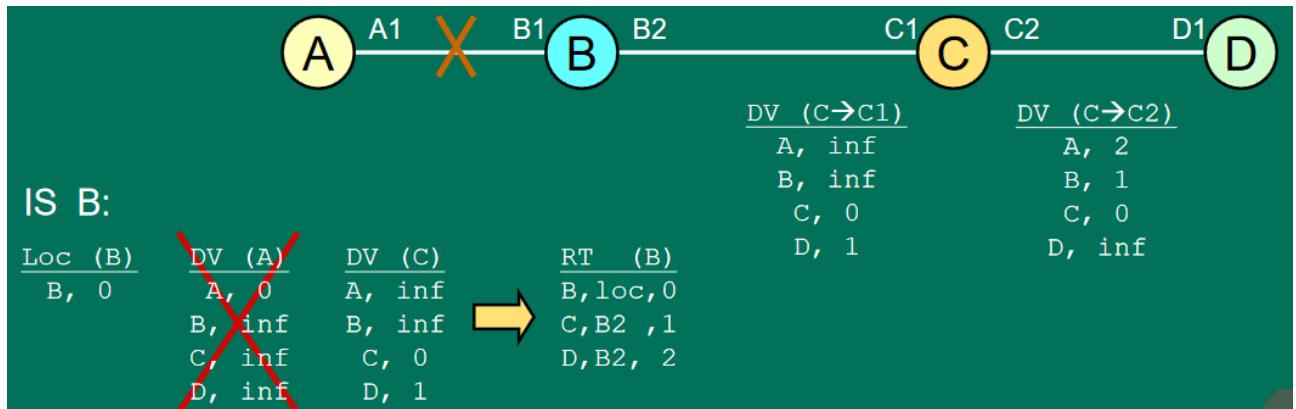


Il Path Hold Down soltanto, però, non risolve il problema del count to infinity, poiché D vedrà A raggiungibile tramite C e dunque lo scriverà nella DV. Utilizzandolo, invece, in accoppiata allo Split Horizon tale informazione non viene inviata e il problema del count to infinity viene risolto.

I loop possono verificarsi quando nelle route si verifica un aumento del costo. Un concetto potrebbe essere quello di non utilizzare l'aumento del costo nel DV (ma mostrarlo soltanto, senza inoltrarlo). Si può implementare con il Path Hold Down, ma ciò potrebbe comportare il blocco dei percorsi che in realtà sono validi.

Route Poisoning

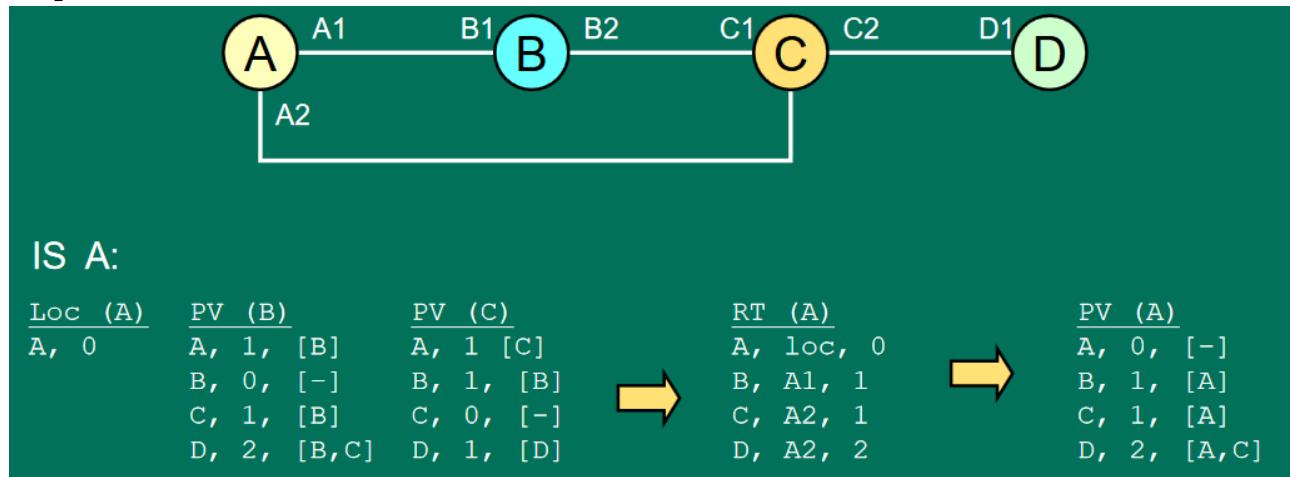
Si rende un determinato percorso a distanza **infinita**. Impostando direttamente il contatore a infinito si evita il problema del count to infinity. In altre parole, in caso di guasti si dice che un nodo non è più raggiungibile, invece di inserirlo in quarantena (come nel path hold down). Può essere dunque visto come un'alternativa al Path Hold Down. Non è più necessario utilizzare un timer.



Il Distance Vector in definitiva è facile da implementare e mettere in piedi, poiché le configurazioni richieste sono molto poche. La complessità di convergenza varia da $O(n^2)$ a $O(n^3)$, perciò conviene sconsigliato per reti di grande dimensione.

Path Vector

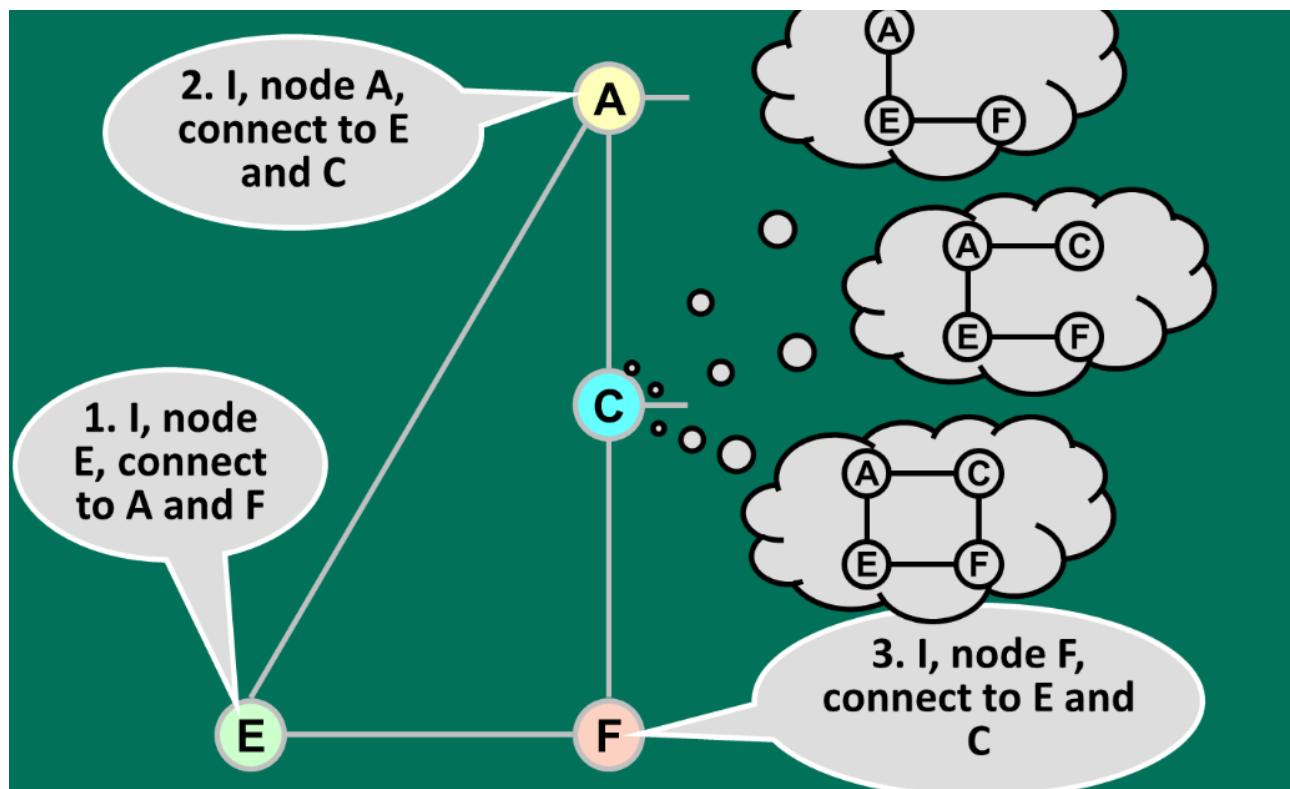
Algoritmo che **elimina il problema dei loop**. Molto familiare al Distance Vector, ma con delle varianti. Alla DV viene sostituita la tabella di Path Vector, dove viene inserito il **costo** di un link ma si tiene traccia **anche del percorso**.



Il calcolo del percorso è però oneroso perché man mano che i vicini comunicano quali nodi riescono a raggiungere direttamente il router deve calcolare i percorsi e svolgere dei calcoli più complessi.

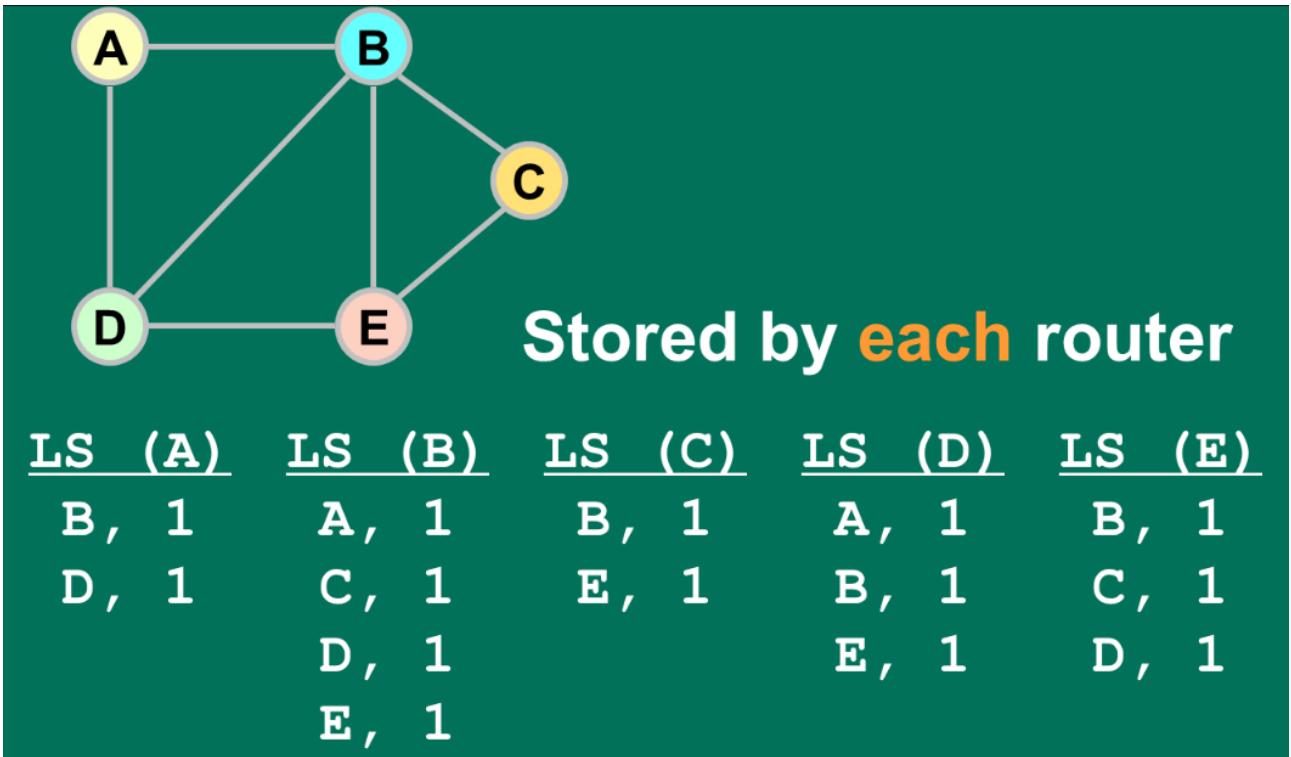
Algoritmo Link State

Un singolo nodo riesce a calcolare dinamicamente la topologia sapendo negli altri nodi gli elementi a cui sono connessi.



L'informazione sullo stato dei link viene diffusa con l'utilizzo di una **mappa locale** e l'informazione non viene inviata soltanto ai vicini ma **su tutti i nodi**, in modo da permettergli di avere una visione globale della rete.

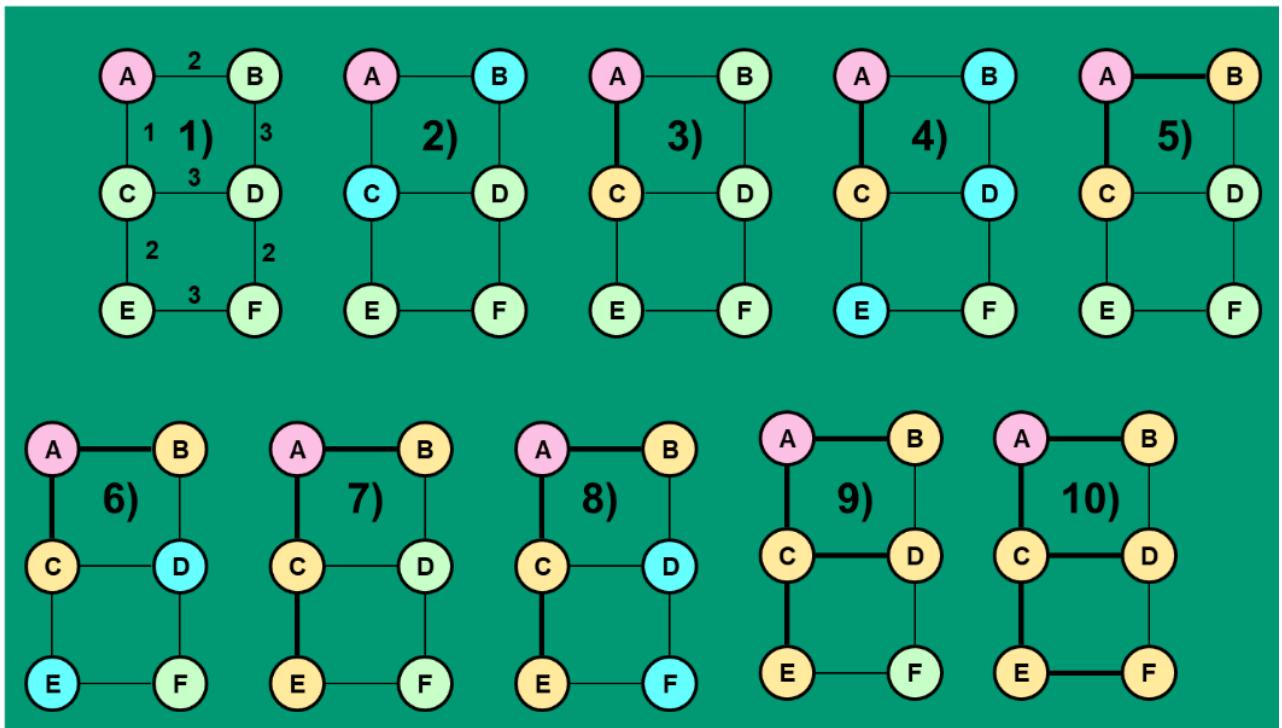
Ogni nodo andrà dunque a salvare il link state di tutti gli altri nodi.



Algoritmo di Dijkstra

L'algoritmo di Djikstra ha una complessità $L \cdot \log (N)$ dove L è il numero di link ed N è il numero di nodi e si basa sul concetto di **Shortest Path First**.

(root) (TEMP) (PATH)



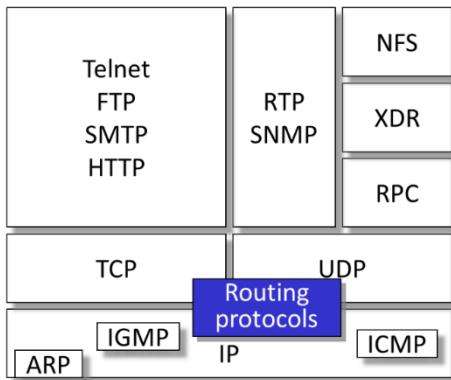
Nell'esempio in foto viene scelto un nodo **root** (A). Il nodo, a sua volta fa sì che si possano associare come nodi **temporanei** i vicini rispetto ad A. Dei nodi temporanei si sceglie quello a distanza minima e lo si etichetta come nodo **path**. Dal nuovo path generato si associeranno i nodi temporanei vicini (B,D,E) e si ripete il procedimento.

Le caratteristiche sono:

- Convergenza rapida
- I link state sono “piccoli”, cioè viene memorizzata solo l’informazione sul costo
- Raramente genera loop
- Facile capire e risolvere i guasti
- L’implementazione è complessa (richiede un flooding selettivo)

Un link state viene generato quando vi sono cambiamenti di topologia oppure tramite **timer periodici** (nel caso in cui si verifichino guasti di cui non ci si accorge).

Internet Routing Architecture



I vari protocolli di routing sono nella fascia tra il livello 3 e il livello 4. In particolare, si definisce **protocollo per lo scambio** di informazioni nella rete, il protocollo che determina la miglior route per ogni destinazione (basandosi sugli algoritmi di routing).

Un protocollo deve: **definire delle metriche**, definire la **codifica dei pacchetti**, specificare i **timer** (se l'algoritmo li richiede) e definire dei parametri.

Si definisce **dominio di routing** un set di router che si basano sullo stesso protocollo di routing, interconnesse in una porzione di rete. Inoltre, esiste anche il concetto di **redistribution**, ovvero la possibilità per un router di appartenere a più domini di routing, al fine di distribuire le informazioni tra un dominio ed un altro.

Le regole di redistribuzione sono definite dall'amministratore: gli annunci possono essere filtrati, viene definita una metrica di conversione ed anche la priorità. Tutto ciò viene fatto perché nella realtà i sistemi di rete si basano sul concetto di **Autonomous System**: un insieme di sottoreti raggruppato in base a delle topologie e a dei criteri di organizzazione. Ovvero, negli AS verranno poi definiti i domini di routing. Un esempio di AS sono i provider che permettono l'accesso a internet ai clienti.

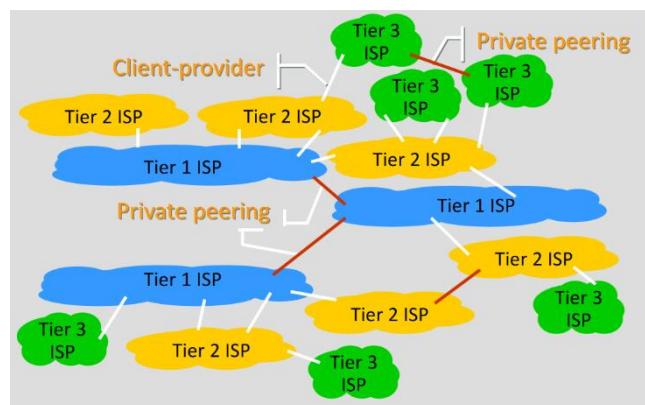
Tramite questo sistema è possibile avere un miglior coordinamento e riuscire meglio a partizionare la rete. Un AS viene identificato dallo IANA attraverso un **numero di due byte**. Una parte di questi byte sarà privata.

All'interno di un AS saranno presenti dei dispositivi detti **border gateway** posizionati al confine tra un AS e un altro. Tali dispositivi posti ai bordi della rete dovranno comunicare tra loro, ma anche con i dispositivi interni alla rete. Un possibile protocollo per la comunicazione tra i dispositivi di bordo e l'interno della rete può essere **iBGP**, mentre un protocollo per la comunicazione tra Border gateway di AS diversi è **eBGP**.

Si definisce **Exterior Routing** il routing tra AS. Non deve essere necessariamente creato seguendo lo shortest path, poiché la scelta viene effettuata basandosi su delle policy.

In figura è presente l'interconnessione tra ISP, ma non è detto che debbano essere per forza ISP (possono essere anche aziende che comunicano tra loro). Si definisce **private peering** una connessione tra entità paritetiche.

A queste reti viene inserito un elemento detto **NAP/IXP** (**Neutral Access Point/Internet eXchange Point**), entità terze che mettono a disposizione degli spazi (infrastrutture) dove è possibile che ISP diversi creino delle comunicazioni 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.



Protocolli di routing

Si distinguono due famiglie di protocolli:

- **Interior Gateway Protocol (IGP)** intra-domain routing: protocolli di routing all'interno dello stesso dominio di routing
- **Exterior Gateway Protocol (EGP)** inter-domain routing: protocolli di routing tra diversi domini (e quindi diversi AS)

La famiglia relativa ai protocolli IGP ha l'obiettivo di distribuire le informazioni topologiche (attraverso i vari approcci visti precedentemente). Si va a scegliere la route in base alla “best” route (i percorsi vengono scelti in base a decisioni topologiche o di rete).

Un protocollo EGP ha l'obiettivo di **distribuire informazioni tra AS** (ad esempio il concetto dei costi) e sceglie i percorsi in base a determinate politiche.

IGP – Distance Vector

- **RIP**: Routing Information Protocol
- **IGRP**: Interior Gateway Routing Protocol
 - **E-IGRP**: Enhanced IGRP

IGP – Link State

- **OSPF**: Open Shortest Path First
- **Integrated IS-IS**

EGP

- **BGP**: Border Gateway Protocol
- **IDRP**: Inter Domain Routing Protocol
- **Routing statico** può essere un'opzione valida (guardando la figura precedente, un Tier 3 ISP che ha accesso solo alla Tier 2 ISP, è possibile utilizzare un routing statico per indirizzare tutto il traffico a tale rete)

RIP

Il protocollo RIP è un protocollo di tipo IGP – Distance Vector. Caratteristiche di questo protocollo sono:

- Conteggio degli hop (distanza tra due hop)
- Al più sono supportati 15 hop tra due nodi
- Messaggi di aggiornamento periodici
 - Si aggiornano le distance vector
 - Ogni 30 s
 - Tramite un timeout
- La convergenza è di 3 minuti

IGRP

Protocollo Cisco proprietario nato per superare diverse limitazioni del protocollo RIP. Non è mai diventato uno standard ma è sempre stato un'alternativa a RIP.

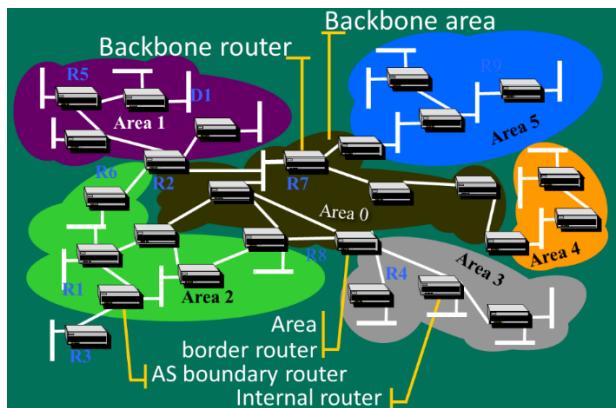
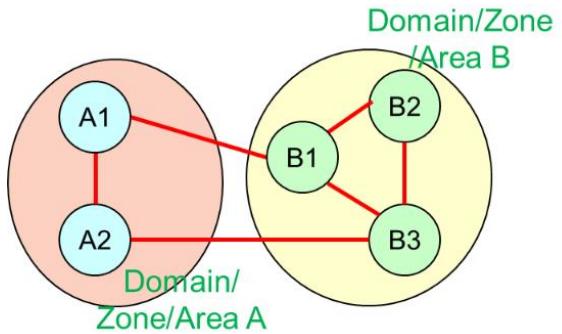
OSPF

Il protocollo OSPF è un protocollo IGP link state di **tipo gerarchico**. I routing domain vengono divisi in aree e successivamente vengono effettuate le aggregazioni delle informazioni delle varie aree.

Un algoritmo gerarchico implica che un router conosce i router raggiungibili all'interno di un dominio. Ciò che invece viene raggiunto all'esterno non è noto poiché mascherato. Ad esempio, il nodo B1 potrebbe mostrare l'indirizzo di una intera sottorete e mostrare che può raggiungere l'intera rete B.

Quando una destinazione non è nel proprio dominio, bisogna inoltrare il pacchetto a un edge router affinché esso lo consegna a destinazione.

Nel routing gerarchico loose (può essere *strict* se non si ha visibilità oppure *loose* se si vuole dare visibilità massima a un certo dominio) si hanno problemi di scalabilità. Nel caso di routing gerarchi loose, non si rappresenta più un dominio con un indirizzo di rete, ma ogni router avrà un indirizzo.

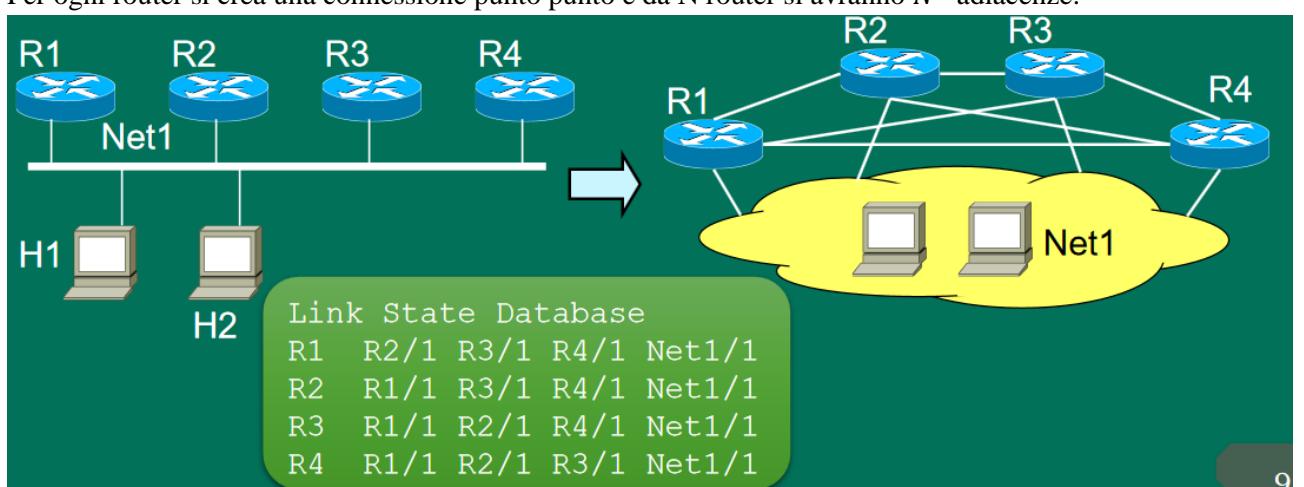


L'architettura OSPF prevede una topologia di rete divisa in aree (AS) e delle **backbone area** che sono da interconnessione tra AS, seguito da dei **backbone router** che permettono di dialogare tra aree diverse. Saranno presenti anche **border area** (che posseggono sorgente o destinazione appartenente a quell'area). Infine, l'**AS boundary router** è il router di confine in tale router al bordo. Infine, i **router interni** si interessano solo dell'interconnessione tra elementi di un'area.

OSPF utilizza un routing gerarchico *strict*, fornendo dunque una scalabilità massima.

Broadcast Network

Per ogni router si crea una connessione punto punto e da N router si avranno N^2 adiacenze.



Si ottiene dunque una rete di tipo **stella** dove ogni singolo nodo diventa raggiungibile per il sistema.

Integrated IS-IS

Estensione del protocollo OSI, utilizzando comunque il concetto di routing gerarchico. È pensato per reti di grandi dimensioni e per far dialogare gli ISP.

BGP

Si basa sul concetto di Path Vector. In particolar modo all'interno di un AS vengono rilevate le destinazioni raggiungibili. Possiede determinate policy per calcolare la route. BGP viaggia sopra **TCP** perciò ha ottime caratteristiche di **affidabilità**. La quantità di traffico scambiata è ridotta poiché lo scambio avviene solo dopo un cambiamento. Avviene soltanto su esplicita configurazione dei vicini tramite una sessione di peering.

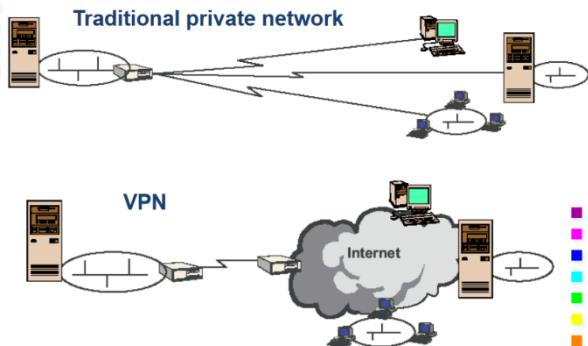
IDRP

Evoluzione di BGP per le reti ISO/OSI. Viene portato su una connessione TCP/IP e supporta **IPv6**. Non è molto usato, si continua a preferire BGP.

VPN

Una Virtual Private Network è una connessione su un'infrastruttura condivisa al fine di forzare delle policy in una rete privata. Una VPN si rende necessaria perché parlando di infrastruttura pubblica si parla di una infrastruttura dove l'utente non può avere una chiara visibilità e non può garantire determinate proprietà. L'infrastruttura pubblica può essere quella di un generico ISP.

Quando si parla di **policy** non ci si sofferma solo sull'aspetto della sicurezza, ma anche garantire **Quality of Service**, affidabilità, indirizzamento.



Nel caso in cui si volesse mantenere le stesse policy, la soluzione da utilizzare sarebbe quella di utilizzare dei link appropriati tra (ad esempio) le case dei dipendenti e la sede.

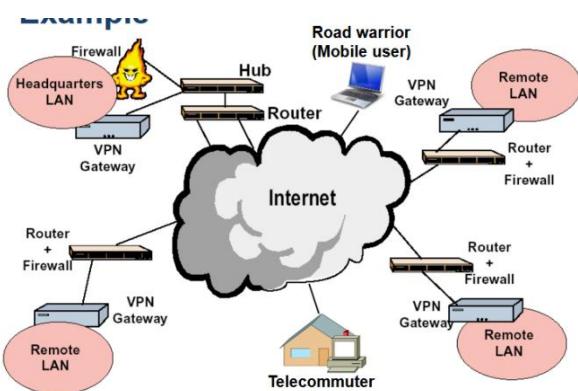
In alternativa, volendo realizzare una VPN, è chiaro che appena si utilizza una struttura pubblica è necessario creare dei **tunnel sicuri** che generano la VPN.

Gli elementi chiave sono i seguenti:

- **Tunnel**: si indica un encapsulamento/cifratura sicura del traffico della corporate network che deve transitare sulla rete condivisa. Attraverso tale encapsulamento è possibile garantire le determinate proprietà.
- **VPN Gateway**: dispositivo di terminazione che genera il traffico della corporate network. Può essere associato a un tunnel endpoint.

Motivazioni

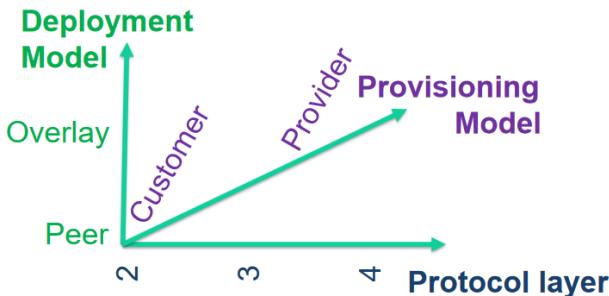
- Le VPN sono in grado di tagliare i costi rispetto a soluzioni di connettività costose (link dedicati)
- Le VPN sono in grado di creare flessibilità da un punto di vista della gestione degli accessi
 - È possibile garantire limitazioni in base a quale traffico verrà abilitato dal firewall
 - Garantisce limitazioni agli accessi anche nei confronti di internet



Sulla sinistra è presente il dipartimento (sede generale), dall'altro lato invece gli utenti dello stesso dipartimento che però sono fuori dalla sede aziendale. Sono presenti anche altre sedi della stessa azienda.

Bisogna dunque fare in modo che una singola rete di una sede veda le altre come se fossero un'unica rete.

Le tipologie di accessi vengono gestite dai VPN Gateway mentre la gestione degli accessi viene gestita dal firewall.

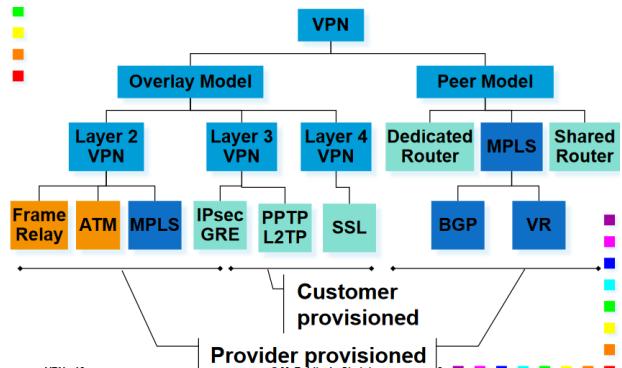


Osservando il grafico in 3D è possibile capire che è possibile avere VPN al livello 2,3,4. È possibile inoltre caratterizzare le VPN in base al **deployment model** ed anche in base al **provisioning model** (cioè se la VPN viene fatta dall'utente in maniera trasparente al provider, oppure se è il provider a fornire il servizio).

Appiattendo la rappresentazione tridimensionale, è possibile dare una rappresentazione ad albero come in figura. In particolar modo le VPN possono essere considerate come **overlay model** o **peer model**. In base all'overlay model è possibile avere implementazioni al livello 2-3-4. In base al livello vi sono diverse implementazioni.

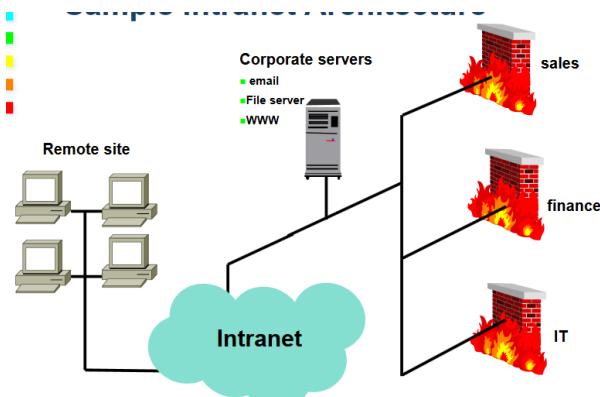
È possibile avere 3 tipologie diverse di VPN:

- **Access VPN** o **remote VPN** che permette di connettere un host ad una rete remota, virtualizzando un accesso di connessione
- **Site-to-site VPN**: connetto due network remote, anche detto virtualized line
- **End-to-end VPN**: connessione remota tra due host



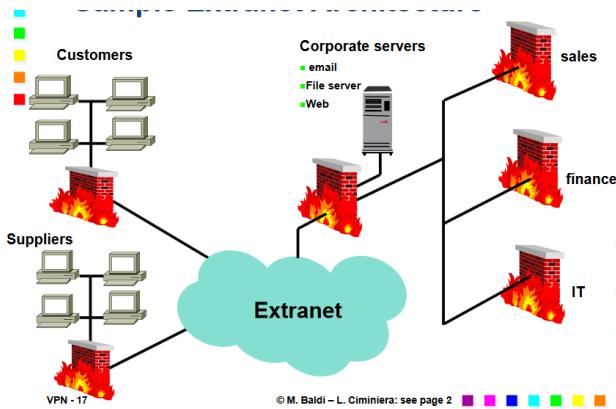
Gli scenari di deployment sono invece i seguenti:

- **Intranet VPN**
 - Sono presenti diversi dipartimenti/sedi dell'azienda, entro cui si vuole applicare le stesse policy e far sì che siano applicabili in tutte le sedi distaccate
- **Extranet VPN**
 - Interconnessione di diverse entità su una infrastruttura condivisa.
 - L'azienda potrebbe avere diverse politiche di accesso/sicurezza in base alle entità.



Sulla sinistra è presente una intranet (remote site) mentre dall'altro lato è presente la sede principale, che possiede dei server (corporate server) che devono essere raggiunti da tutti e dei dipartimenti, che a loro volta posseggono regole di accesso diverse.

Una volta instaurata la VPN tra le due sedi, qualsiasi elemento della VPN vede le risorse come se fosse all'interno della rete stessa.



In questo caso si vuole realizzare una **extranet**, realizzando comunque delle VPN ma inserendo un firewall all'ingresso della rete detto **border firewall**, per poter filtrare gli accessi (differenziare le risorse tra customers e suppliers). Questo viene fatto in modo da dare a ogni VPN delle policy diverse (VPN1 per customers, VPN2 per suppliers).

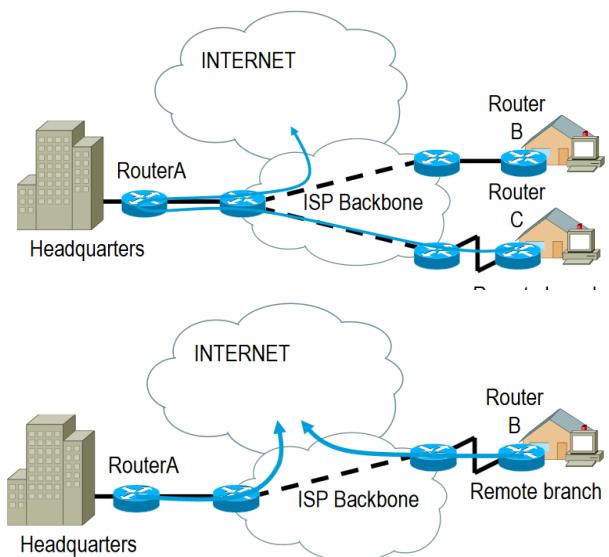
Problemi di Extranet

- Accesso ristretto alle risorse della rete da reti interconnesse
 - Firewall alla VPN
- Overlapping dell'indirizzamento
 - Network Address Translation
- Problemi di standard utilizzati
 - Abilita l'*interoperabilità* tra organizzazioni diverse
- Controllo del traffico
 - Evita che il traffico dei partner comprometti le performance della rete centrale

Accesso a internet

L'accesso a internet tramite una VPN può essere:

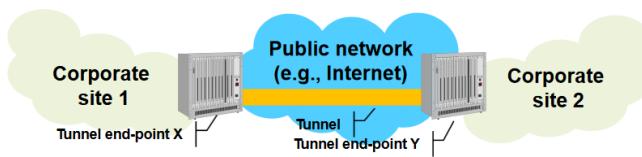
- **Centralizzato:** gli utenti remoti attraverso l'uso di un IP pubblico della sede possono accedere a Internet. La prima foto sulla destra ne rappresenta un esempio: il traffico dell'utente viene prima indirizzato verso gli headquarters e successivamente inoltrato a Internet.
- **Distribuito** (connessione volontaria): in questo caso è l'utente stesso, e successivamente il tipo di gateway, che in base alla tipologia di traffico decide se connettere in maniera sicura l'utente all'azienda, oppure indirizzarlo a Internet.



Caratteristiche di una VPN

- **Separazione dei dati:** in base al traffico che arriva sul VPN gateway si può decidere dove forwardarlo
 - Tunneling
- **Incremento della protezione**
 - Encryption
- **Prevenire il tampering** (contraffazione)
 - Integrità
- **Identificare la sorgente**
 - Autenticazione

Site 2 site VPN Tunneling



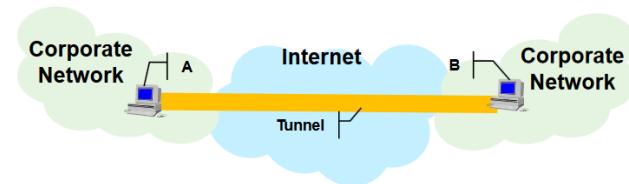
incapsulato in un tunnel).

I singoli gateway al bordo della rete creano un tunnel per l'infrastruttura pubblica. Il traffico che arriva da un gateway a un altro sarà traffico in tunnel con la VPN, mentre il traffico tra un host di una delle due reti e il gateway sarà in chiaro (e non incapsulato in un tunnel).

Con questo metodo è possibile generare diverse tipologie di tunnel a seconda del tipo di traffico. Inoltre, è possibile raggruppare i tipi di comunicazione. Perciò, si può inserire nel tunnel un **insieme di traffico**. Il VPN Gateway è un elemento “fidato” poiché è lui che andrà a decifrare i messaggi ricevuti. Efficace se la corporate network è **sicura** (dunque no sniffing, manomissione, etc..).

La VPN può indirizzare il traffico proveniente da più host anche con un solo tunnel. Maggiore aggregazione.

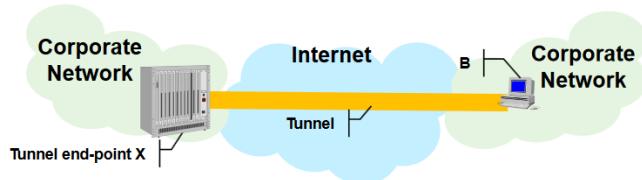
End 2 end VPN Tunneling



Il tunnel viene generato direttamente dall'host, non considerando i nodi intermedi. Il tunnel sarà dunque diretto dall'host di partenza fino all'host destinazione di due network diverse.

È necessario creare un tunnel per ogni host, anche se la sicurezza è garantita fino all'host, il quale si occuperà di decifrare il messaggio.

Remote VPN Tunneling



In questo caso la connessione con un tunnel direttamente da un host, ma termina al VPN gateway, per poi proseguire in chiaro all'interno della corporate network di destinazione.

Anche nel caso del Remote VPN sono presenti gli stessi problemi del caso end2end e site2site, poiché per ogni singolo host andrà aperto un canale specifico (end2end) e la corporate network deve essere sicura (site2site, altrimenti sniffing, manomissione).

Overlay Model

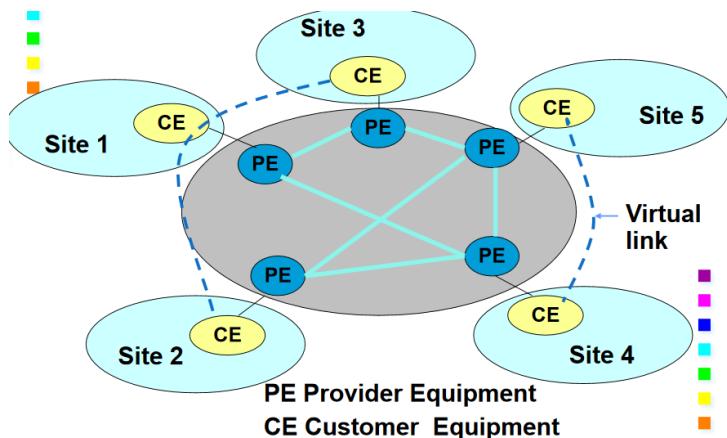
L'infrastruttura pubblica dove passa la comunicazione **non** partecipa alla realizzazione della VPN. Ciò vuol dire che a livello infrastrutturale, la rete pubblica non è conoscenza del fatto che su di essa possa passare una VPN (in realtà gli ISP fanno analisi del traffico quindi ne sono a conoscenza). Ogni VPN gateway deve essere in contatto con gli altri VPN gateway e deve occuparsi di aggiungere l'header corretto ai pacchetti.

Peer Model

Ogni VPN gateway interagisce con un router pubblico (è un peer) per **scambiare informazioni di routing** e andando a creare una connessione tra gateway di una stessa VPN lungo la rete pubblica.

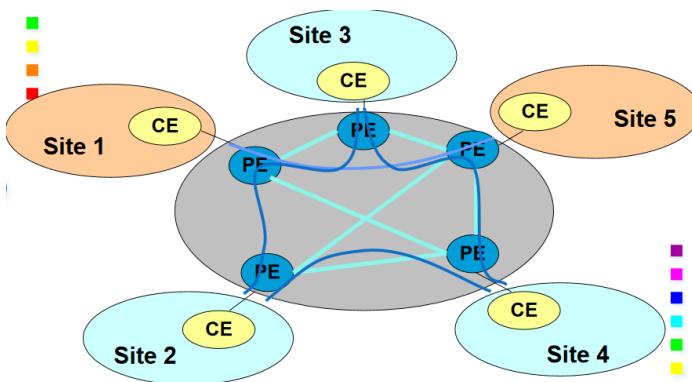
Customer Provisioned VPN

In questo caso l'utente ha l'onere di acquistare, configurare e gestire i dispositivi che implementano le funzionalità VPN. Il provider della rete non sarà al corrente che il traffico generato dall'utente sia VPN. Tutte le funzionalità sono implementate in questo dispositivo VPN dell'utente. Il tunneling terminal col CE (Customer Equipment). Questa soluzione permette la creazione di link virtuali (in figura tratteggiati).



Provider Provisioned VPN

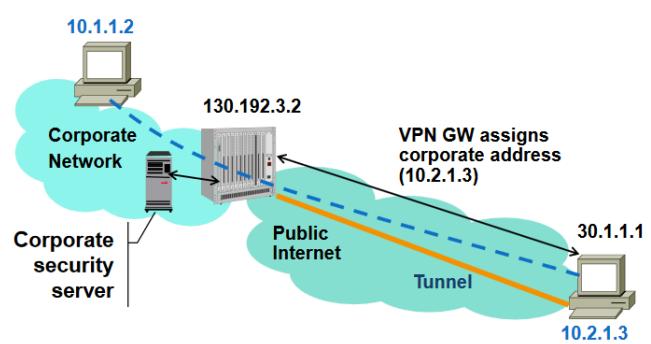
In questo caso il terminatore del traffico VPN sarà il PE (Provider Equipment). Perciò è necessario concentrarsi su cosa succede nel link tra il PE e il CE. Il traffico lungo tale link potrebbe potenzialmente risultare vulnerabile. Si sposta la configurazione al provider e non più all'utente.



Tutte queste soluzioni hanno pro e contro e nessuna di queste è migliore in assoluto.

Access VPN Customer Provisioned

Nell'esempio ci si trova in una VPN remota a livello di customer provider. L'endpoint avrà un indirizzo pubblico e uno privato. Il tunnel viene creato al livello di indirizzo pubblico dell'host e lo si va a creare tra l'host e il VPN Gateway, il quale a sua volta esporrà un indirizzo pubblico. L'indirizzo privato del router destinazione rimane nello stesso range di quello di partenza.



10.2.1.3 indirizzo privato

30.1.1.1 indirizzo pubblico (per mandare il pacchetto)

Access VPN Provider Provisioned

In questo caso l'utente domestico deve collegarsi al dispositivo NAS del provider, che sa che il traffico dell'utente remoto deve andare verso la specifica corporate network. Si forwarda il traffico al NAS (magari anche tramite autenticazione, opzionale) e successivamente viene instradato al Corporate Gateway, il quale a sua volta deciderà dove effettuare il forward.

Differenza tra Customer Provisioned vs Provider Provisioned (Remote VPN)

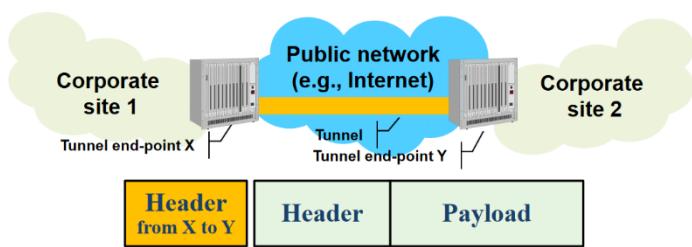
- **Customer Provisioned**

- Il remote host possiede 2 indirizzi
- Il remote host apre/chiude il tunnel
- Può essere utilizzato in qualsiasi connessione ISP

- **Provider Provisioned**

- Il remote host ha 1 indirizzo
- Il NAS termina il tunnel VPN
- Il remote host è sempre su VPN (e di conseguenza l'accesso a internet è centralizzato)
- Richiede accesso a specifici ISP

Tunneling



Quando si parla di tunnel, bisogna considerare che un pacchetto/frame tra sottoreti private che deve passare su una infrastruttura pubblica deve aggiungere un altro header al pacchetto.

In sostanza, si aggiunge un header contenente le nuove sorgenti e destinazioni cui il pacchetto deve arrivare.

Il tunnel porta anche alla creazione di **topologie VPN** di diverso tipo, che possono essere **Hub** o **Mesh**.

- **Hub**

- Si crea una stella: il nodo centrale (headquarter) si occuperà di creare dei tunnel a gruppi tra le varie sottoreti remote
- Il routing è sub-ottimo
- Buono per piccoli numeri di tunnel
- L'hub potrebbe diventare il collo di bottiglia

- **Mesh**

- Alto numero di tunnel (ma difficili da configurare manualmente)
- Routing ottimizzato

Layer 2 VPN

- **Virtual Private LAN Service**

- Emula le funzionalità di una vera e propria LAN. Viene utilizzata per connettere elementi distinti di LAN che passano su una infrastruttura pubblica.
- Il VPN Gateway simula il concetto del **bridge**, perciò l'operazione di routing viene fatta col **MAC address**.

- **Virtual Private Wire Service**

- Emula il concetto di **link**
- Può trasportare qualsiasi protocollo
- Si va a creare un incapsulamento di livello successivo (invece di utilizzare lo stesso livello)

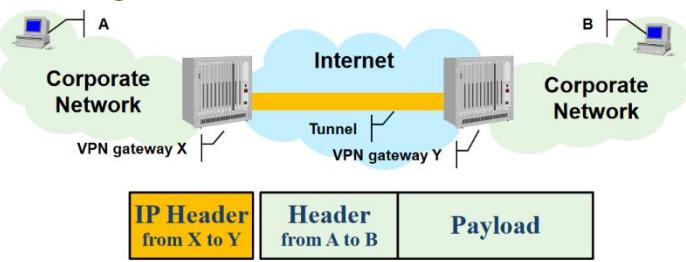
- **IP-Only LAN-like Service**

- L'equipment del customer è un router IP (non switch Ethernet)
- Dunque, si parte a incapsulare il payload IP. Si utilizzeranno i protocolli ICMP e ARP per trasportare tali informazioni.

Layer 3 VPN

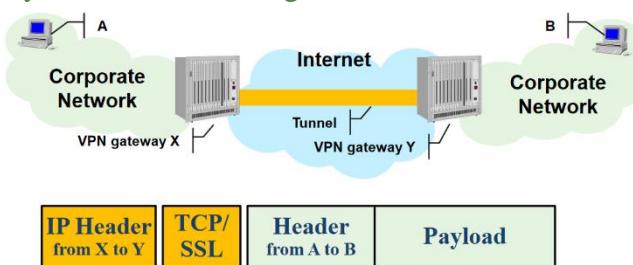
I dispositivi in questo caso sono gli IP router e gli host IP. I pacchetti del livello 3 vengono forwardati lungo una rete pubblica. Il pacchetto IP viene incapsulato al livello 3, dunque con un altro header IP (**GRE**, **IPsec**) oppure può essere fatto al livello 2 con un pacchetto IP (**L2TP**, **PPTP**).

Tunneling IP in IP



A e B sono due indirizzi di livello IP non necessariamente pubblici (perché a VPN creata loro saranno come nella stessa rete). A inoltrerà il traffico al gateway, il quale attiverà il tunnel e all'header IP di A viene aggiunto un ulteriore header IP contenente come destinazione il gateway di B.

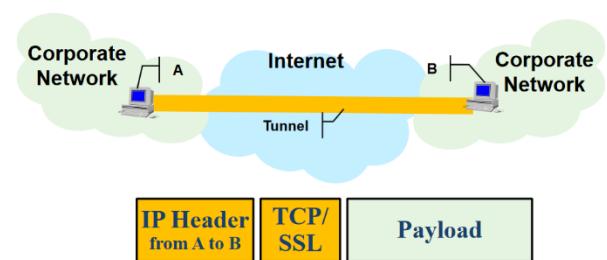
Layer 4 VPN Tunneling: s2s

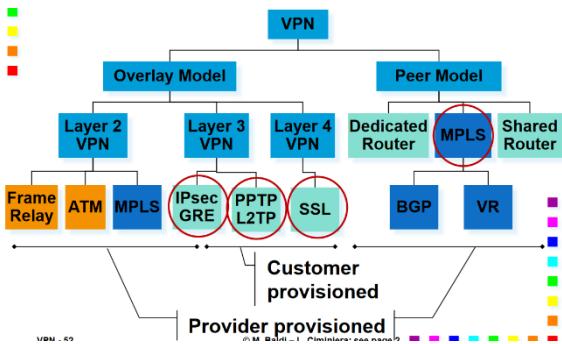


Si instaura una sessione TCP ma piuttosto che creare una connessione TCP col nodo destinazione, si sceglie di aggiungere un livello protocollare al livello 4 creando una connessione sicura (SSL/TLS) tra i gateway. Una volta creata la sessione sicura, si viene inserito nuovamente l'header di livello 3, per fare in modo che il pacchetto venga consegnato correttamente. (praticamente IP -> TCP/SSL -> IP)

Layer 4 VPN Tunneling: e2e

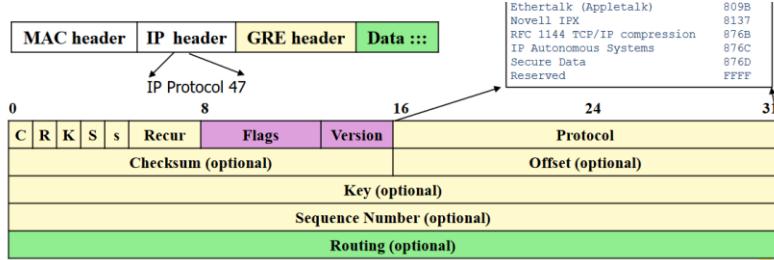
Il canale sicuro viene realizzato nella sessione sicura fra host, a cui viene aggiunto l'header del pacchetto IP in modo che i vari router sappiano chi è la destinazione finale. La differenza col caso precedente è che non c'è l'IP header incapsulato nel pacchetto. In questo caso l'header IP non è protetto (TCP/SSL può esserlo optionalmente ma in generale non lo è).





Nella foto a sinistra, il riepilogo dei protocolli suddivisi per tipo di provisioning.

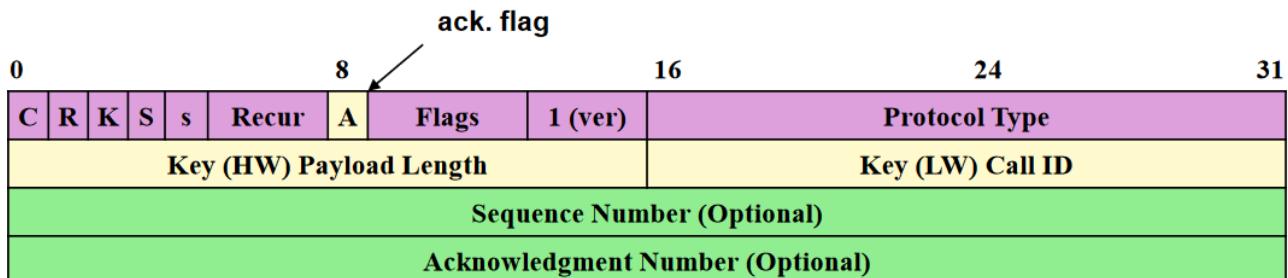
GRE – Generic Routing Encapsulation



infatti nell'header GRE è presente il campo **protocol** (nessun protocollo supera il livello 3 IP).

- C,R,K,S sono flag che indicano la presenza di campi opzionali (o meno)
- Il flag “s” serve per specificare il tipo di routing (campi Key, Seq No, Routing opzionali in foto)
- Recur serve per sapere se si sta effettuando un incapsulamento dentro l'incapsulamento

Successivamente l'header è stato migliorato (version 1), aggiungendo ulteriori campi:



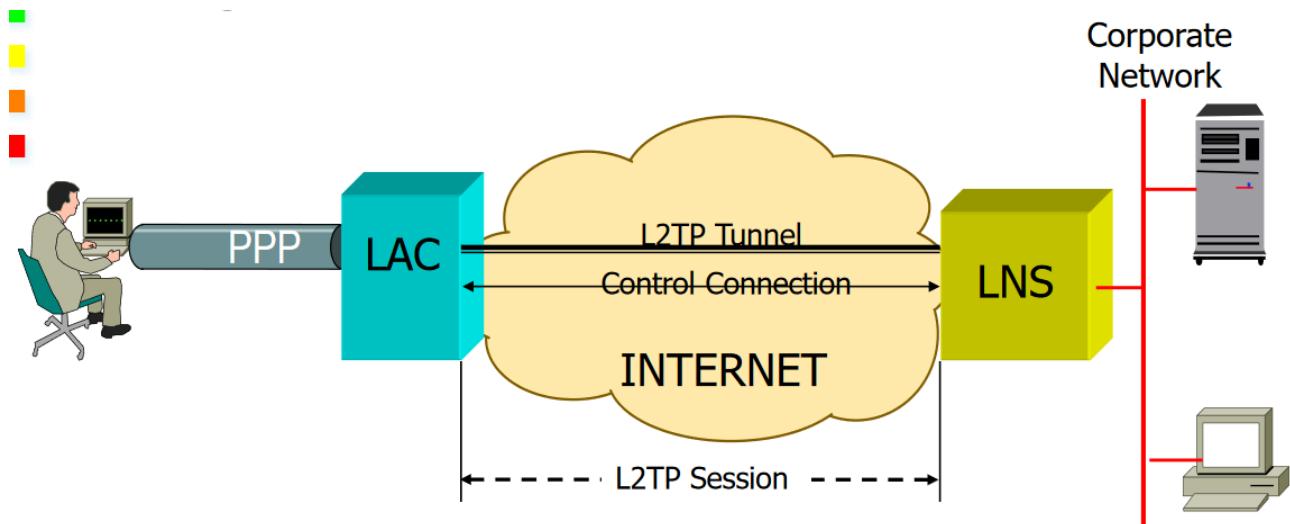
- È stato aggiunto il flag A che serve per fare l'acknowledge e sapere se il pacchetto è stato consegnato o meno
- **Acknowledgment Number** contiene il valore dell'ack
- Viene aggiunta una **key** divisa tra 16 bit alti e 16 bit bassi che vanno a identificare la **sessione**. Da un lato si identifica la lunghezza del payload, dall'altra il **call ID** (session ID).

Non fornisce alcuna forma di protezione, ma ha solo l'obiettivo di riuscire a utilizzare lo stesso tipo di indirizzamento, facendo sì che la rete privata possa avere un unico IP pubblico che è il Gateway GRE, mascherando tutto il traffico nella rete. Non fornisce **sicurezza**, ma solo **tunneling**.

Utilizzando GRE è possibile utilizzare altri meccanismi quali: **controllo di flusso**, tramite un meccanismo a finestra; **pacchetti fuori ordine** vengono scartati; **timeout** ricalcolati a ogni ricezione di ack; **controllo di congestione** facendo in modo che i timeout non causino la ritrasmissione.

L2TP – Layer 2 Tunneling Protocol

Il layer 2 tunneling protocol era un protocollo che partiva dal frame di livello 2 con un pacchetto IP. In questa soluzione, che è **indipendente dal livello 2 utilizzato**, è presente un sistema di protezione tramite **IPsec** che è sicuro ma complicato (poiché IPsec fornisce un pacchetto di livello 3 che a sua volta ha subito tunneling).



L'esempio in figura è un esempio di **provider provisioned deployment mode**. Si tratta di un utente che, attraverso una connessione Point-to-Point si connette al dispositivo per l'accesso a Internet (LAC). Qualora non sia presente un'unica tipologia di trama (a prescindere dal livello 2) si crea un tunnel sulla connessione di rete, creando così una connessione di rete con la Corporate Network.

Si crea dunque un tunneling attraverso una struttura pubblica direttamente nel sistema di **dial-up**.

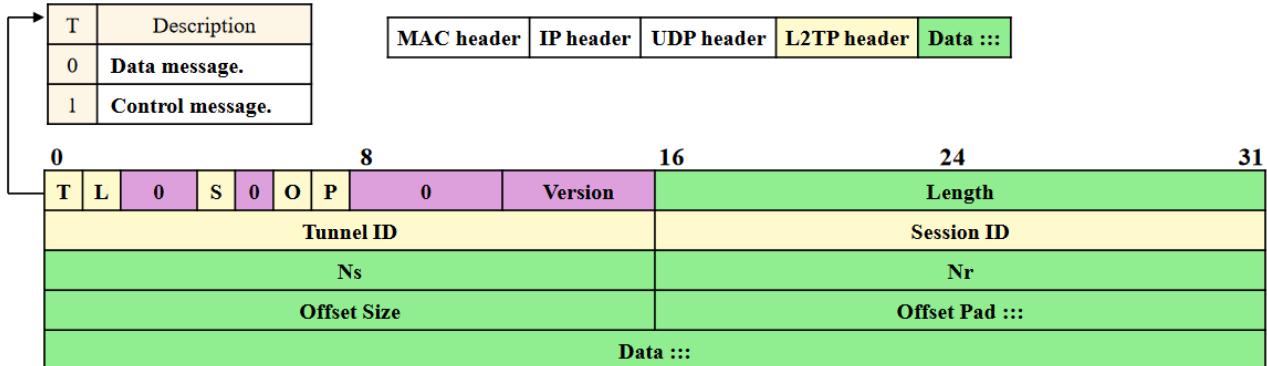
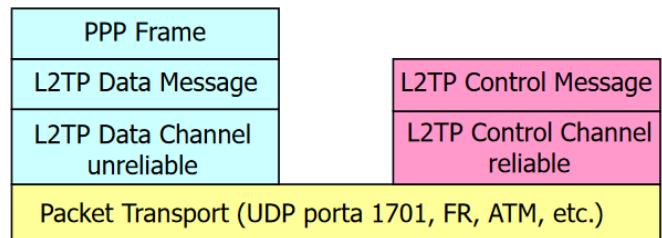
Gli elementi che caratterizzano questa soluzione sono:

- **L2TP Access Concentrator (LAC)**: crea il tunnel di tipo Layer 2 ed in questo caso fa anche da Network Access Server (NAS)
- **L2TP Network Server (LNS)**: funziona da Corporate Gateway.

Per avere un esempio di tipo di Customer Provisioned, le funzionalità del LAC vanno sostituite inserendo un LNS anche a lato utente.

Header L2TP

- Control Message
- Data Message



Si parte da un frame PPP, seguito da un L2TP Data message e da un L2TP Data Channel unreliable. Solo a quel punto, si ha il pacchetto trasporto.

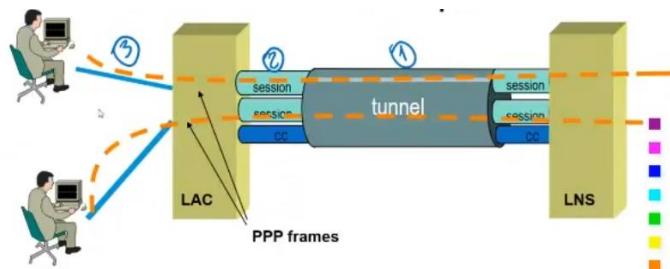
Si tratta dunque di: trama MAC, pacchetto IP, UDP e poi l'header del protocollo stesso seguito dal dato. Per realizzare questo sistema, partendo dalla trama dove in questo caso il dato è una trama al livello 2 si va a **risalire lo stack ISO/OSI**. La parte di header è dunque molto grande.

- **L, S, O** indicano la presenza di altri campi
- **P** è il priority flag
- **Ver** campo per la versione
- **Tunnel ID/Session ID**
- **Ns** è il sequence number
- **Nr** è il sequence number del prossimo messaggio da ricevere
- **Offset**

Si possono dunque creare **diverse sessioni all'interno dello stesso tunnel**. La creazione del tunnel procede come segue:

- Si stabilisce una **connessione di controllo** per un tunnel tra LAC (dispositivo del provider) ed LNS (dispositivo aziendale)
- Si stabiliscono una o più sessioni scatenate dalle richieste utente (una per utente)
- Il sistema ha la caratteristica che la parte di controllo deve essere messa in atto **prima** che un utente invii richieste di connessione
- Una sessione, a sua volta, **deve** essere stabilita prima del tunneling dei frame PPP

In ordine, sarà: tunnel, sessione, utente.



Creazione di un tunnel

- I peer possono autenticarsi
- Un **segreto condiviso** deve esistere tra LAC ed LNS
- Per creare la chiave di tale segreto, si utilizza il protocollo **CHAP** che prevede una sfida proposta a un altro peer. Se il peer risponde correttamente, viene condiviso un segreto.
- A questo punto, i tunnel end-point si scambiano il **local ID** del tunnel.

Sequence Number

- Viene utilizzato per evitare ritrasmissione errata del data stream e di replay attack.
- Non esiste un vero e proprio meccanismo di ACK
- Gli unici pacchetti a utilizzare gli ack sono i **pacchetti di controllo**.

Caratteristiche di sicurezza

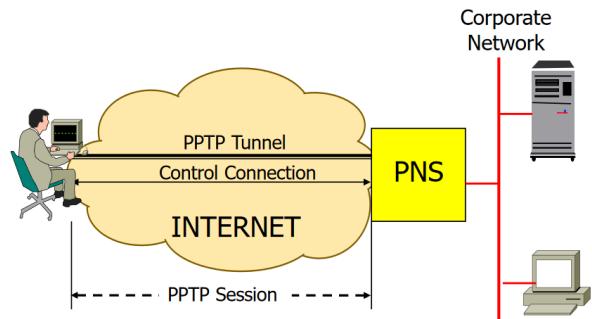
- *Tunnel endpoint authentication*
 - L'autenticazione viene effettuata solo durante la creazione del tunnel
 - Un utente può dunque effettuare lo sniffing del traffico e può instradare pacchetti malevoli nella sessione
- *Packet level security*
 - Essendo che il meccanismo prevede di avere una parte di livello 4, è possibile utilizzare **IPsec** per meccanismi di sicurezza avanzati
 - Cifratura, autenticazione a livello IP dopo il livello trasporto.
- End-to-end authentication
 - Previsto dall'utilizzo di IPsec [che richiede dunque un ulteriore tunneling]

PPTP Point-to-Point Tunneling Protocol

Veniva impiegato nei sistemi di connessione dial-up. Prevede un debole sistema di protezione.

La soluzione si basa sul tipo customer provisioned come in figura. È presente il **PPTP Network Server** (PNS) che corrisponde al corporate gateway che va a creare un tunnel col terminale utente (**remote VPN**) e la sessione viene stabilita tra l'host e il PNS. Il tunneling è di tipo **PPP frames over packet switched networks**.

È possibile trasformare la soluzione nel tipo provider provisioned, andando ad inserire il **PPTP Access Concentrator** (PAC) al lato utente.

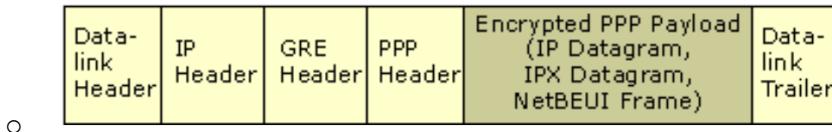


Connessioni PPTP

Sono presenti anche in questa soluzione due tipologie di connessione:

- **PPTP Data Tunneling**

- *PPP tunneling*
- *GRE (of PPP over IP)*



- La trama cifrata viene incapsulata dentro GRE, andando ad aggiungere un altro header PPP seguito da header GRE.

- **Control Connection**

- Data tunnel setup, management, e tear-down
- TCP encapsulation

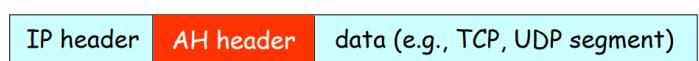
Nell'header PPP viene specificato il tipo di messaggio riferito al tipo di controllo che si vuole inviare.

IPsec

Il protocollo IPsec possiede due varianti.

La variante **Authentication Header Protocol**

(AH) effettua l'autenticazione dell'IP della



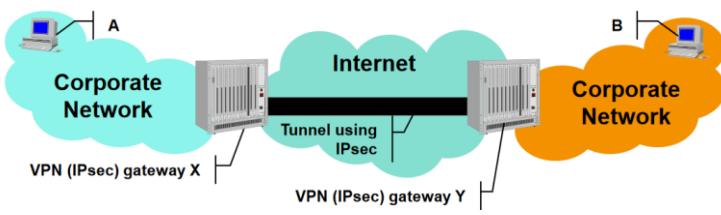
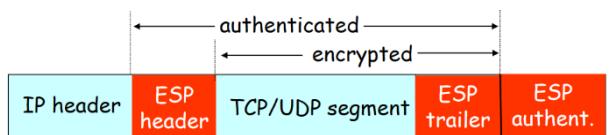
sorgente ed applica dei meccanismi che garantiscono l'integrità del dato. In questo caso, però, **non** si ha **confidenzialità**. L'header del protocollo AH viene inserito tra l'header IP originario e il payload, come in figura. Il protocollo IP sa che all'interno il protocollo successivo non è più TCP ma **AH**. Dal punto di vista del routing non vi sono problemi perché l'instradamento rimane identico a quello senza AH.

Per verificare la firma vengono utilizzati degli algoritmi di crittografia e l'autenticazione viene effettuata tramite firma digitale.

Il protocollo IPsec lavora al livello IP perciò nel payload si possono trovare TCP, UDP, ICMP.

La variante **Encapsulating Security Payload (ESP)** a

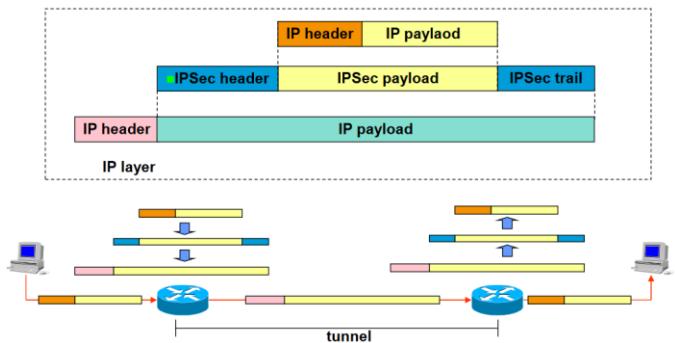
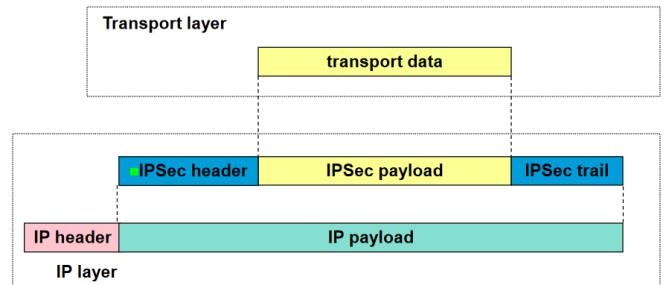
differenza di AH riesce a garantire anche confidenzialità dei dati. Vi è anche una forma di integrità sempre nel payload (l'header IP non viene protetto). La tecnica migliore è AH+ESP per garantire anche l'integrità dei dati.



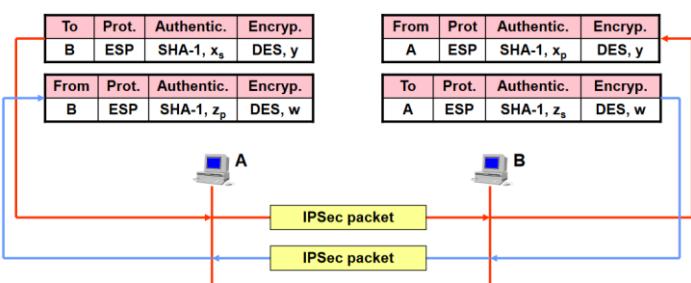
Il protocollo IPsec permette di realizzare tunnel al livello gateway o end-to-end dove il singolo pacchetto viene incapsulato e su di esse vengono posti meccanismi di cifratura e/o authentication (autenticazione + integrità).

Il protocollo IPsec si può utilizzare in due varianti: **transport mode**, dove il payload proveniente dal livello trasporto viene incapsulato attraverso l'header IPsec, che viene concatenato insieme al dato del livello trasporto. A quel punto, l'IP header **originario** (cioè quello che sarebbe stato senza IPsec) viene spostato all'esterno di IPsec. Può essere attuata nelle soluzioni **end-to-end**.

L'altra variante è **tunnel mode**, dove si va a proteggere l'intero pacchetto di livello 3 (IP header+payload) e si va ad aggiungere di nuovo l'header IPsec e a tal punto si aggiunge un nuovo header IP. Questa soluzione viene attuata nelle soluzioni **site-to-site** poiché sarà il gateway che creerà i nuovi pacchetti col nuovo IP header. In foto si nota come è il gateway che aggiunge il nuovo IP header.



La parte in cui viene creata la comunicazione sicura (collegamento tra due host nel caso del transport mode, oppure collegamento tra due gateway nel caso tunnel mode) specificherà il tipo di algoritmi di utilizzati per la confidenzialità nel caso di ESP oppure per l'integrità per ESP/AH. Per ogni organizzazione viene dunque caratterizzata una **Security Association (SA)**, cioè un luogo in cui vengono contenute diverse informazioni tra cui i protocolli utilizzati per creare la sicurezza dei protocolli. La connessione è **unidirezionale** e vengono negoziate prima di iniziare lo scambio dei pacchetti IPsec.



Nel caso di soluzione tunnel mode, nella SA si inserisce anche la tipologia di traffico che deve essere protetta in un determinato tunnel.

Considerando che per garantire la confidenzialità del traffico i due host devono scegliere una chiave condivisa, è necessario l'utilizzo di un protocollo specifico. IPsec utilizza **Internet Key Exchange (IKE)** come protocollo per lo scambio delle chiavi. La variante più sicura del protocollo IKE si basa sul possesso dei certificati.

SSL VPN

Le SSL VPN sono delle VPN che vengono realizzate al livello **trasporto (sessione)** tramite la sicurezza di SSL (che è una variante sicura di TCP) e sono in grado di realizzare **Site-to-site VPN** e **Remote access VPN** e in particolar modo possono permettere di creare accesso sicuro **ai servizi** (e non più tra host come su IPsec).

IPsec è molto complicato da complicare e da gestire, perciò le SSL VPN sono preferite poiché sono più semplici da realizzare (per quanto SSL è un protocollo molto articolato). Uno degli aspetti che favorisce la soluzione di livello 4 è che qualora si sia in grado di prendere il controllo del software che mette in piedi il protocollo IPsec, le sue operazioni essendo eseguite al livello kernel, le mal configurazioni possono essere catastrofiche (mentre con SSL VPN no).

Un altro vantaggio nell'utilizzo delle soluzioni VPN al livello 4 è che **non vi sono problemi col NAT**. Questo perché l'autenticazione viene fatta al livello trasporto e non al livello IP. Invece, uno dei grossi svantaggi di soluzioni di questo tipo è proprio l'esecuzione di operazioni al livello più alto della pila protocollare, perché ciò implica maggiore vulnerabilità ad attacchi DOS.

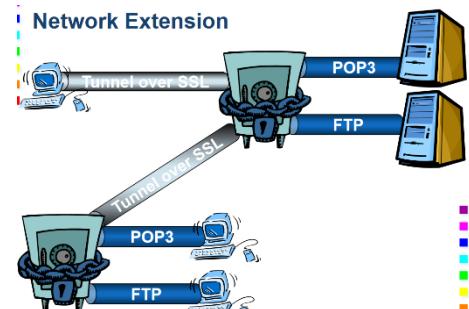
Le SSL VPN sono spesso utilizzate in soluzioni **edge to edge**.

Siccome nelle SSL VPN si collegano host a servizi, non esiste uno specifico protocollo che standardizza queste VPN.

Caratteristiche delle SSL VPN

- **Network Extension**

- Site-to-site connectivity tramite la creazione di canali sicuri tra l'host e il VPN gateway, il quale si andrà ad interfacciare con i server che tipicamente non hanno connessioni sicure. Le vulnerabilità in tale tratto rimangono presenti. Inoltre, può essere instaurata una connessione sicura anche tra due diversi VPN gateway.



Queste altre portano le VPN ad essere definite **pseudo SSL VPN**.

- **SSL'ed protocols:** molti protocolli sono protocolli non sicuri, perciò li si rende sicuri attraverso il porre questi protocolli sotto al protocollo SSL/TLS. Molti protocolli hanno una versione sicura proprio perché viaggiano su connessioni di questo tipo.
- **Application translation:** si definisce ciò quando rispetto al protocollo originale bisogna realizzare un **protocollo sicuro**. Si fa sì che il protocollo sia sicuro andando a porre l'applicazione su di un canale SSL. Certi server tale procedura non riescono ad applicarla, perciò sarà il Gateway a dialogare in maniera sicura col client ma allo stesso tempo sarà lui che decifrerà la sicurezza al livello trasporto del client e inoltrerà il pacchetto verso il server non sicuro.
- **Port forwarding:** considerando che il traffico interno è cifrato, si va a mascherare la porta utilizzata per far sembrare il traffico di un certo tipo piuttosto che un altro. Questo viene fatto per non dare informazioni sulla tipologia di traffico. Sarà poi il gateway/client in grado di decifrare il contenuto.
- **Application proxying:** il VPN gateway va a parlare il protocollo richiesto, perciò è richiesto conoscere entrambi i protocolli (sia quello verso l'utente che quello verso il server), poiché andrà a mantenere aperte due diverse connessioni.

La gestione del protocollo TCP è più complessa, dunque potrebbe portare a problemi di **performance**.

VPN Gateway Positioning & Anomalie

Problemi derivanti da una errata configurazione delle VPN.

Realizzare una VPN non è semplice anche perché possiede diversi problemi di configurazione. In particolare, bisogna capire dove inserire la VPN e il firewall in modo che possano coesistere senza problemi. Il firewall è essenziale per una azienda, poiché è necessario proteggersi da eventuali attacchi DOS ed inoltre il firewall è in grado anche di ispezionare i pacchetti. La posizione del VPN Gateway rispetto al firewall è però cruciale:

- **Dentro la rete**

- In questo modo, il firewall non può leggere il contenuto dei pacchetti poiché essi sono ancora criptati all'arrivo al firewall e devono dunque essere inviati (e fatti passare) verso il VPN Gateway. Dunque, eventuali pacchetti malevoli potrebbero transitare all'interno della rete.

- **Parallelo**

- In questo modo potrebbero esserci degli accessi non controllati

- **Fuori dalla rete**

- In questo modo il firewall può leggere il contenuto dei pacchetti poiché il VPN Gateway li sbusterà prima, ma lo si espone ad eventuali attacchi DOS

- **Integrato (VPN Gateway + Firewall)**

- Soluzione ideale, massima flessibilità (ma molto costoso)

L'**Intrusion Detection System** (IDS) è un ulteriore dispositivo che viene utilizzato per ispezionare i pacchetti e capire se una richiesta è legittima o meno. Può essere integrato nel firewall, oppure in un VPN Gateway.

Un altro problema del VPN è quello dei **NAT**. I NAT, a seconda del tipo di IPsec usato (AH,ESP) oppure Tunnel Mode/Transport Mode è possibile non avere corretta interazione tra VPN gateway e NAT.

Anomalia di Monitorability

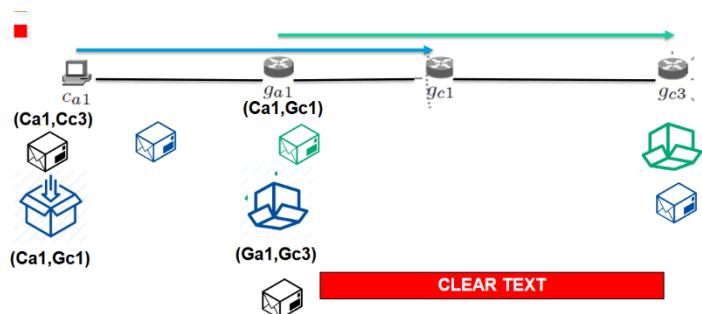
Nel caso di una soluzione s2s, il gateway risulta vulnerabile. A seconda della tipologia di traffico che si vuole scambiare, realizzare s2s o e2e ha diverse caratteristiche. Questo perché il gateway deve essere un nodo protetto, visto che può “vedere” i dati che vengono scambiati. Dunque, un accesso malevolo a questo dispositivo potrebbe portare a uno sniffing dei dati. Soluzioni s2s sono per questo motivo sconsigliate.

Skewed Channel

Lo skewed channel è un'anomalia che si verifica quando due tunnel si **overlappano** (sovrappongono), poiché potrebbe essere rimossa la confidenzialità in una parte della comunicazione.

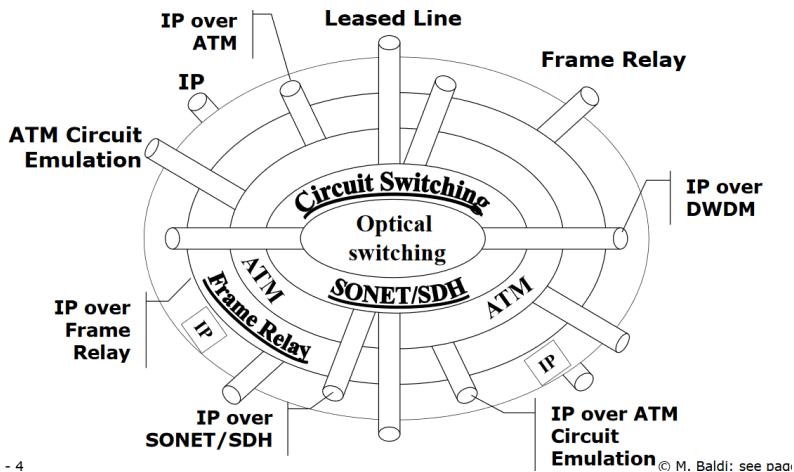
Nel caso in figura, ca1 vuole creare un tunnel con gc1 per comunicare con gc3. Allo stesso tempo, ga1 crea un tunnel con gc3. Dunque, non

appena ga1 riceve il pacchetto da ca1, leggerà che è per gc3 e dunque lo incapsulerà lungo il tunnel. Una volta arrivato a gc3, vedrà che non è un pacchetto per se ma per gc1, dunque lo ritrasmetterà. Da questo momento viaggerà in chiaro perché verrà sbustato.

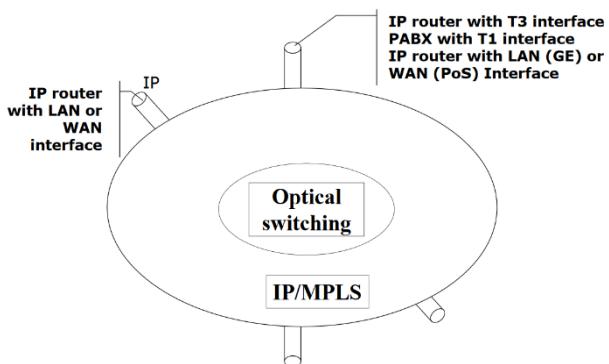


MPLS – Multi-Protocol Label Switching

"MPLS is the enabling technology for the New Broadband (IP) Public Network"



alta (Gbit/s), dunque non possono arrivare fino all'utente finale. L'operatore costruisce dunque un backbone su cui costruisce una rete di **commutazione a circuito** utilizzando SONET/SDH. Il problema di queste tecnologie è che il collegamento impegnava delle risorse nella rete sia quando si sta trasmettendo qualcosa che non, per tale motivo la rete viene impegnata per non trasmettere nulla molto spesso. Per risolvere questo problema, vengono utilizzati le reti a circuito per instradare una commutazione a pacchetto basata su **ATM**, che permetteva all'operatore di gestire anche il traffico (**traffic engineering**). Il problema di ATM è che si tratta di una tecnologia molto costosa. Essendo nata anche la necessità di collegare molti router IP alla rete, nel caso in cui questi dovessero interfacciarsi direttamente con ATM il tutto sarebbe molto costoso. Per tale motivo, attorno alle reti ATM si è creata una rete con tecnologia **frame relay** che permette di collegare i router in modo molto poco.



L'idea di MPLS è quella di cambiare il modo con cui si inoltrano i pacchetti IP. Non si vuole più guardare il next hop (indirizzo destinazione) a cui inviare il pacchetto, ma ad una **etichetta**, posta davanti al pacchetto IP. Per inserirla, è necessario dunque utilizzare un ulteriore header e quindi l'header MPLS. Mentre l'indirizzo IP identifica una singola destinazione, ed è necessario usare una soluzione di lookup che è il longest prefix matching, MPLS tramite le etichette può assegnare una o più destinazioni su dove inoltrare tale pacchetto, utilizzando le etichette come **indice**. Il vantaggio è che con gli indirizzi IP bisogna scorrere una tabella alla ricerca del prefisso dove instradare, mentre con le etichette si utilizza **un solo accesso in memoria**. Ma la ragione per cui MPLS è importante è perché si può fare traffic engineering (essere in grado di decidere che percorso fa nella rete il pacchetto). L'etichetta non viene in genere associata a una interfaccia ma a una connessione.

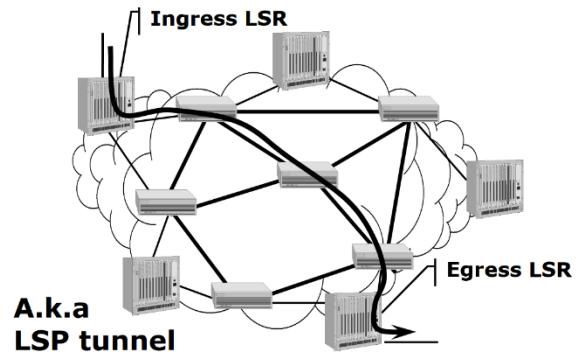
MPLS nasce per sopperire alle mancanze di efficienza di IP (poiché non era nata per servire una rete pubblica). Prima dell'avvento di MPLS, gli ISP stavano creando delle reti formate da protocolli a "cipolla", cioè un insieme di tecnologie sviluppate in modo indipendente che non erano facili da gestire. Al centro è presente l'**optical switching** che gli operatori utilizzano per formare il backbone ad alta velocità. La granularità di queste reti è però molto

Visto tutto questo insieme di protocolli, si è pensato di creare **MPLS** in modo da sostituirli tutti. Viene mantenuta la rete ottica per permettere di mantenere alte velocità. IP viene mantenuto poiché MPLS è pensato proprio per integrarsi con il livello IP. MPLS **utilizza lo stesso piano di controllo** di IP (e dunque si integrano), ma non solo. Lo stesso piano di controllo si può utilizzare **anche per la fibra ottica**. In sostanza, si possiede un **unico piano di controllo**.

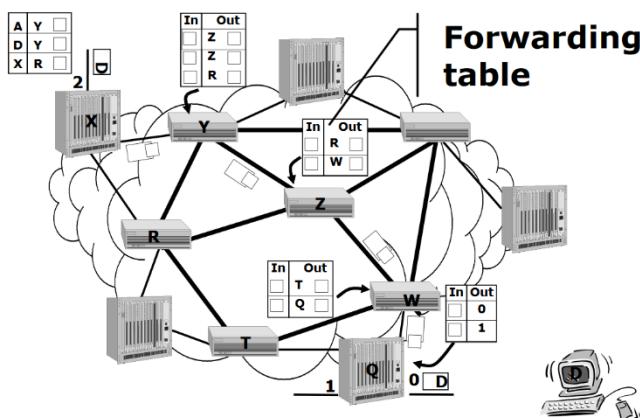
Label **IP packet**

MPLS introduce un paradigma orientato alla connessione (a circuito) nelle reti IP. La rete continua ad essere commutazione di pacchetto, ma semplicemente esiste un concetto di circuito virtuale che non occupa risorse quando non utilizzato.

MPLS non viene utilizzato in tutte le reti, ma solo in alcune dette **MPLS network/MPLS cloud**. Nella rete MPLS sono presenti dei router detti **Label Switch Router (LSR)** che fanno la commutazione a pacchetto basata su etichetta. L'etichetta viene inserita dai router che stanno al bordo della rete detti **Label Edge Router (Ingress/egress LSR)**. Ai pacchetti viene messa un'etichetta quando entrano nella rete e viene tolta quando escono dalla rete MPLS. Affinchè due stazioni possano mandarsi dei pacchetti è necessario creare una "connessione" all'interno della rete detto **Label Switched Path (LSP)**. Ciò vuol dire che gli LSR devono essere a conoscenza di tale LSP per sapere cosa fare dei pacchetti che arrivano con quella specifica etichetta (esiste anche una modalità tale per cui non sia necessario dover esplicitamente creare prima le LSP).



Label Switching



osserverà che le "etichette azzurre" [etichetta che verrà assegnata alla fine] devono andare all'uscita 0 ma senza etichetta (poiché la rete MPLS termina). Quando si crea l'LSP bisogna decidere che etichetta usare, ma X deve scegliere un'etichetta che non è usata su alcun link e inoltre le etichette devono essere **univoche** per ogni LSP. Infine, se le connessioni sono molte le etichette possono diventare "lunghe", ma ciò non lo si vuole. Cambiando dunque le etichette ad ogni nodo è più facile scegliere le etichette perché queste dovranno essere univoche **solo tra due router**.

Quando un pacchetto arriva alla rete MPLS, il pacchetto arriva al Label Edge Router, si guarda l'indirizzo di destinazione e si ricerca una entry nella tabella per andare da D (utente finale) che mostra di aggiungere una "etichetta nera" e di inviarlo al router vicino Y. Quest'ultimo non guarderà più l'indirizzo di destinazione, ma si guarderanno soltanto le etichette. Osserverà che le "etichette nere" sono da inviare a Z e va inserita una "etichetta viola". Ecco da dove viene il termine **label swapping**. Il router finale Q

Componenti chiave di MPLS

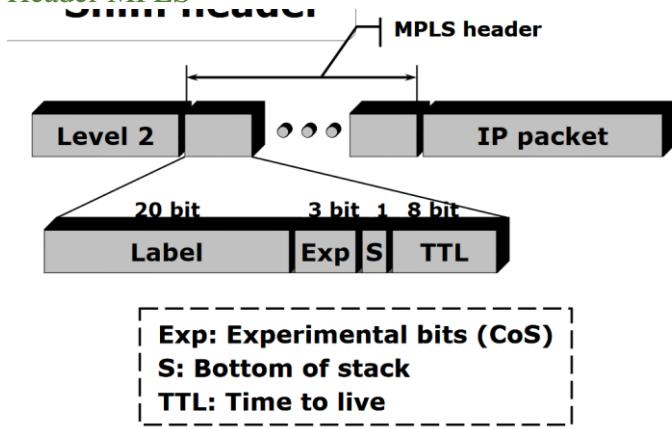
- **MPLS "header": contiene l'etichetta**
- **Protocolli per la distribuzione delle etichette:** protocolli di **segnalazione**.
- **Protocolli di routing potenziati:** i router della rete MPLS devono sapere come raggiungere le destinazioni che stanno fuori dalla rete MPLS. Stessi protocolli di IP potenziati per portare informazioni che vengono usate come dei vincoli nella scelta dei percorsi

IP sceglie il percorso più breve

MPLS sceglie il percorso più breve **che rispetta determinati vincoli**.

Quando un router deve inviare un pacchetto alla destinazione D, deve poter creare un LSP verso tale destinazione. Per crearlo, esso si integra con IP e utilizza gli stessi algoritmi di routing per l'assegnazione delle etichette. Una volta fatto ciò, i router della rete MPLS non guarderanno più la destinazione IP ma solo le etichette.

Header MPLS



L'intestazione header è chiamato anche **Shim header**, e viene definito così perché l'intestazione MPLS è in realtà un'intestazione modulare fatta di uno o più moduli. Tali moduli sono inseriti tra l'intestazione di livello 2 e quella di livello 3. Ognuno dei moduli contiene l'**etichetta** lunga **20 bit** (unica solo su un link), quindi finché non si ha più di 1 milione di flussi che passano su un link questi bastano. Successivamente sono presenti 3 bit sperimentali che in realtà vengono utilizzati per creare diverse

classi di servizio (cioè si tratta in modo diverso i pacchetti provenienti da diverse etichette). Il bit **S** è il **bottom of stack**. Quando il router MPLS riceve le trame, viene visto soltanto il primo modulo ed il bit sta ad indicare se dopo questo modulo sono presenti altre etichette o meno (1 ultima etichetta, 0 ci sono altre etichette). I moduli vengono gestiti come uno stack e l'aggiunta/rimozione avviene sempre dalla testa (sx nella foto in alto). Infine, è presente anche il **TTL** di 8 bit.

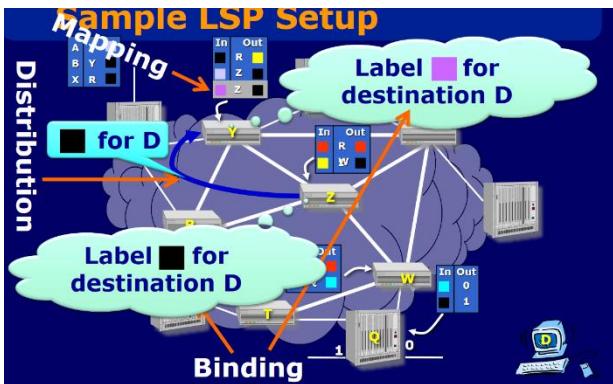
Le reti ATM e frame relay utilizzano il label switch, perciò negli header di livello 2 è già presente un campo per contenere un'etichetta. Per tale motivo, è possibile utilizzare tale campo per MPLS (senza creare un header aggiuntivo). A questo punto, si sta affermando che se si inserisce l'etichetta in un campo di un protocollo di livello 2, si sta dicendo che uno switch ATM/Frame Relay in realtà sono degli switch MPLS.

Setup LSP: selezione del path e delle etichette

Per la creazione degli LSP è necessario introdurre la **forwarding equivalence class (FEC)** che permette di specificare cosa viene trasportato da un LSP. In altre parole, è l'insieme di quei pacchetti che sono trattati allo stesso modo dagli LSR (ricevono la stessa etichetta e dunque effettuano lo stesso percorso).

Gli LSR per creare gli LSP effettuano le seguenti operazioni:

- **Label binding:** scegliere un'etichetta che deve essere associata ai pacchetti che appartengono a un certo FEC. Il binding viene effettuato **downstream**, cioè che tra due nodi che si trovano ai capi di un link quello che esegue il binding è colui che sta a valle. Per sapere di dover usare tale etichetta, l'LSR deve essere **notificato**. L'LSR per assegnare l'etichetta può farlo in due modi: **unsolicited**, dunque, senza che nessuno glielo chieda; **on-demand** dietro richiesta di qualcuno.
- **Label mapping:** creare un'associazione tra l'etichetta che arriva **in ingresso** e quella **in uscita** ed anche decidere a quale **next hop** deve essere inviato il pacchetto. Le etichette le sceglie l'LSR, mentre il next hop tramite routing.
- **Label distribution:** notifica da parte di un LSR verso gli altri LSR che lo circondano dell'etichetta scelta per un certo FEC. Dopo che un LSR effettua il binding, lo comunica agli altri (a seconda di come è stato fatto il binding).



Nell'esempio in figura Y vuole avere un'etichetta per i pacchetti che vanno a D (sta lavorando sul FEC di D). Quando sceglie l'etichetta, non può ancora fare il mapping. A un certo punto anche Z vuole fare il binding per lo stesso FEC. A quel punto viene distribuita a tutti i vicini e, per esempio, a Y. A questo punto Y ha gli ingredienti per costruire il mapping: conosce l'etichetta di Y, quella di Z e il next hop (che è Z). Dunque, il mapping viene creato.

Static label binding (and mapping)

In questo caso è il gestore della rete che gestisce l'etichetta da usare e lo impone ai vari nodi della rete. Ciò equivale a creare i circuiti permanenti in ATM. **Non è scalabile** e non c'è **interoperabilità** tra sistemi di controllo. È inoltre avere LSP che passano tra diversi operatori.

Dynamic Label Binding

Può essere scatenato a due livelli:

- **Data/traffic driven:** scatenato dal fatto che arriva un pacchetto
- **Control driven:** la decisione di associare etichette al FEC è scatenato dall'arrivo di un messaggio di controllo (**signaling**). Questo si utilizza nel caso di solicited binding. È possibile anche che si scateni dell'arrivo di un messaggio di routing (a seguito della scoperta tramite protocollo di routing di una certa destinazione).

Control Driven Label Binding

La creazione degli LSP control driven da origine a due tipi di LSP diversi:

- **Topology based:** un LSP viene creato quando viene scoperta una route per una destinazione. In questo caso il FEC è la destinazione (ovvero un prefisso) - routing
- **Creazione esplicita degli LSP:** messaggio esplicito che chiede l'associazione di una etichetta a quel particolare FEC. (signaling)

Protocolli per la distribuzione delle etichette

MPLS possiede tre diverse alternative:

- **Routing protocol: BGP**
 - Utile soprattutto quando si vuole creare LSP con modalità **topology based**.
- **Label distribution Protocol (LDP)**
 - Protocollo proprietario derivante da TDP (Cisco)
 - Deprecato
- **Resource reservation protocol (RSVP)**
 - Preesistente ad MPLS e progettato per **supportare qualità del servizio** nelle reti.
 - Il più utilizzato

Se è necessario creare un LSP è necessario utilizzare **uno solo** tra questi standard.

Protocolli di routing

Servono per determinare il percorso che fa l'LSP. L'LSR costruisce la sua tabella di routing e la usa quando fa il mapping, poiché deciderà un next hop già calcolato col protocollo di routing. È proprio ciò che **indirettamente determina il percorso dei pacchetti**. I protocolli utilizzati sono quelli già esistenti, in particolare: **OSPF, IS-IS, BGP-4**. Non si utilizza RIP perché non funziona bene per reti grandi.

Questi protocolli permettono ai router non solo di scambiarsi informazioni topologiche, ma i protocolli vengono modificati per **trasportare informazioni che servono per vincolare le informazioni di routing** come ad esempio:

- Capacità dei link
- Utilizzo dei link
- Dipendenza tra link (utilizza per recuperare i guasti)

Le versioni modificate di questi protocolli di routing si utilizzano per fare il **constraint based routing**, fondamentale per supportare il **traffic engineering**. I protocolli modificati sono: **OSPF-TE, IS-IS-TE**. BGP non viene usato per fare constraint based routing.

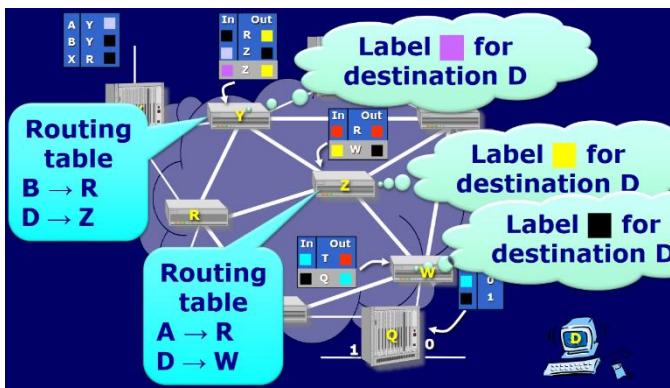
Modi di routing

- **Hop-by-hop routing**: classico
- **Explicit routing**: si vuole utilizzare delle informazioni che cambiano in modo dinamico

Hop-by-hop routing

Ogni LSR decide il next-hop, cioè il prossimo LSR sul percorso dell'LSP. Ciò è proprio quello che succede su IP. Il router decide sulla base di una tabella di routing. Nella creazione di LSP succede che:

- Quando un router sceglie un'etichetta (Label binding) lo **mappa al next-hop**. Da solo, decide chi è il next-hop. Ogni router sul percorso LSP farà la stessa cosa.
- Associa nella sua tabella di forwarding all'etichetta scelta, quella scelta dal next-hop.



Ancora una volta (Y non ha ricevuto pacchetti verso D), Y sceglie l'etichetta viola per D, Z sceglie quella gialla sempre per D. Z distribuisce l'etichetta a Y e dunque quest'ultimo sa con che etichetta mandare i pacchetti diretti a D. Y guarda la tabella di routing e andrà a controllare come andare da D. Per poter andare a D bisogna passare attraverso Z. A questo punto l'entry in tabella di forwarding può essere creata e dunque tutti i pacchetti per D andranno verso Z con etichetta gialla. La stessa cosa accadrà con gli altri router. Ognuno sceglie per conto suo chi è il next hop.

Explicit constraint based routing

In MPLS esiste una cosa simile al source routing detta explicit constraint based routing. In questo caso si fa routing **nel piano di controllo**. Un singolo switch sceglie il percorso per l'**intero LSP** (ad esempio l'ingress LSR) e successivamente comunica in esplicito (explicit routing) che l'LSP deve passare su tale percorso. La scelta del percorso viene utilizzata utilizzando il **constraint based routing** perché immaginando di voler scegliere un percorso che passi attraverso link carichi <50%, questa è un'informazione dinamica e dunque nelle reti MPLS accade che ogni volta che cambia il carico dei link bisogna continuamente inviare dei messaggi

con l'informazione ai vari router e questo vuol dire che Y e Z potrebbero avere nello stesso momento una versione diversa della rete. In MPLS si utilizzano informazioni dinamiche assicurandosi che i router non creino percorsi incoerenti, e per fare ciò invece di far scegliere il percorso hop-by-hop lo si fa calcolare tutto ad un **unico router**. In tal modo il calcolo del percorso sarà coerente anche se non preciso (non conosce il carico di ogni nodo).

A questo punto è necessario modificare anche i protocolli per la distribuzione delle etichette, perché non servono solo più per associare l'etichetta a un FEC ma servono per dire anche ad un nodo su quale nodo passare. I protocolli modificati sono:

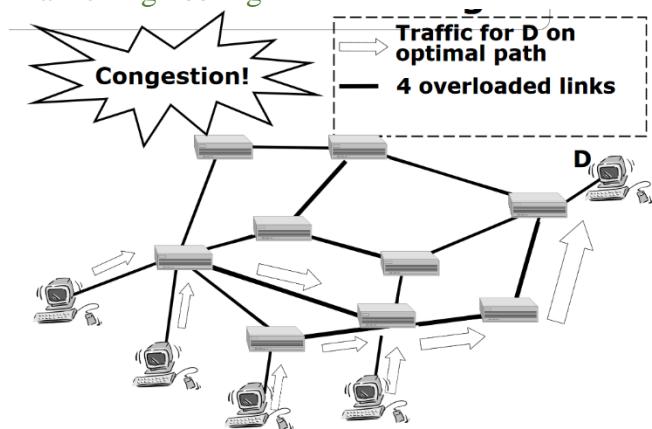
- CR-LDP
 - Constraint-based routing LDP.
- RSVP-TE
 - Permette di trasportare informazioni sul percorso da seguire

Questi vengono usati insieme ai protocolli di routing modificati.

Queste modifiche ai protocolli permettono di:

- Fare **traffic engineering**
- Aiuta a fornire **un servizio a qualità garantita** (non si usa)
- Traffic engineering **per classe**
- Sinergia con i servizi differenziati (*DiffServ*)
- **Recupero veloce dei guasti (<50ms)**

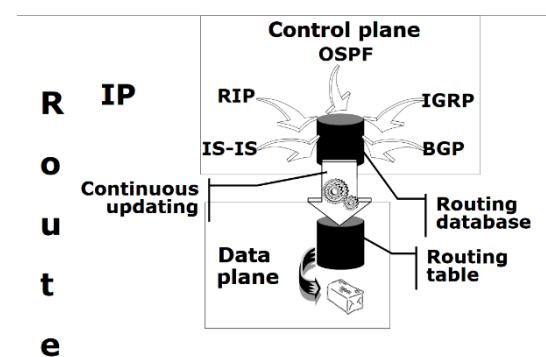
Traffic Engineering

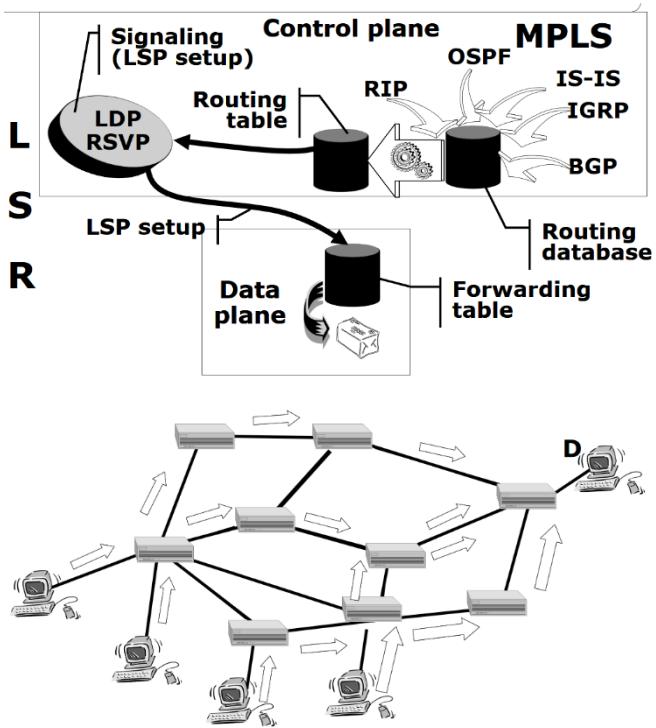


percorsi diversi. Farlo con IP ciò è un problema.

Anche se fosse possibile aggiornare le informazioni su IP istantaneamente riguardo al carico della rete, questi andranno a modificare il piano dati. I protocolli di routing eseguono il piano di controllo: quando questi scoprono un nuovo percorso, vanno a modificare la tabella di routing che si trova nel piano di dati (utilizzata quando arriva un pacchetto per farne l'inoltro). Ogni volta la tabella viene aggiornata e accade che il traffico si sposta secondo la nuova tabella. In sostanza si crea un imbuto da un'altra parte. Ciò è dovuto che un cambiamento topologico si riflette immediatamente nel piano dati e tutto il traffico si sposta.

Su una rete, utilizzando IP tradizionale, il traffico segue i percorsi indicati in foto. La dimensione della freccia indica la quantità di traffico che va a D. Si nota che il traffico va a imbuto, cioè si incanala su dei link che diventano dei colli di bottiglia. Si crea **aggregazione** poiché tutto il traffico viene inoltrata verso specifici router, ma allo stesso tempo 4 dei link in figura diventano dei link **congestionati** mentre altri rimangono **sottoutilizzati**. Un ISP vorrebbe distribuire il traffico in modo da arrivare a D utilizzando





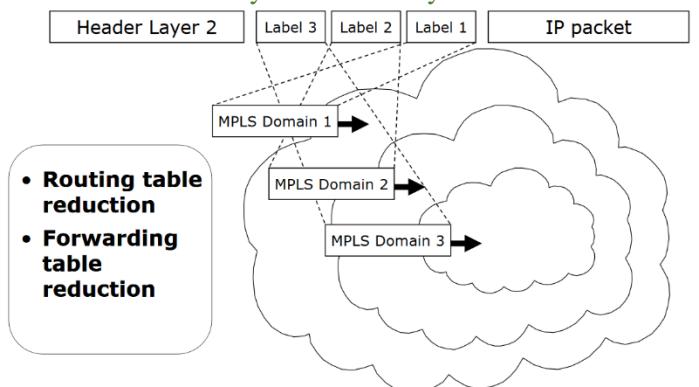
Utilizzando IP, questo non è possibile.

Fast Fault Recovery

Una volta creato l'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. Si può utilizzare il seguente metodo:

- Creare un altro LSP che funge da **backup** del link, fatto non quando il link si rompe ma a priori. Quando due nodi si scoprono, si creano 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**.

Label Stack hierarchy and scalability



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. Si crea quando viene fatta la **distribuzione delle etichette** nel setup dell'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.

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.

Penultimate Hop Popping (PHP)

È possibile che non sia l'ultimo nodo a eliminare l'etichetta, ma il penultimo LSR. A quel punto l'Edge Router si ritroverà un pacchetto senza etichetta. In tal caso, vuol dire che l'Edge Router guarderà un'altra etichetta oppure l'indirizzo IP. Per utilizzare il PHP il nodo deve essere al corrente che il nodo precedente ha rimosso l'etichetta.

Questo si fa perché in molti casi l'ultimo nodo non ha bisogno di vedere quella etichetta, visto che se non ci sono altre etichette l'ultimo nodo dovrà comunque guardare l'indirizzo IP per portarlo a destinazione.

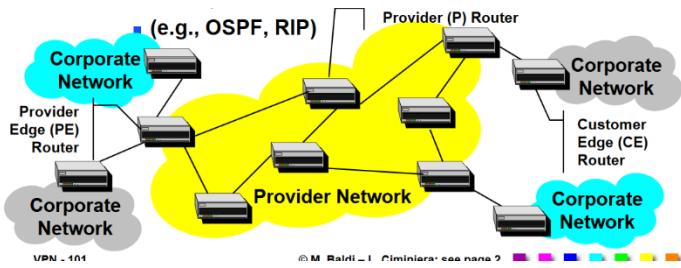
- **PHP implicito:** l'etichetta più esterna viene rimossa
- **PHP esplicito:** l'etichetta non viene rimossa, ma ci si scrive “0” all'interno

Soluzioni MPLS-based

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 nel backbone non possono contenere tale IP destinazione. Un'alternativa sono i NAT oppure i **tunnel**. L'LSP in realtà sono dei tunnel, visto che gli LSR non guardano l'indirizzo IP destinazione ma l'etichetta, perciò basta utilizzare le etichette per indirizzare i pacchetti a destinazioni diverse.

La prima possibilità è quella di utilizzare **soluzioni MPLS basate sul livello 2**, andando a creare LSP tra tutti i router di bordo della rete MPLS che devono fornire connettività aziendale. Questa soluzione viene chiamata anche **Pseudo Wire Emulation End-to-End (PWE3)** perché è come se tra due aziende si creasse un cavo per l'inoltro dei pacchetti. Per implementare questa soluzione si utilizzano **due etichette**:

- Innanzitutto, si crea una LSP per collegare i dispositivi di bordo dell'operatore (PE). I Customer Edge (CE) sono collegati ai PE (Label Edge Router).
- Quando i pacchetti arrivano al CE, questo li inoltra al PE il quale affiggerà una etichetta diversa in base al tipo di traffico (ethernet o IP).

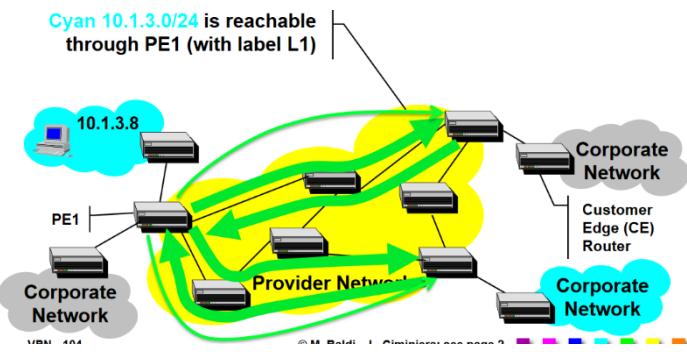


L'altra alternativa sono le **MPLS-based Layer 3 VPNs**, che consentono di creare gli LSP automaticamente. L'idea è quella di creare LSP tra i vari PE e poi creare degli LSP all'interno per il collegamento delle reti private attaccate a questi PE. Questa soluzione funziona **solo per il trasporto di pacchetti IP**. Per rendere tutto

automatico è necessario scambiarsi 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.

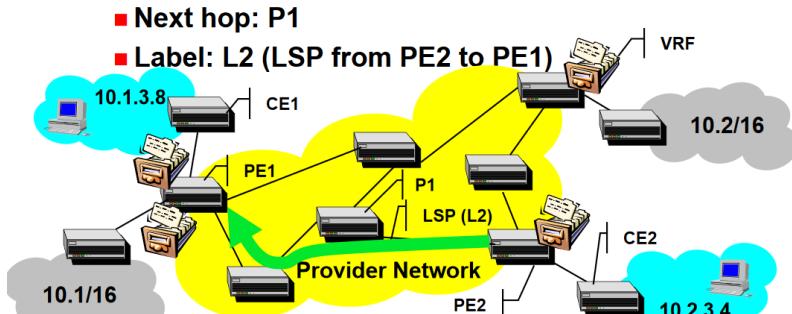
A questo punto ci sono due alternative che formano due diverse soluzioni:

- Una soluzione utilizza il protocollo **I-BGP** per scambiare informazioni di routing. In questo caso, un router, nel dire agli altri router che una destinazione è a lui direttamente raggiungibile, comunica anche altre informazioni “La rete cyano 10.1.3.0/24 è raggiungibile”. In particolare, lo comunicherà al CE azzurro. Il PE connesso a quel CE si costruirà una tabella in cui scriverà che quando bisogna inviare pacchetti alla destinazione 10.1.3.0/24 si deve usare l’etichetta L1 e si devono inviare al router che gli ha comunicato tale informazione. Questo router non avrà un’unica tabella, ma tante quante sono le reti private a cui lui è collegato.
- L’altra utilizza i **router virtuali**.

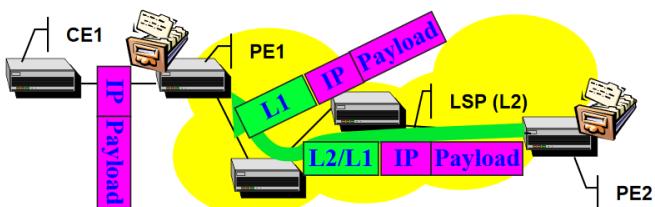


Esempio di routing

- PE2 looks-up 10.1.3.8 in cyan VRF
 - Next hop: PE1
 - Label: L1 (distributed by PE1 for cyan 10.1.3.0/24)
- PE2 looks up PE1 in main table
 - Next hop: P1
 - Label: L2 (LSP from PE2 to PE1)



- PE2 has pushed L1 and L2 on label stack
- P routers forward packet to PE1 using L2
- Last hop before PE1 pops L2 (PHP)
- PE1 receives packet with L1
 - PE1 pops L1: plain IP packet
 - PE1 uses L1 to route packet to proper output interface



I benefici di questa soluzione sono:

- 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 il backbone
- Il provider non ha un backbone virtuale per cliente
- La VPN può passare tra provider diversi
- La sicurezza non è basata sulla sicurezza ma ci si fida del service provider

MPLS/BGP VPN

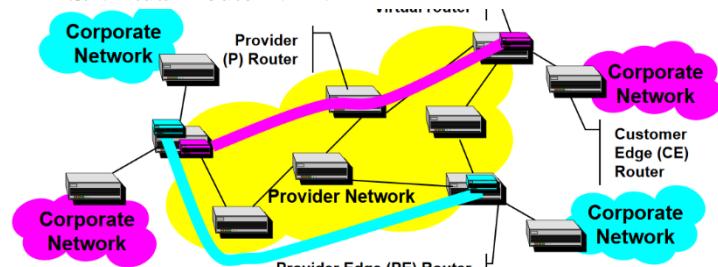
Oltre all'indirizzo si specifica la VPN a cui appartengono (ed è il motivo per cui si usa BGP). Il BGP è un protocollo che può facilmente essere esteso e in questo caso si chiama MP-BGP (Multi-Protocol GBP) poiché permette di trasportare informazioni su indirizzi che non sono indirizzi IP.

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**.

Route Distinguisher IP Address

È proprio questo campo che annuncia l'indirizzo IP "cyano".

MPLS/Virtual Router VPN



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.

Reti ottiche

Il concetto principale per le reti ottiche è il **WDM – Wavelength Division Multiplexing** cioè la possibilità di trasmettere sullo stesso mezzo a frequenze diverse. Nelle reti ottiche il mezzo è la fibra: si prendono più segnali ottici e li si trasmette nella stessa fibra. Esistono più tipi di WDM:

- DWDM – Dense WDM
 - 1 fibra tanti canali
- CWDM – Coarse WDM
 - 1 fibra pochi canali

Inizialmente la ragione per cui è stato fatto il WDM era per poter riutilizzare la stessa fibra.

Successivamente però, si è notato che avendo tanti segnali ottici si potrebbe costruire un commutatore che riconosce i vari canali e li inoltra su diverse fibre. Questo si definisce **commutatore ottico** (wavelength switch). L'idea 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**.

Le reti ottiche si utilizzano nel centro della rete perché permettono di commutare un canale ottico. 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.

Optical Core

Con un commutatore ottico si vorrebbe spostare un segnale ottico che arriva su una fibra per spostarlo su un'altra fibra, continuando a farlo nel dominio dell'ottica (e quindi senza leggere i bit). Per implementare questa idea, si utilizzano degli **specchi** (superfici riflettenti) che prendono il segnale in modo da riflettere il segnale più volte per spostarlo. Il problema è che lo specchio non deve essere fisso, ma mobile e in più si vuole dei commutatori che commutano fino a 1000 segnali diversi. Esiste una tecnologia **MEMS** per incidere nel silicio dei **micro-oggetti**. Un'altra possibilità è quella di fare degli **ologrammi**.

Tutto ciò si traduce in **alti costi** e grosse difficoltà di implementazione. Per tale motivo, si è finiti a fare dei **core elettrici**, che traducono il segnale in bit, lo spostano e lo riportano in fibra.

Vantaggi delle Optical Core:

- Potenzialmente non costosi [quando la tecnologia è matura]
- Indipendenti dal bit rate e dal segnale
 - Ciò si traduce in scalabilità illimitata (non sarà necessario aggiornare i dispositivi in futuro)
- Bassi consumi di potenza
 - Le operazioni hanno costi di potenza molto bassi

Svantaggi delle Optical Core:

- Alti costi di produzione [tecnologia immatura]
- Alta attenuazione (e no rigenerazione)

I core elettrici posseggono dei dispositivi che commutano i canali ottici con un circuito elettronico, che costa poco ma consuma più potenza.

Quality of Service

Le applicazioni multimediali sono molto diverse rispetto alle applicazioni tradizionali rispetto cui internet è stata creata. Sono diverse perché le applicazioni multimediali generano un **flusso continuo di dati** ed il profilo generato da questo flusso deve essere **lo stesso** di quello che viene ricevuto, in modo che il ricevitore possa semplicemente riprodurre le informazioni che riceve e mostrarlo all'utente senza doverlo memorizzare o ritemporizzare.

Le applicazioni multimediali sono anche **interattive**: interazioni con un altro essere umano o con un computer. Ciò richiede **bassi tempi di risposta**. Molte applicazioni multimediali come i video richiedono **grande banda di trasmissione**.

I requisiti di rete variano in base al tipo di applicazioni:

- **Streaming**: perdita limitata dei dati; ritardi costanti
- **Interattività**: bassi ritardi, ottenibili fornendo garanzie sulla qualità del servizio offerto

Per ottenere qualità del servizio devono essere presenti molte risorse nella rete, tanta capacità trasmissiva, buffer più grandi nei nodi della rete, capacità di commutazione, capacità di elaborazione. Tutto ciò richiede che vi siano tecnologie adeguate.

Il vero problema che la QoS deve risolvere 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. I ritardi variano molto nel tempo a seconda del carico sul nodo.

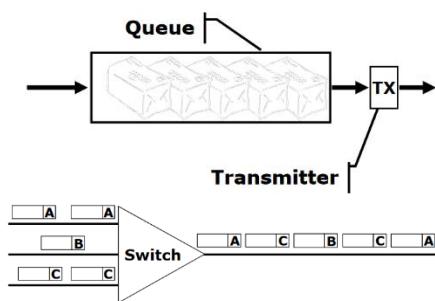
Le contromisure che si possono realizzare nella rete sono le seguenti:

- **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.

Classificazione

Per effettuare la classificazione del traffico bisogna andare a vedere qualcosa che individui univocamente i pacchetti che appartengono a una certa comunicazione. Per fare ciò si guarda la seguente quintupla: IP sorgente, IP destinazione, protocollo trasporto, porta destinazione e porta sorgente. Fare questa operazione non è banale: è necessario un meccanismo che sappia già, prima ancora che arrivino i pacchetti, come indirizzarli in base alla quintupla. Bisogna dunque utilizzare delle implementazioni hardware come: 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.

Scheduling



Per lo scheduling è possibile utilizzare una coda semplice FIFO, ma questo non risolve il problema del QoS, poiché l'ultimo pacchetto che arriva sarà l'ultimo a uscire sempre. L'effetto del FIFO è il **multiplexing statistico**, cioè i pacchetti vengono sequenziati sul link di uscita in maniera casuale in base all'ordine d'arrivo.

Per poter garantire il singolo flusso bisogna guardare i pacchetti che arrivano e in base al flusso bisogna inserirli in **code multiple** in modo da servirli “intelligentemente” in base all’urgenza. L’urgenza è scelta in base a diversi metodi:

- Priority queuing
- Round Robin
- Class Based Queuing (CBQ)
- Weighted Fair Queueing (WFQ): il più sofisticato. Permette di garantire che ogni flusso nella rete riceva lo stesso servizio che riceverebbe se avesse un collegamento dedicato da ingresso a uscita nella rete con un certo bitrate.
- Deadline queuing

Controllo del traffico

Il controllo del traffico si può effettuare al livello di:

- pacchetto e viene detto **policing del traffico**. La rete all’ingresso (dove sono collegati gli end system) deve controllare quanto traffico entra e non farne entrare più di una certa quantità. Lo **shaping** si assicura che i dati vengano inviati verso la rete rispettando i limiti di traffico che tale connessione permette di generare.
- chiamate: meccanismi di segnalazione con prenotazione delle risorse. In questo modo si può accettare/rifiutare la richiesta di una connessione nel caso in cui essa sia possibile (risorse disponibili) o meno. Il protocollo che si occupa per fare ciò è **RSVP** (Resource reSerVation Protocol).
- A priori: ingegnerizzazione opportuna della rete, dimensionando la rete in base a ciò che ci si aspetta e limitando il numero di utenti. A priori è possibile effettuare anche traffic engineering, andando a controllare la distribuzione del traffico lungo la rete.

Standard

Esistono due standard per supportare QoS: **IntServ** e **DiffServ**.

- **IntServ**: vuole garantire QoS ed usa le contromisure già citate nel modo più sofisticato possibile: si effettua la prenotazione delle risorse (RSVP); garantisce QoS a ogni singolo flusso. Tutto ciò però è molto complesso e poco scalabile. Per quanto lo standard sia pronto e implementato nei router, non viene utilizzato.
- **DiffServ**: non garantisce QoS, perciò non è neanche richiesta la prenotazione delle risorse. Si effettua semplicemente una differenziazione delle classi di traffico tramite un particolare campo detto **DS Field**. Combinando questa differenziazione, insieme al network/traffic engineering e all’accesso controllato, è possibile limitare i pacchetti presenti nel buffer. Bassa efficienza (best effort), semplicità e scalabilità. Sempre più usato.