

TECNOLOGIE & SERVIZI DI RETE

0) RIASSUNTO IPv4

IPv4 ha 32 bit e viene usato per identificare **ROUTER** e **HOST**; un indirizzo IPv4 è diviso in 2 parti:

- **network part** → bit più alti, identificano la rete dove l'host si trova
- **host part** → bit più bassi, identificano l'host nella rete

Ci sono alcuni indirizzi speciali:

- **N bit + tutti 0** → **NETWORK** (o subnetwork) ID
- **Tutti 1** → **BROADCAST**
- **N bit + tutti 1** → **DIRECTED BROADCAST** (solo per la rete)
- **127 + N bit** → **LOOPBACK**

Le rappresentazioni possono essere "classes" o "classless" (CIDR) per determinare la separazione tra network e host part. Le **CLASSI** sono:

- **A** → **0 + N bit**
- **B** → **10 + N bit**
- **C** → **110 + N bit**
- **D** → **1110 + N bit**
- **E** → **11110 + N bit**

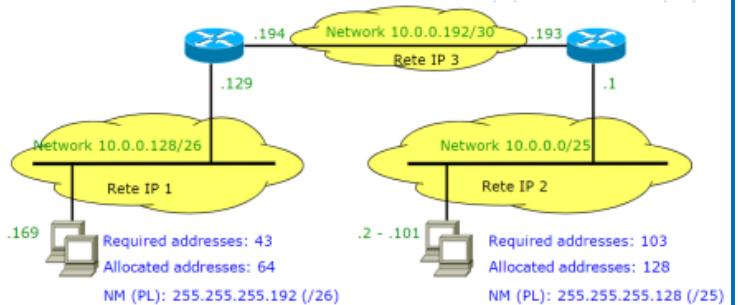
Il sistema **CIDR** (Classless InterDomain Routing) rende la network part di lunghezza variabile perché è composto da **networkID + prefix_length** (**/x** è il numero di bit della rete) oppure **netmask** (una serie di bit posti a 1 che determinano quali bit identificano la rete con un and bit a bit) [es. **prefix_length** → 200.23.16.0/23; **netmask** → 200.23.16.0 255.255.255.254.0]. L'indirizzo IP viene diviso in 4 gruppi da 8 bit (ogni bit avrà quindi un valore tra 0 e 255 = $2^8 - 1$) [in realtà non tutti i valori sono permessi da 0 a 255, infatti il più grande è 252].

⚠ Per verificare se un indirizzo è corretto basta prendere il **prefix_length /x** e controllare che l'ultimo numero puntato sia multiplo di 2^{32-x} [es. 130.192.1.4/30 ha $4\%2^{32-30}=4\%4=0$ quindi va bene, mentre 130.192.1.1/29 ha $1\%2^{32-29}=1\%8!=0$ quindi non va bene] (è evidente che un indirizzo che termina con .1 non sarà mai corretto).

Il **ROUTING** degli host avviene attraverso la **ROUTING TABLE** (2 colonne che identificano destinazione [indirizzo IP] e interfaccia): quando viene inviato un pacchetto, si cerca un match all'interno della tabella per identificare dov'è la destinazione, e se c'è più di 1 match, si va di **LONGEST PREFIX MATCHING** (prefisso più lungo) [i router sono disegnati con un cerchio con all'interno una X].

Riguardo all'**IP ADDRESSING METHODOLOGY**:

1. Localizzare le reti IP
2. Individuare n° di indirizzi richiesti
3. Calcolare n° di indirizzi allocabili
4. Verificare il range di validità degli indirizzi
5. Calcolare la netmask/prefix_length
6. Calcolare address range
7. Calcolare gli indirizzi degli host



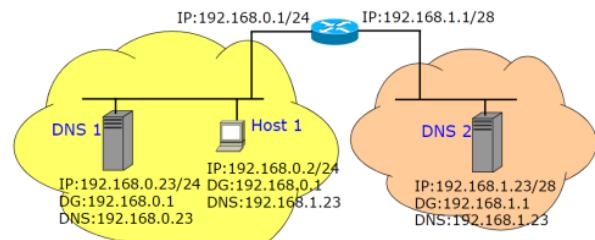
Dall'**ESEMPIO** qui a destra, per riuscire a trovare le sottoreti, si parte dalla più grande (ovvero dal prefix con valore minore), ovvero:

- | | |
|---------------|---|
| 10.0.0.0/24 | # tutta la rete |
| 10.0.0.0/25 | # subnet2 → $32-25=7 \rightarrow 2^7=128$ indirizzi → ultimo indirizzo: 10.0.0.127 |
| 10.0.0.128/26 | # subnet3 → $32-26=6 \rightarrow 2^6=64$ indirizzi → ultimo indirizzo: 10.0.0.191 |
| 10.0.0.192/30 | # subnet4 → $32-30=2 \rightarrow 2^2=4$ indirizzi, ma solo 1 usabile (quindi punto-punto, infatti <u>2 sono per indirizzo di rete e indirizzo broadcast</u>) |

⚠ Bit per la maschera = $256 - 2^{n_{\text{zeri_rimanenti}}}$; Netmask = $256 - 2^{\text{bit}}$

⚠ Quando bisogna verificare la validità di un indirizzo con prefix_length che supera 8, il controllo è da fare sul gruppo precedente, ovvero $2^{(32-x)-8}$ con l'operazione di modulo.

Esercizio (trovare l'errore nella configurazione di rete): si può notare come il problema sia che il prefix della rete arancione è /28 con quindi indirizzi da .1 a .14 (in quanto lo .0 è per la rete e il .15 è per il broadcast), quindi il .23 non è presente (si dovrebbe avere uno /27 oppure cambiare l'indirizzo del DNS da .23 a .10 per esempio).



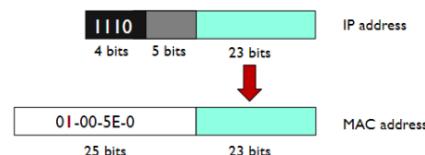
L'ALBERO DI INSTRADAMENTO stabilisce, a partire da un router della rete, i percorsi minimi per raggiungere tutti i nodi (salvando destinazione, next hop e numero di nodi toccati). Il piano di indirizzamento si fa sempre partendo dalla sottorete più grande, ovvero con prefix minore [esempio, se una rete ha 120 host, ha bisogno di $120+3$ indirizzi, quindi $123 < 128 = 2^7$ perciò avrà prefix_length di $(32-7) = /25$].

⚠ In un pc windows, il ping viene eseguito 4 volte: nella comunicazione tra 2 macchine (A e B) nella stessa rete avviene ARP request ($A \rightarrow$ broadcast), ARP response ($B \rightarrow A$), ICMP request ($A \rightarrow B$) e ICMP response ($B \rightarrow A$); il ping (ovvero i passaggi di ICMP request e response) viene eseguito 4 volte.

⚠ Dato un indirizzo di rete (es. 130.192.16.0/23), per calcolare il suo indirizzo di broadcast si fa così: $32 - /23 = 9$, quindi 9 bit a 1, ovvero 130.192.17.255. La sottorette che finisce con .36/29 (sbagliata perché gli ultimi $32-29 = 3$ bit devono essere a 0, invece 36 non ha gli ultimi 3 bit 100), è compresa nella rete .32/29 (multiplo di 8 [$32-29 = 3 \rightarrow 2^3 = 8$] più grande possibile, ma più piccolo di 36, quindi 32) [il broadcast della rete è 39, ovvero quello appena prima del successivo multiplo di 8 [40]].

Riguardo al **MULTICAST**, i pacchetti vengono indirizzati da una sorgente verso più destinazioni, selezionando però **solo alcuni host** (non tutti come il broadcast), cioè ci sono dei gruppi in cui gli host possono entrare/uscire. Viene implementato in IPv4 mediante **IGMP** (protocollo che permette ad un router IPv4 di scoprire quali gruppi multicast sono presenti in una rete ad esso direttamente connessa); il multicast si usa maggiormente in **IPv6**. Un host deve richiedere le informazioni di un gruppo multicast direttamente ai router.

Ogni gruppo multicast ha un indirizzo IPv4 di **classe D**; questo perché, seppur il multicast (group delivery) viene eseguito a livello 2, si può associare un indirizzo di livello 2 ad uno di livello 3 attraverso un **MAPPING IP-MAC**: il MAC (livello 2) è formato da 48 bit di cui la parte alta, solitamente riservata al produttore, ha invece qui costante 01-00-5E-0 (25 bit, identifica la mappatura [l'ottavo bit è sempre a 1 in multicast per Little Endian]), mentre la parte bassa è formata dai 23 bit meno significativi dell'indirizzo IP (livello 3).



I router trovano i dispositivi su ciascuna LAN mediante il protocollo **IGMP**, formando tra i router un albero di distribuzione per ogni gruppo verso tutte le LAN.

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

⚠ IPv4 non è in grado di rispondere alle esigenze di *control engineering* e *traffic engineering*; oggi però **IPv4 è ancora largamente utilizzato**.

1) IPv6

IPv6 ha un **maggior numero di indirizzi**; è più **efficiente** sulle LAN, supporta **Multicast** e **Anycast**, è **sicuro**, **plug and play**, ha **traffic differentiation**, **mobilità** e **QoS** (Quality of Service).

Gli indirizzi IP vengono assegnati dalla **IANA** che fornisce a ciascun **RIR** (Regional Internet Registry) un blocco di indirizzi IP con prefix /8; le RIR dividono i blocchi in blocchetti più piccoli da assegnare alle NIR (National IR) e alle LIR (Local IR).

⚠ Ogni indirizzo IPv4 può essere in 1 dei seguenti stati:

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

Ciò comporta **problemi di scalabilità** dovuti alla dimensione delle routing table, alle risorse limitate dei router e alle limitazioni dei protocolli di routing (la scalabilità dei protocolli di routing risulta attualmente **non risolvibile**).

Gli indirizzi IPv6 hanno lunghezza pari a **128 bit** (quindi **2¹²⁸ indirizzi**); non si usa più il punto, ma si usa una notazione con gruppi di **2 Byte (4 cifre esadecimale)** separati da **:**. Questa notazione si può comprimere:

- rimuovendo i gruppi pari a 0000 comprimendoli in 0, oppure rimuovendo gli zeri iniziali dei gruppi (es. 1080:0000:0000:0007:0200:A00C:3423:A089 in 1080:0:0:7:200:A00C:3423:A089);
- omettendo gruppi di soli zeri rimpiazzandoli con 1 solo **::** (es. da prima 1080::7:200:A00C:3423:A089).

Riguardo al **ROUTING** non viene modificata la struttura usata in IPv4 ad eccezione della lunghezza degli indirizzi; non è più necessario usare le classi. Il **prefix_length**, per quanto attualmente sia fisso a 64, è pensato per essere anche < 64 in maniera **flessibile** (**mai > 64** però). Cambiano solo alcune cose nella terminologia:

- **Link** = physical network
- **Subnetwork** = link, set di host con lo stesso prefix
- **On-link** = gli host hanno lo stesso prefix (comunicazione diretta)
- **Off-link** = gli host hanno diverso prefix (comunicazione attraverso router)

Riguardo al **MULTICAST**, l'equivalente IPv6 dell'indirizzo multicast IPv4 224.0.0.0/4 è **FF00::/8** che si divide in:

- **Well-known Multicast** → **FF00::/12**, riservati per le comunicazioni di servizio
- **Transient** → **FF10::/12**, transitori
- **Solicited-node Multicast** → **FF02::1:FF00::/104**, simile ad un broadcast in ARP

L'indirizzo **IPv6 multicast** è composto da:

- 8 bit iniziali → identificano che è **multicast** (tutti a 1) [quindi tutti gli indirizzi con inizio **FF**: sono multicast]
- 4 bit → **T flag**, specifica se well-known o transient
- 4 bit → **scope**
- 112 bit → **group ID**

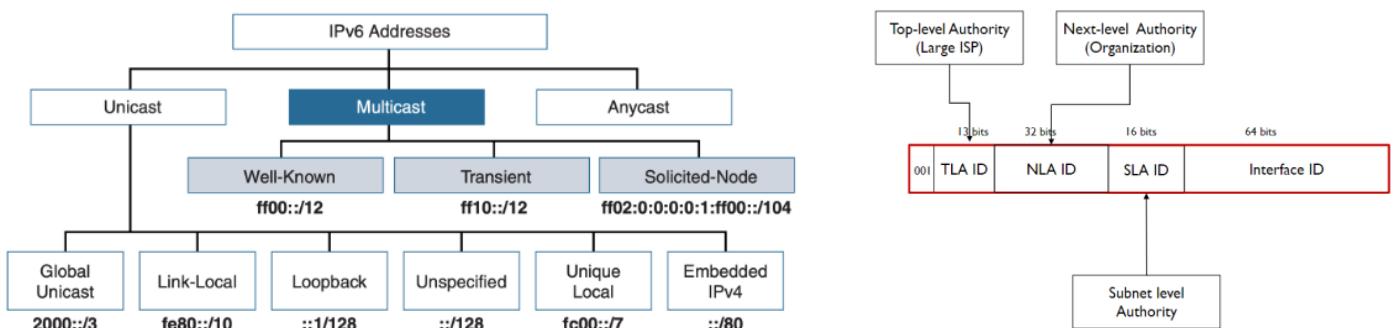
⚠ In IPv6 scompare il **BROADCAST** per questioni di sicurezza.

L'**UNICAST** rimane disponibile in IPv6, con indirizzi:

- **2000::/3** → **Global Unicast**
- **FE80::/10** → **Link-Local** = assegnati in automatico ai link all'accensione di un router:
 - **Link local** (FE80::/64) → assegnati quando più router devono parlare tra loro
 - **Site local** (FEC0::/10) → deprecati
 - ⚠ Quando un dispositivo si accende, prenderà in automatico un indirizzo link local dipendente dal MAC della propria scheda di rete.
- **::1/128** → **Loopback** (in IPv4 ricorda che era 127.0.0.1), con finalità di test (inviare un pacchetto a sé stesso)
- **FC00::/7** → **Unique Local Addresses (ULA)** = usati in modo simile agli indirizzi Global Unicast, ma sono per uso privato e non indirizzamento su internet (quindi come gli indirizzi privati su IPv4, ovvero 192.168.x.x o 10.x.x.x). L'ottavo bit è detto **Local Flag** (o **L flag**) e differenzia questi indirizzi in:
 - **FC00::/8** → L flag = 0, ovvero assegnato in futuro
 - **FD00::/8** → L flag = 1, ovvero assegnato localmente (**gli unici attualmente validi**)
- **::80** → **Embedded IPv4**, usato per rappresentare indirizzi IPv4 dentro indirizzi IPv6 (l'indirizzo IPv4 è inserito negli ultimi 32 LSB bit, preceduti da 80 bit a 0 e 16 bit FFFF)

⚠ Un pacchetto unicast **non specificato** contiene solo 0 (usato come sorgente per indicare l'assenza di indirizzo, una sorta di placeholder).

Gli indirizzi unicast sono di tipo aggregato, sono raggiungibili e indirizzabili globalmente, oltre ad essere plug & play. Sono attualmente disponibili in un range tra 3FFF:: e 2000:: (tutti questi indirizzi hanno quindi i primi 3 bit [MSB] posti a 001). Sono indirizzi **distribuiti geograficamente globalmente in modo gerarchico**: i prefissi per il Global Routing ([Global Routing Prefix](#)) sono assegnati da multi-level authorities e hanno struttura:



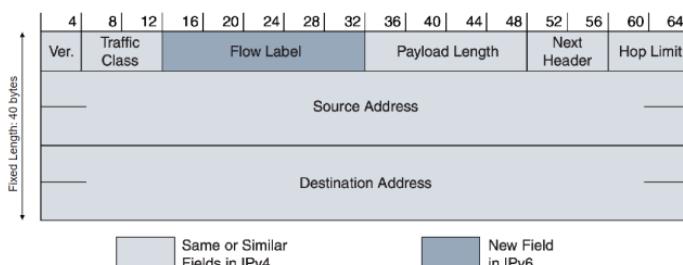
Riguardo l'**ANYCAST**, sono indirizzi assegnati a più di un'interfaccia con la possibilità di avere lo stesso indirizzo anycast su dispositivi diversi; un pacchetto indirizzo ad un indirizzo anycast viene inviato all'interfaccia più vicina con quell'indirizzo (l'anycast però **non è ancora usato**).

I **protocolli in IPv6** sono simili a quelli usati in IPv4 ma con alcune differenze:

- IP → alcune modifiche
- ICMP, ARP e IGMP → inglobati in **ICMPv6** (non è più possibile usare ARP e IGMP per risolvere indirizzi IPv6)

L'**HEADER IPv6** ha una lunghezza di **40 Byte** (320 bit); rispetto ad IPv4 sono stati rimossi l'**header checksum** (perché i controlli di errore vengono già eseguiti a livello 2) e la **frammentazione** (i pacchetti più grandi di MTU vengono scartati, restituendo alla sorgente il messaggio d'errore "ICMPv6 Packet Too big").

⚠ Il checksum su UDP diventa opzionale in IPv6.



I singoli campi sono:

- **version**
- **traffic class** (priorità del traffico, QoS)
- **flow label** (etichettare un certo tipo di traffico, es. datore di lavoro che controlla il traffico dei dipendenti)
- **payload length** (lunghezza dei dati)
- **hop limit** (n° router attraversabili prima che il pacchetto venga scartato; se = 1, pacchetto inviato direttamente al destinatario; se = 255, pacchetto non viene mai scartato)
- **next header** = consente di puntare ad un altro header con ulteriori informazioni per estendere l'header (creando una catena di header, "chaining"); composto da:
 - **next header** [tipo di header successivo]
 - **Hop-by-Hop Extension Header** → inserire dei vincoli per far capire all'hop se il pacchetto vada scartato o meno; se presente, viene inserito subito dopo l'header IPv6; viene anche usato per inserire campi opzionali (optional TLV [Type-Length-Value])
 - **Routing Extension Header** → permette alla sorgente di un pacchetto di specificare il percorso di destinazione, indicando 1 o più router intermedi
 - **Fragmentation Header** → usato per la frammentazione dei pacchetti se serve
 - **Authentication and Encapsulation Header** → usati per la sicurezza da IPsec (sicurezza in IP)
 - **length**
 - **extension header** [l'effettivo header successivo]
 - **extension data** [dati dell'header successivo]

Ora parliamo del modo in cui IPv6 si interfaccia con i livelli più bassi:

- **INCAPSULAMENTO** → previsto un campo dove viene specificato il contenuto del livello superiore (approccio "dual stack"). I pacchetti IPv6 sono incapsulati nel frame (trama) di livello 2. Un dispositivo con un'interfaccia IPv6 e IPv4 può ricevere sia pacchetti IPv6 sia pacchetti IPv4.
- **ADDRESS MAPPING** → un indirizzo di un pacchetto IPv6 viene associato ad un MAC di destinazione usando:
- **IP unicast address (neighbor discovery)** → ICMPv6 sostituisce ARP, è basato su **multicast** e sfrutta il **Solicited-Node Multicast Address**: tutti gli host si iscrivono e vengono mappati in indirizzi del tipo FF:02::1:FF:xx:xx/104 (dove xx:xx sono gli ultimi 24 bit LSB dell'indirizzo unicast IP, in modo da ottenere 1 host per gruppo); questo indirizzo viene usato come indirizzo di destinazione di un pacchetto di Neighbor Solicitation.

La risoluzione di un indirizzo (**Address Resolution**) avviene attraverso ICMP Neighbor Solicitation:

1. Il richiedente invia un frame al Solicited-Node Multicast Address (contenente l'indirizzo IPv6 dell'host target come abbiamo visto prima)
2. Viene inviata dal target una risposta ICMP Neighbor Advertisement, con cui viene mandata la risposta all'indirizzo unicast del richiedente). La mappatura tra IPv6 e MAC viene memorizzata nella cache dell'host (come avveniva per ARP)

⚠ A causa della mancanza del broadcast in IPv6, il numero di MAC aumenta molto!

- **IP multicast address (mapping)** → si basa sull'ethernet multicast (ricorda che a differenza del broadcast può essere filtrato dalla scheda di rete [NIC]). Per il trasporto di pacchetti multicast IPv6, viene riservato l'indirizzo composto MAC composto da **33-33-xx-xx-xx-xx** (dove xx-xx-xx-xx sono gli ultimi 4 Byte [32 bit] LSB dell'indirizzo IPv6) [es. **FF0C::89:AABB:CCDD** viene incapsulato in **33:33:AA:BB:CC:DD**].

⚠ L'inoltro dei pacchetti IPv6 su una LAN non usa meccanismi di neighbor discovery proprio per questa regola di mapping in MAC.

⚠ La **transizione da IPv4 a IPv6** sta avvenendo in modo **incrementale, seamless** (cioè senza cambiamenti) e **dual stack**, usando 3 meccanismi: **Address Mapping** (visto sopra), **Tunneling** e **Translation Mechanisms**.

Parliamo ora di **ICMPv6**, che fa diagnostica, neighbor discovery, multicast group management e issue notification, includendo però anche operazioni che in IPv4 facevano **ARP** e **IGMP**. L'**HEADER ICMPv6** ha al max 576 Byte, con struttura base (che vedremo ampliata poi nei vari casi) del tipo [a dx vediamo i valori che può assumere **Code**]:

	Code	Spiegazione	tipo			
8	8	16				
Type	Code	Checksum				
Message Body						
1	Destination Unreachable	Errore	131	Multicast Listener Report	Informativo	
2	Packet too big	Errore	132	Multicast Listener Done	Informativo	
3	Time exceeded	Errore	133	Router Solicitation	Informativo	
4	Parameter Problem	Errore	134	Router Advertisement	Informativo	
128	Echo Request	Informativo	135	Neighbor Solicitation	Informativo	
129	Echo Reply	Informativo	136	Neighbor Advertisement	Informativo	
130	Multicast Listener Query	Informativo	137	Redirect	Informativo	

	8	8	16
Type	Code	Checksum	
Identifier		Sequence Number	
Data			

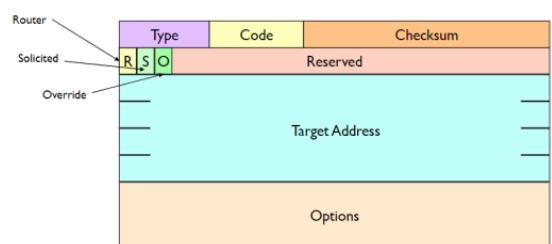
⚠ Echo request e reply vengono usate da ping e hanno struttura →

	8	8	16
Type	Code	Checksum	
Identifier		Sequence Number	
Data			

⚠ La **Neighbor Advertisement** ha la stessa struttura della Neighbor Solicitation, ma si aggiungono dei **flag**:

- **R** (router flag) → true se arriva da un router
- **S** (solicited flag) → se arriva da un nodo che ha fatto richiesta di risoluzione
- **O** (override flag) → se la host cache deve essere aggiornata o meno

⚠ Quando si ha un collegamento che fa affidamento al data link layer multicasting services, è necessario che ICMPv6 sappia i membri on-link (ovvero gli host interessati a ricevere i pacchetti).



Sempre parlando dell'header ICMPv6, per scoprire gli host si fa:

- **Multicast listener query** → router manda query per capire se host è interessato a ricevere i pacchetti multicast
- **Multicast listener report** → host risponde al router dicendo che è interessato a ricevere i pacchetti multicast
- **Multicast listener done** → host manda un messaggio di fine per avvisare che non è più interessato

⚠ Oggi la gestione del multicast viene rappresentata solo a livello 3 (quindi diviene solo compito del router e non dello switch).

Parlando invece del **DEVICE CONFIGURATION** (configurazione del prefisso di un indirizzo) in IPv6 si ha:

- **Manual Configuration**
- **Stateful Configuration** (informazioni recuperate mediante DHCPv6)
- **Stateless Configuration** (generate automaticamente, con il prefisso dell'indirizzo ottenuto dal router) = permette di non usare il protocollo DHCP per la configurazione automatica dei dispositivi; ha come fasi:
 - Generazione di un indirizzo link local
 - Verifica dell'unicità dell'indirizzo (**DAD** = Duplicate Address Detection)
 - Il dispositivo si pone in ascolto di un messaggio di router advertisement o manda una solicitation per scoprire le informazioni sull'indirizzo privato (attraverso la **Router/Prefix Discovery**, se l'host non ha mandato una solicitation, potrebbe essere direttamente il router a mandare l'informazione)

Una volta scoperta la parte alta dell'indirizzo:

- si verifica se anche nella sottorete l'indirizzo sia univoco
- avviene l'iscrizione al corrispondente IPv6 Solicited Node Multicast Address
- la comunicazione on-link viene abilitata

Un vantaggio è quello del "renumbering" (tramite l'advertisement vengono riconfigurati tutti i dispositivi in modo automatico).

- **Hybrid (Stateless DHCP)**, ulteriori informazioni oltre l'indirizzo recuperate mediante DHCPv6)

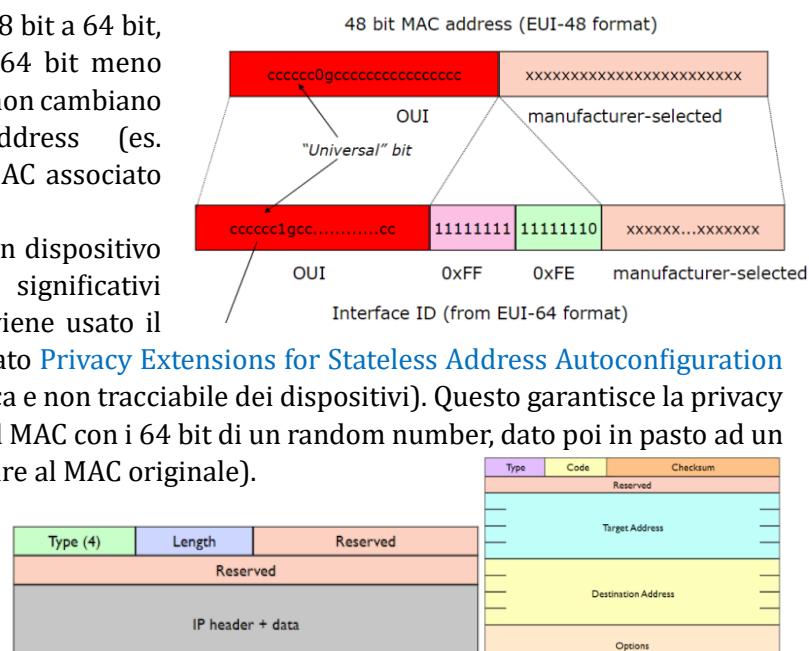
⚠ Una **ROUTER SOLICITATION** viene mandata solamente ai router (all routers, FF01::2). Una **ROUTER ADVERTISEMENT** è composta da:

- **M flag** = se messo a 1, significa che l'indirizzo è stato configurato tramite DHCPv6
- **O flag** = se messo a 1, significa che sono presenti altre configurazioni (es. DNS server)
- **reachable time** = millisecondi che il router impiega per raggiungere un host
- **retrans timer** = tempo per cui ritenere l'indirizzo valido
- **option**:
 - **lifetime** (tempo di vita dell'indirizzo)
 - **preferred lifetime** (tempo oltre cui è meglio non usarlo più)
 - **L** (se usato in un on-link)
 - **A** (prefisso usabile per configurazione automatica)
 - **prefix** (il prefisso)

EUI-48 to EIU-64 estende un indirizzo MAC da 48 bit a 64 bit, inserendo 11111110 (FE) e 11111111 (FF). I 64 bit meno significativi di un indirizzo IPv6 di un'interfaccia non cambiano mai quando viene usato un MAC address (es. FE80::0201:06FF:FEA5:3A4C può avere come MAC associato 00:01:06:A5:3A:4C).

Proprio per questo motivo, riuscire a tracciare un dispositivo non è complicato in quanto i 64 bit meno significativi dell'indirizzo IPv6 non cambiano mai quando viene usato il MAC address; proprio per ciò è stato implementato **Privacy Extensions for Stateless Address Autoconfiguration in IPv6** (consentire una configurazione automatica e non tracciabile dei dispositivi). Questo garantisce la privacy al livello 3 (rete); funziona così: unisce i 64 bit del MAC con i 64 bit di un random number, dato poi in pasto ad un algoritmo di hashing (in modo da non poter risalire al MAC originale).

ICMP Redirect viene usato per informare, nella stessa sottorete, un host A che per raggiungere un determinato host B è più conveniente usare un altro router [dx header e message format].



Parlando di **SCOPED ADDRESS**: un dispositivo può avere più interfacce con lo stesso indirizzo, perciò un pacchetto viene mandato su un'interfaccia piuttosto che un'altra in base allo “scopo” (tipo di programma che lo ha generato). Un indirizzo *scoped* è composto da un indirizzo IPv6 seguito da % e un numero identificativo dell'interfaccia (es. FE80::0237:00FF:FE02:A7FB%19) [oggi **non più usato** il byte di scope].

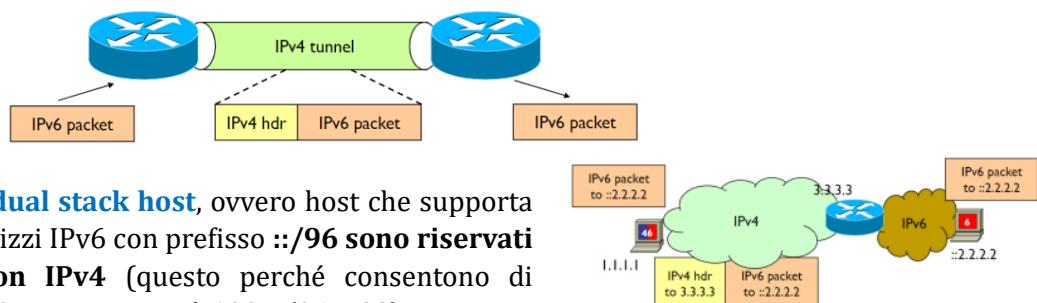
Il **ROUTING** si differenzia in:

- **On the fly routing (FORWARDING)** = **usa le routing tables** e permette di determinare qual è la migliore porta di uscita verso la destinazione
- **Proactive routing (ROUTING)** = **crea le routing tables** (al fine di individuare i percorsi per i pacchetti) [se le tabelle vengono create manualmente, si parla di “**static routing**”]

Le routing tables in IPv6 sono basate sul “**longest prefix match**” (come in IPv4). Tipi di **PROTOCOLLI** di routing:

- **Integrated routing** → 1 protocollo di routing per entrambe le “*protocol families*” (sia IPv4 sia IPv6)
- **Ships in the night** → ogni family address ha il suo protocollo di routing (protocolli indipendenti tra loro)

La **TRANSIZIONE DA IPv4 a IPv6** ha visto all'inizio un approccio “**dual stack**” [1] (supporto per entrambi), ma poi è prevalso il “**tunneling**” [2] (incapsulare un pacchetto IPv6 in uno IPv4, al fine di emulare il link diretto tra dispositivi IPv6, ma in una infrastruttura IPv4).



1. Host centered solutions (dual stack host), ovvero host che supporta sia IPv4 sia IPv6:

gli indirizzi IPv6 con prefisso **::/96 sono riservati per la compatibilità con IPv4** (questo perché consentono di mantenere liberi i 32 bit LSB per IPv4, cioè $128 - /96 = 32$)

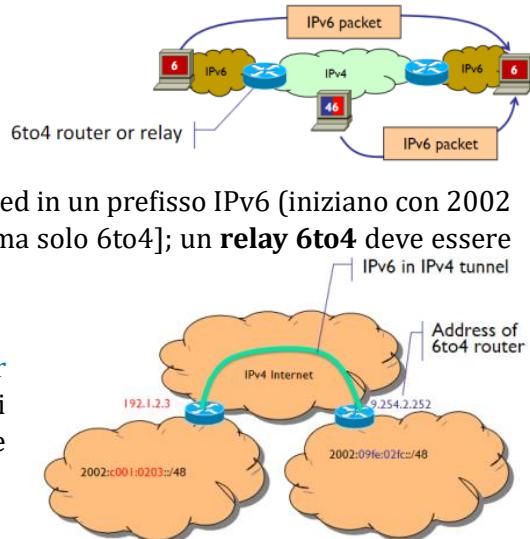
⚠ Il protocollo **6over4** usa una rete IPv4 per emulare una VLAN; l'indirizzo IPv4 è usato per la generazione automatica di un interface ID IPv6 dell'indirizzo link local (poco usato per il poco supporto a multicast IPv4).

⚠ **ISATAP** si differenzia: al posto di usare il multicast, usa una soluzione con un prefisso di rete 0000:5FE (rispetto 6over4 non serve IPv4 multicast); usa **DNS**, ma ha come limitazione che ogni indirizzo deve avere associato un *hostname* (infatti è proprio l'hostname che fa la richiesta, e non l'indirizzo IPv6). Non ha bisogno di fare data-link address discovery perché l'indirizzo IPv4 è incluso nell'indirizzo IPv6 (negli ultimi 4 Byte).

⚠ **Configurazione automatica**: vengono usati indirizzi IPv4 e indirizzi DNS; il nome del dominio è ottenuto con DHCPv4. L'indirizzo IPv6 link local viene generato automaticamente, mentre l'interface ID è ottenuto dall'indirizzo IPv4. Per ottenere la “potential router list” si usa una query DNS (a meno che non sia fornita da DHCPv4); inoltre, si esegue periodicamente una router discovery verso tutti i router su link prefixed per l'autoconfigurazione.

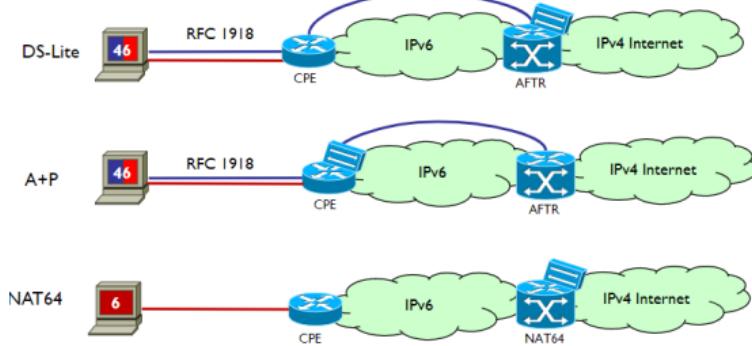
2. Network centered solutions (tunneling): si configurano intere reti **IPv6 dentro una struttura IPv4** (dovendo quindi rinunciare a delle funzionalità IPv6 ed avendo un range di indirizzi ridotto).

⚠ Attraverso il protocollo **6to4**, gli indirizzi dei relay sono embeddi in un prefisso IPv6 (iniziano con 2002 e sono indirizzi pubblici) [non è pensato per comunicazioni 4to6, ma solo 6to4]; un **relay 6to4** deve essere per forza il **default gateway** per i router 6to4.



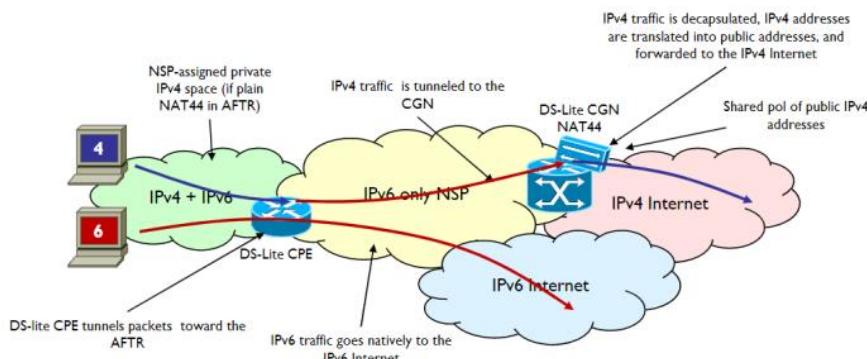
⚠ Le comunicazioni avvengono mediante un **tunnel broker server** (trova i tunnel server e fa da mediatore tra le configurazioni dei tunnel). Vengono usati tunnel IPv6 in IPv4; è una soluzione centralizzata.

3. **Scalable Carrier-grade solutions** (soluzioni **miste**): usate dai grandi provider; i server/client IPv4 possono comunicare con host IPv6 e IPv4 (**dual stack**), ma si usa comunque un **tunnel**. Si basano sul concetto di **mapping tra un indirizzo IPv4 e IPv6**, associando una porta ad un indirizzo privato (stiamo parlando del **NAT**, Network Address Translation; più in particolare del **LSN [Large Scale NAT]**) [si possono avere più livelli di NAT in cascata]; il NAT però comporta un single point of failure. La posizione del NAT varia nei diversi protocolli usati:

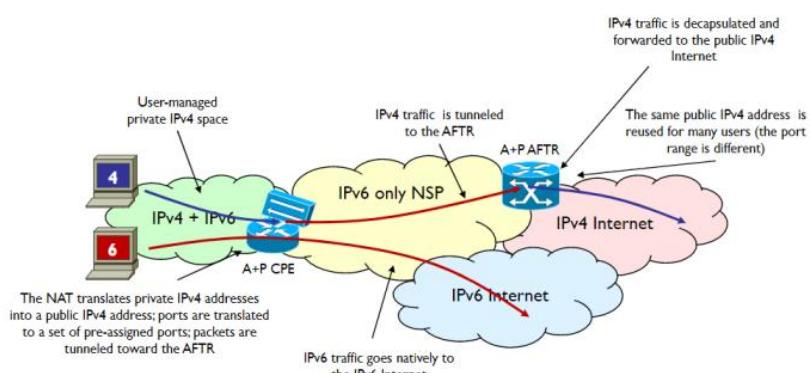


L'uso di **AFTR** (Address Family Transition Router) abilita gli host IPv4 a comunicare con altri host IPv4 usando una rete IPv6 (può quindi connettere strutture IPv6 con infrastruttura IPv4); ha 2 funzioni: **NAT** e **HW per il tunneling**. Viene usato da DS-Lite e da A+P:

- **DS-Lite (Dual-Stack Lite)** → protocollo caratterizzato da ISP (Internet Service Provider) che usano una “**backbone**” (infrastruttura di rete) **IPv6**. Ciò consente di avere sottoreti IPv4 o IPv6. È una soluzione **complessa**: da IPv4 privato si arriva sull'home gateway (il CPE) che effettua tunnel verso l'AFTR mediante una rete IPv6 su cui è installato un LSN (Large Scale NAT). Il **NAT è sull'AFTR** (guarda immagine sopra). Ha limitazioni: single point of failure (unico NAT per tutti i clienti), il cliente non ha controllo sul NAT e non si possono usare static mapping e port forwarding.



- **A+P (Address plus Port)** → possibilità per il **cliente di avere sotto controllo il NAT** (sposto la complessità sulle foglie). Il range di TCP/UDP è assegnato a ciascun customer. Questa soluzione parte da un indirizzo IPv4 privato che arriva sul CPE e viene convertito in indirizzo pubblico, e solo dopo viene fatto 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. Il **NAT è nel CPE**.



Il **MAP (Mapping Address & Port)** usa invece un approccio “**stateless**”, non associando dei range di porte ma dei set di porte (**Port Set** = insieme di porte non necessariamente contiguo), dove ad ogni CPE viene associato un indirizzo pubblico IPv4 (usa infatti la stessa rete pubblica per non avere limitazioni) e un **PSID** (Port Set ID, identifica un set di porte). Il PSID ha 16 bit, divisi in campi:

- A (dominio)
- PSID (set di porte)
- J (insieme di porte)

⚠ Non porre i primi bit a zero, altrimenti diventa una “well-known port”.

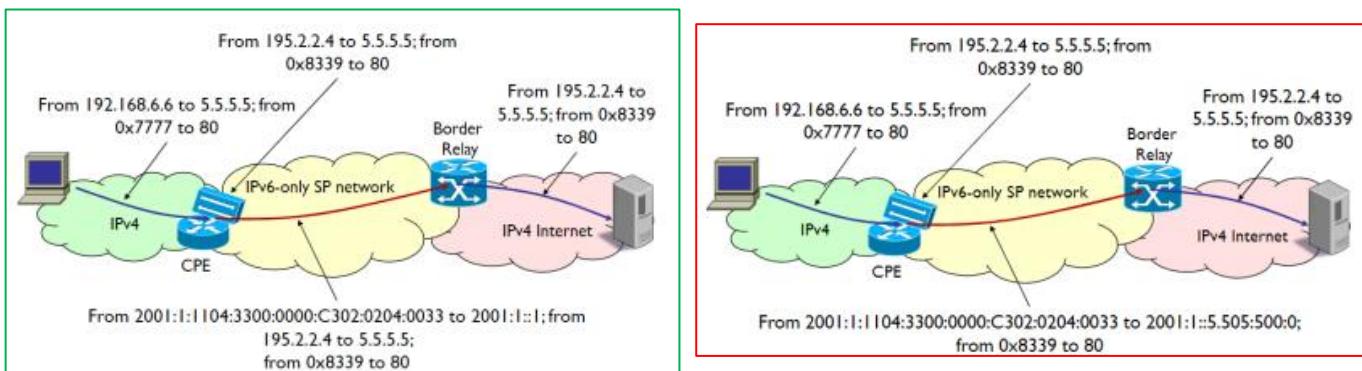
⚠ L'**Embedded Address** (EA) contiene i bit di PSID e parzialmente l'indirizzo IPv4 (che identificano univocamente il CPE)

Il MAP **funziona così**: inserire all'interno del pacchetto l'informazione di stato (per diminuire quella da mettere nel Border Relay) e **mappare queste informazioni negli indirizzi IPv6 del CPE**. 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 da internet). L'indirizzo e la porta del client IPv4 sono mappati in un unico indirizzo IPv6 (prefix routed dal CPE). L'indirizzo del server pubblico IPv4 è mappato in un unico indirizzo IPv6 (prefix router dal Border Relay). 2 tipi di MAP:

- **MAP-E** (with Encapsulation) = pacchetti IPv4 vengono tunnelizzati
- **MAP-T** (with Translation) = header dei pacchetti IPv4 tradotto in header IPv6 (e poi ritradotto in IPv4)

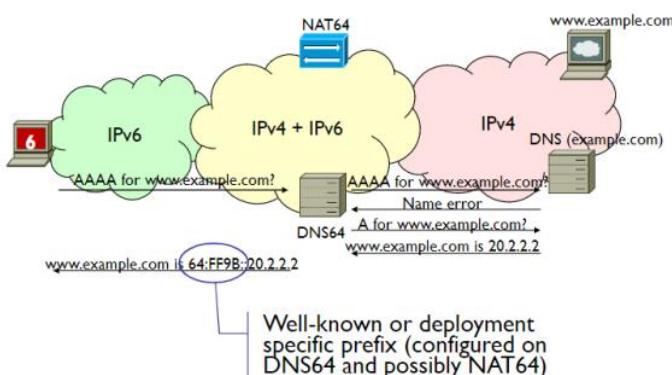
Le **regole di mapping** sono IPv6 prefix rule, IPv4 prefix rule e EA bits length. Tramite queste informazioni il CPE si calcola l'indirizzo IPv6 da usare nell'interfaccia esterna.

⚠ L'indirizzo del **Border Relay** (BR) deve essere conosciuto da tutti i CPE, anche se più BR possono avere lo stesso indirizzo (anycasting). Inoltre, mentre nel **MAP-E** il BR termina il tunnel, nel **MAP-T** il BR si occupa della traduzione degli header.



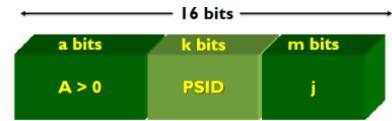
Il **NAT64**, usando il NAT, **traduce indirizzi e pacchetti IPv6 in IPv4**. Risolve il problema della rete IPv6 (con host IPv6) che deve comunicare con la rete pubblica IPv4. Il **DNS64** rimuove l'header IPv6 per inserire l'header IPv4 (per questo **NAT64+DNS64**). Ha i seguenti passi (“**name resolution**”) [sotto img]:

- A. Host IPv6 esegue query DNS di tipo IPv6 AAAA
- B. DNS64 inoltra la query dal DNS autoritativo verso il dominio della richiesta
- C. DNS autoritativo (che si trova in una rete IPv4) non può ricevere la richiesta, e per questo motivo viene ripetuta per un indirizzo IPv4 (necessaria quindi rete IPv4-IPv6 intermedia)
- D. DNS64 riceve l'indirizzo IPv4 che risolve il dominio, ne fa l'embedding in un indirizzo IPv6 con un prefix di default 64:FF9B
- E. L'indirizzo IPv6 viene restituito all'host che l'ha richiesto



⚠ Gli svantaggi sono **dover usare il DNS** (per l'hostname) e **non si può usare DNSSEC** (la firma della risposta a un record DNS) perché DNS64 modifica i record.

⚠ Si possono risolvere indirizzi solo quando a questi sono associati dei nomi.



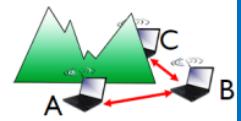
2) RETI WIRELESS e CELLULARI

Le **RETI WIRELESS** permettono la comunicazione tra dispositivi senza un cavo fisico. Sono composte da:

- **Wireless host** = trasmettono e ricevono dati; possono essere stazionari o mobili
- **Base station ("Access Point", AP)** = inviano pacchetti tra rete cablata e rete wireless
- **Wireless link** = collegamento tra host e base station; accesso controllato da protocolli ad accesso multiplo

La **RETE CELLULARE** può essere gestita mediante un'infrastruttura in cui le base stations connettono i wireless host alla rete cablata (con l'**handoff** i wireless host cambiano base station) oppure può essere gestita **ad hoc** (i wireless host si connettono direttamente tra loro senza usare una base station).

⚠ Gli **svantaggi** rispetto ad un link cablato sono **degrado del segnale**, **interferenza** tra dispositivi, multipath propagation ("fading", effetto dovuto ai rimbalzi del segnale sugli ostacoli) e **comunicazioni più complesse**. È inoltre presente il problema del **TERMINALE NASCOSTO**: dati 3 nodi (a,b,c) se b comunica con entrambi i rimanenti, questi potrebbero non sapere della reciproca presenza e generare interferenze.



⚠ Definito **SNR** (Signal to Noise Ration) come relazione tra il segnale ricevuto e il rumore [quindi indice della qualità del segnale], dato un livello fisico, aumentarne l'alimentazione comporta un aumento di SNR e una riduzione del **BER** (Bit Error Ratio). Il valore di SNR può variare a causa della mobilità, adattandosi dinamicamente al livello fisico.

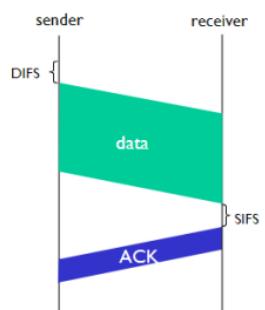
2.1) Wireless LAN (802.11)

Un **BSS** (Basic Service Set, ovvero una cella) in **modalità infrastruttura** contiene un host wireless e una base station, mentre in **modalità ad hoc** solo l'host. **Ogni host che vuole connettersi** esegue prima una scansione delle reti ("scanning") e poi rimane in attesa di un "beacon frame" (contenente SSID [nome dell'access point] e MAC). Il dispositivo si conserverà al beacon fram più forte (per aumentare la qualità della connessione). 2 scanning:

- **Passive scanning** (beacon frame **invia dagli AP** [base stations] **all'host**):
 - 1) Beacon frame inviato dagli AP all'host
 - 2) Host manda association request all'AP scelto
 - 3) AP conferma associazione mandando un association response all'host
- **Active scanning** (beacon frame **richiesto dall'host all'AP**)
 - 1) Probe request (dall'host)
 - 2) Probe response (dagli AP)
 - 3) Association request (dall'host all'AP scelto)
 - 4) Association response (dall'AP all'host)

L'accesso di più dispositivi su 1 canale wireless richiede **CSMA** per eliminare le collisioni tra 2 o più nodi che trasmettono insieme. In Ethernet si usa **CSMA/CD** (Collision Detection), in Wireless si usa **CSMA/CA** (Collision Avoidance) che vuole evitare le collisioni con la trasmissione già in corso di altri nodi. **Funzionamento CSMA/CA**:

- **TX** (dispositivo che invia):
 - Se il canale è **libero** ("in idle") per un tempo "**DIFS**", inizia a trasmettere
 - Se il canale è **occupato**, viene avviato un **random backoff time** che lo pone in attesa il TX prima del nuovo tentativo. Se anche al nuovo tentativo è occupato, il TX ripete il processo aumentando il backoff
- **RX** (dispositivo che riceve):
 - Se il frame è ricevuto correttamente, viene inviato un ACK frame dopo un tempo "**SIFS**" (necessario per evitare il problema del terminale nascosto prima citato).



⚠ Il Collision Avoidance non è però **deterministico**: per renderlo tale, bisogna usare un sistema per riservare il canale usando dei pacchetti di "prenotazione" (**RTS** = Ready To Send [invia dal TX]; **CTS** = Clear To Send [invia dal RX che ha ricevuto il pacchetto RTS verso tutti i dispositivi in ascolto, in modo da far partire la trasmissione dal TX e in modo che tutti gli altri siano in attesa]).

Il **frame (trama)** scambiato contiene:

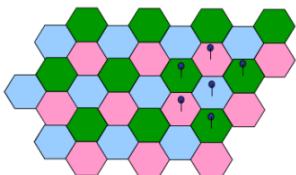
- ✓ Frame control = contenente:
 - Protocol version
 - Tipo (RTS, CTS, ACK, dati)
 - Bit per power management
- ✓ Duration
- ✓ Address 1 = MAC address del RX
- ✓ Address 2 = MAC address del TX
- ✓ Address 3 = MAC address dell'interfaccia del router a cui l'AP è connesso
- ✓ Seq control (per gli ACK)
- ✓ Address 4 (usato solo in modalità ad hoc, dove troviamo solo l'host)
- ✓ Payload
- ✓ CRC = controllo di errore

⚠ Dal punto di vista energetico, esiste il **node-to-AP**, con cui l'AP viene a sapere che non deve inoltrare i frame al nodo, il quale si sveglierà prima del prossimo beacon frame.

2.2) Reti cellulari

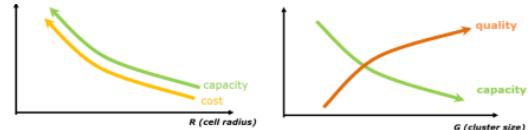
Le **RETI CELLULARI** sono reti wireless che coprono aree geografiche molto vaste usando **CELLE** (zone) adiacenti. Le celle si dividono in **macrocelle** e **microcelle** in base alle loro dimensioni (copertura). Anche qui c'è il problema di accesso multiplo condiviso sul canale, ma viene risolto con la **MULTIPLAZIONE**:

- **FDMA** (frequenza)
- **TDMA** (slot temporale [tempo])
- **CDMA** (codice ortogonale [gruppo di segnali da cui posso recuperare i singoli segnali])
- **SDMA** (spazio [distanza], ovvero posso riusare frequenze se stazioni lontane)



Un gruppo di celle si dice **CLUSTER** (vedi la figura sopra a dx); le celle dello stesso colore si dicono “**co-channel cells**” (ovvero hanno lo stesso set di canali). Il cluster ha 2 dimensioni:

- **R** (raggio delle celle) → se <, ho < n° utenti, > costo
- **G** (n° celle del cluster) → se >, ho > costo, > qualità, < capacità



⚠ Per diminuire le interferenze e aumentare la capacità si fa:

- **splitting** = non usare celle delle stesse dimensioni
- **sectoring** = usare antenne non omnidirezionali (solo direzioni richieste)
- **tilting** = non usare angolo a 90° per la trasmissione
- creazione di **femtocelle** (celle non fisse in base alle necessità)
- **shaping** = usare antenne direzionali per avere celle con forme ad hoc (es. “shaping” sulle autostrade) oppure usare una copertura multi-livello (“umbrella coverage”)

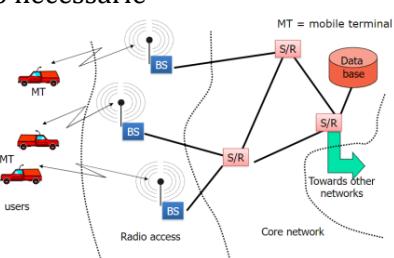
Riguardo al **Power Control** (gestire al meglio l'uso di potenza/energia in base alle necessità), 2 strategie:

- In UPLINK (terminale → ripetitore):
 - a catena **aperta** (“**open loop**”) = senza reazione; il sistema misura la qualità del segnale ricevuto (infatti il terminale si regola automaticamente sulla potenza di trasmissione)
 - a catena **chiusa** (“**close loop**”) = con reazione; il sistema riceve feedback sulla qualità del segnale
- In DLINK (ripetitore → terminale), si ha semplicemente “downlink power control”

Riguardo all'**allocazione delle frequenze**, questa può avvenire in vari modi:

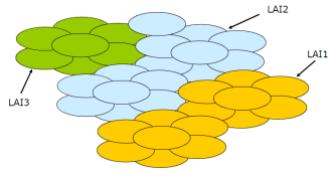
- **FCA** (Fixed Channel Allocation) = assegnate staticamente, modificate raramente
- **DCA** (Dynamic Channel Allocation) = assegnate da 1 controller centrale quando necessarie
- **HCS** (Hybrid Channel Allocation Scheme) = mista tra FCA e DCA

Parliamo ora di **ARCHITETTURA DELLA RETE CELLULARE**: le reti sono costituite da **MT** (Mobile Terminal) che si connettono alle **BS** (base station, AP) radio, le quali si connettono a dei **core network** usando degli switch (commutatori a pacchetto o circuito). I core network sono fatti da un set di server che si occupano di gestire le connessioni/risorse in modalità cablata (“wired”).



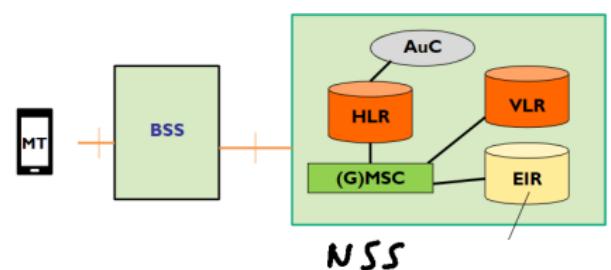
C'è una fase di **registrazione** (permette ad un terminale mobile di connettersi alla rete, che lo autentica).

Un'altra procedura è il **MOBILITY MANAGEMENT** (gestione della mobilità dell'utente), composto da:

1. **ROAMING** = capacità di un terminale di essere tracciabile quando si sposta nella rete; per salvare le posizioni, la rete viene divisa in **LA** (location areas, gruppi di celle adiacenti caratterizzati da un ID univoco)
2. **LOCATION UPDATING** = ogni volta che un utente si sposta verso un'altra LA
3. **PAGING** = il sistema notifica un MT di una chiamata o di un data delivery: il sistema manda la richiesta in broadcast a tutti gli MT della LA e il terminale che riceve la richiesta risponde con un messaggio di conferma
4. **HANDOVER** = abilità il trasferimento di una connessione attiva da una cella ad un'altra quando il MT si sposta nella rete. Si classifica in:
 - o **Intra vs Inter Cell** (nella stessa cella [INTRA] o in celle diverse [INTER])
 - o **Soft vs Hard** (se sono attivi entrambi i canali radio [SOFT] o solo 1 alla volta [HARD])
 - o **MT vs BS initiated** (se l'avvio di un handover lo fa il MT o la BS)
 - o **Backward vs Forward** (se la segnalazione di un handover avviene tramite BS di origine [BACKWARD] o BS di destinazione [FORWARD])

Parliamo ora di **EVOLUZIONE DELLA RETE CELLULARE**: 1G [analogico, solo voce] → 2G [GSM] → 2.5G → 3G → 3.5G → 4G [LTE] → 5G. Vediamo nel dettaglio:

- **GSM (2^a Generazione)** = consente l'invio di SMS e servizi come call forward, recall e busy tone. I Mobile Station (MS, ovvero i dispositivi, gli MT visti sopra [Mobile Terminal]) hanno bisogno di una **SIM** per connettersi alla rete (SIM = smart card con una CPU e una memoria in grado di memorizzare le informazioni dell'utente crittografate; l'ID della SIM è l'**MSI**).



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

- o **BTS** (Base Transceiver Station) = interfaccia fisica con il compito di trasmettere e ricevere. Rappresenta l'AP per i dispositivi e trasmette segnale solo verso gli utenti attivi
- o **BSC** (Base Station Controller) = controlla alto numero di BTS (10-100), sono collocati con un MSC (Mobile Switching Center)

Il **NSS** (Network & Switching Subsystem) gestisce chiamate, service support, mobility support e autenticazione; è composto da:

- o **MSC** (Mobile Switching Center) = gestisce la mobility support, call routing tra MT e GSMC (ovvero l'interfaccia tra GSM e le altre reti)
- o **HLR** (Home Location Register) = salva le informazioni degli utenti e i dati dinamici per gestire la user mobility
- o **VLR** (Visitor Location Register) = salva la posizione del dispositivo (MT) attualmente nell'area dell'MSC
- o **AUC** (Authentication Center)
- o **EIR** (Equipment Identity Register) = memorizza le informazioni dei dispositivi rubati

Le **frequenze usate** per il GSM variano in base allo scopo (ricezione o trasmissione) e usano il sistema **FDD** (frequency division duplex). Le trasmissioni sono organizzate in **burst** (simili ai pacchetti, ma funzionano su switching a circuito). I canali possono essere acceduti con **FDMA** o **TDMA**, mentre le frequenze sono divise in **FDM channels**, divisi a loro volta in **TDM frames** composti da **8 slot** [frequenza + slot di tempo identificano il canale fisico GSM].

⚠ **GSM non è full-duplex**: ogni MT trasmette per 1 time slot 1 burst di dati e rimane zitto per i rimanenti 7 slot. I tempi di propagazione non sono però nulli, quindi c'è possibilità di collisioni; la soluzione usata è la **timing advance**: la trasmissione del MT inizia prima del reale inizio del timeslot (a inizio e fine burst sono presenti dei **bit di guardia** per sincronizzare i burst).

⚠ I **canali fisici** del GSM hanno 8 canali (timeslot da 0 a 7), mentre i **canali logici** specificano "cosa" è trasmesso. Sono mappati però nel livello fisico e si dividono in control channels e traffic channels.

- **4G/LTE (4^a generazione)** = utilizzo di **FDMA** (FDM con frequenze portanti più vicine e ortogonal, in modo da fare meno interferenze) al posto del CDMA: in particolare **OFDMA** in downlink (BTS→MT) e **SC-FDM** in uplink (MT→BTS). Le frequenze usate sono:

- 2600MHz = max capacità (aree urbane)
- 1800MHz = alta capacità e poca interferenza
- 800MHz = alte coperture e alte interferenze (aree rurali)

In LTE vengono usati veri **pacchetti** (non più burst); la connessione avviene attraverso un **MME setup**, ovvero la configurazione di un home tunnel dalla rete di casa a quella dell'operatore.

C'è separazione netta tra **control plane** (piano di controllo) e **user plane** (piano di dati): a livello 3 si usa IP, mentre a livello 2 (link) ci sono 3 sottolivelli:

- **Medium access** = equivalente del MAC, si occupa dell'accesso al canale
- **RLC (Radio Link Control)** = frammentazione/assemblaggio dei dati, offre reliable data transfer
- **Packet data convergence** = compressione header e cifratura

La **RAN** (Radio Access Network o "long term evolution", che include tutti i dispositivi che interagiscono con i dispositivi utente) prende il nome di **E-UTRAN**, mentre il **CN** (Core Network, che include tutti i dispositivi responsabili del trasporto da/a internet verso gli utenti) prende il nome di **EPC** (Evolved Packet Core). Le **BS** (Base Station, AP) si chiamano **eNodeB**.

- **EPC** → approccio "clean state" (ripensato da zero), usa il **packet switching transport**. Si occupa di:
 - **Network access control**
 - **Routing** e trasferimento pacchetti
 - **Sicurezza**
 - Gestione della **mobilità**
 - **Radio resource management**
 - **Gestione della rete**
 - **Networking IP**

Le sue principali componenti sono:

- **Mobility Management Entity (MME)** [control plane]
- **Serving Gateway (SGW)** [user plane] = riceve e invia pacchetti tra gli eNodeB e EPC (la CN)
- **Packet Data Network Gateway (PGW)** [user plane] = connette EPC con reti esterne/internet
- **Home Subscriber Server (HSS)** = database di informazioni sugli utenti (simile a HLR del GSM)

Tutte le comunicazioni sono gestite con dei **tunnel** chiamati **BEARERS**, situati tra PGW e SGW; i tunnel possono essere creati per soddisfare servizi specifici. 3 tipi di bearers:

- **S5 bearer** = connette SGW e PGW
- **S1 bearer** = connette eNodeB con SGW
- **Radio bearer** = connette UE e eNodeB

- **E-UTRAN** → consiste principalmente di **eNodeB**. Si occupa di:
 - Gestione delle **risorse radio**
 - **Compressione lossless degli header**
 - **Sicurezza**
 - **Connettività verso EPC**

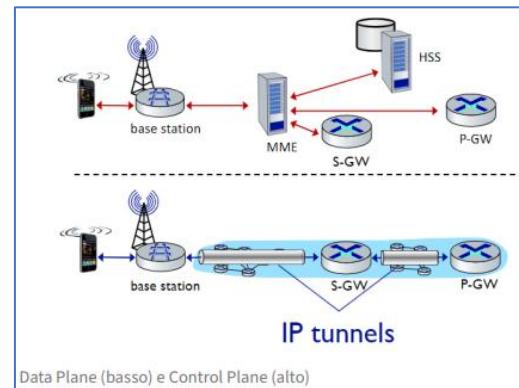
⚠ La tecnologia di tunneling usata per le reti cellulari si chiama **GPRS Tunneling Protocol** (tunnel su UDP).

⚠ Un nodo per associarsi ad una BS (eNodeB) deve fare questi passi:

- ✓ Ogni 5 ms la BS manda su tutte le frequenze un broadcast primary sync signal
- ✓ Il dispositivo riceve il primary sync signal e manda un 2° sync signal alla stessa frequenza
- ✓ Il dispositivo sceglie la BS a cui associarsi

⚠ I terminali possono andare in **sleep mode** per risparmiare energia:

- ✓ **Light sleep**: ogni 100 ms, il dispositivo si sveglia per controllare se ci sono messaggi da inviare o ricevere; se non ci sono messaggi torna a dormire
- ✓ **Deep sleep**: dopo 5/10 secondi di inattività, il dispositivo si mette in deep sleep



Data Plane (basso) e Control Plane (alto)

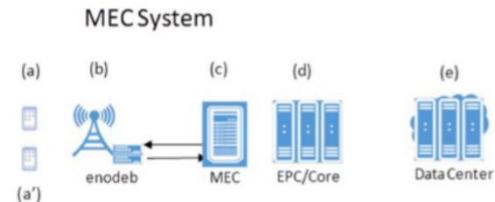
- **5G (5^a generazione)** = l'obiettivo è superare la differenza tra wifi e cellulare: per questo bisogna effettuare "network slices" riservate ad una certa comunicazione che consentano di emulare ciò che faceva il circuito

(virtualizzazione della rete mediante **SDN** che si occupa di controllo disaccoppiato dell'infrastruttura fisica). L'**Orchestrator function** (o network) si occupa di allocare in modo appropriato le risorse di rete e viene supportato il **cross-domain orchestration** (gestire più domini di rete). Utilizzi del 5G sono:

- **eMBB** (enhanced Mobile Broadband) = servizi ad alta qualità per utenti mobili
- **mMTC** (massive Machine Type Communication) = comunicazione industriale a bassa latenza
- **URLLC** (Ultra-Reliable Low-Latency Communication)

Le **tecnicologie usate** sono:

- **Forme d'onda avanzate**
- **MIMO avanzate** (antenne)
- **Millimeter Wave** (spettro ad altissime frequenze)
- **SDN** (Software Define Networking) [spiegato sopra]
- **NFV** (Network Function Virtualization) [sposta servizi di rete dall'HW al SW]
- **SDN/NFV Orchestration** [gestione di tutte le risorse in modo dinamico e flessibile]
- **RAN** (Radio Access Network) [basata sui gNodeB, evoluzione degli eNodeB] = consente l'utilizzo di un numero variabile di slot per subframe, in cui la trasmissione può iniziare in un punto qualsiasi dello slot; supporta slot aggregation per dati pesanti e gli slot più corti consentono uno spacing più elevato
- **MEC** (Mobile Edge Network) ["cloud computing" per mobiles] [dx img]
- **CN** (Core Network) [sono i dispositivi responsabili del trasporto dei dati]

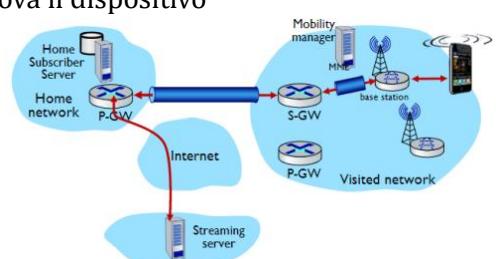


⚠ Una differenza importante tra 4G e 5G è il QoS mediante flussi (invece di end-to-end bearers) e il fatto che 5G utilizza il concetto di **SBA** (Serviced Based Architecture) [sarebbe l'EPC di 5G], la quale usa:

- **AMF** (Access & Mobility Function) = evoluzione del MME
- **SMF** (Session Management Function) = evoluzione del control plane con SGW e PGW
- **UPF** (User Plane Function) = evoluzione del data plane

⚠ Riguardo alla **mobilità in 4G/5G**, è presente una **home network** e una **visited network** (dove si fa roaming): quando accedo alla visiting network, la nuova rete mi dà un indirizzo (spesso privato). Quando un utente si sposta in rete, si devono gestire 4 fasi:

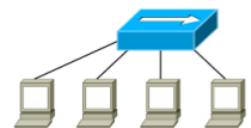
- 1) Associazione alla **nuova BS** (Base Station)
- 2) Configurare la **control plane** informando la rete di dove si trova il dispositivo
- 3) Configurare la **data plane per creare i tunnel** [dx img]
 - a. SGW a BS tunnel = quando il dispositivo cambia BS, cambia l'endpoint ip address del tunnel
 - b. SGW a home-PGW tunnel = implementazione del routing indiretto
 - c. Tunneling via GPT (GPRS tunneling protocol, con UDP)
- 4) **Mobile Handover** (se la cella dovesse cambiare nello spostamento) attraverso le BS in 1 rete cellulare:
 - a. BS sorgente seleziona BS destinazione e invia un handover (HR) request
 - b. BS destinazione preallocata 1 radio time slots e risponde con HR ack con le info del device
 - c. BS sorgente informa il dispositivo del nuovo BS e l'HR risulta completato agli occhi del device
 - d. BS sorgente smette di inviare i datagrammi al device, ma li inoltra alla BS destinazione
 - e. BS destinazione informa MME che è il nuovo BS per il device
 - f. BS destinazione manda un ack alla BS sorgente informando che l'HR è completato (la BS sorgente rilascia le sue risorse)
 - g. I datagrammi del device possono ora usare il nuovo tunnel dal target BS al SGW



3) MODERN LAN DESIGN

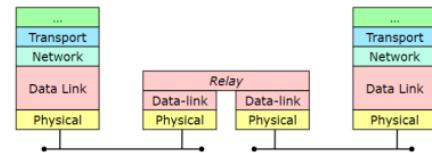
Sul livello fisico si usa **IEEE 802** (802.3 ethernet; 802.11 WiFi). I dispositivi **LAN** (Local Area Network) sono:

- **Hub** (o **ripetitori**) → dispositivi di livello 1 (“**specchi inconsapevoli**” perché essendo di livello 1 non possono comprendere trame ethernet), con **unico dominio di collisione ma separato dominio fisico**. Usano la topologia a stella passiva; interconnettono le reti con lo stesso MAC, ripristinando la degradazione del segnale in modo da coprire maggiori distanze. I ripetitori con più di 2 porte sono gli Hub (necessari per twisted pairs e fiber cabling [fibra ottica])



⚠ **Dominio di collisione** = area con 1 access control algorithm (es. quella coperta da 1 singolo cavo fisico). **Dominio di broadcast** = area in cui un frame può essere propagato (**può includere molti collision domains**).

- **Switch** (o **Bridge**) → dispositivi di livello 2 (quindi in grado di comprendere trame ethernet), con **separato dominio di collisione** (gestisce ed evita le collisioni) **ma unico dominio di broadcast** [su singoli segmenti di rete ci possono essere ancora collisioni che vengono risolte in modalità **full-duplex** e questo rende CSMA/CD non più necessario]. Usano la topologia a stella attiva; sono implementati in software e composti da 2 porte (che hanno diversi MAC). Se ha più di 2 porte è uno Switch. Usa “**store&forward**”, ovvero riceve tutta la trama, ragiona e poi la inoltra verso la porta corretta (che ha individuato grazie al MAC e alla tabella di inoltro).



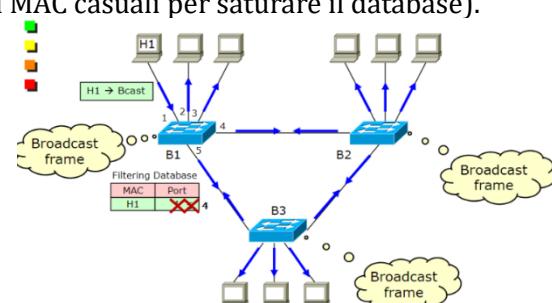
Le **LAN moderne** sono basate su full-duplex, switch e ethernet; viene usata la topologia “**hub&spoke**”, ovvero connessioni punto punto tra gli host e il bridge in modo da non avere domini di collisione. I bridge e gli switch in ethernet prendono il nome di **transparent bridge/switch** (**plug&play**) in quanto per l’utente finale non devono essere percepite differenze tra gli hub e gli switch (devono avere lo stesso funzionamento con o senza bridges).

Ciascuna porta di un bridge ha un indirizzo MAC che non viene usato per eseguire il forwarding dei data frames ma per consentire l’indirizzamento del traffico con i management frames; il **forwarding** avviene attraverso MAC-based forwarding oppure in multicast/broadcast attraverso il **flooding** (invia il pacchetto su tutte le porte eccetto quella da cui il frame è stato ricevuto). Localmente deve essere disponibile il **Filtering Database** (una sorta di routing table che al posto degli indirizzi IP ha gli indirizzi MAC delle destinazioni); questa viene popolata mediante il **backward learning**: quando uno switch riceve una trama, riceve anche il MAC sorgente e, grazie a questo, capisce quale porta usare per raggiungere il dispositivo; per tale motivo la tabella ha 2 tipi di entry:

- **statiche** = non aggiornate dal learning
- **dinamiche** = aggiornate dal learning (eliminate quando le stazioni non esistono più o dopo un lasso di tempo)

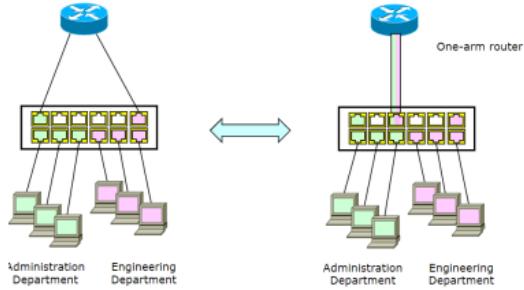
⚠ Un end-system con MAC address non presente nel filtering database è sempre raggiungibile, in quanto un frame inviato su un host inesistente viene inoltrato su tutte le porte. Legato a ciò, una filtering table può subire dei **MAC flooding attack** (vengono generati dei frame con sorgenti MAC casuali per saturare il database).

⚠ Un altro problema è il **broadcast storm**: se viene inviato un pacchetto in broadcast, il pacchetto viene mandato a tutti i bridge e reinoltrato generando un “**loop**” che non termina fino al riavvio degli switch (ciò è dovuto alla mancanza del campo time-to-live nelle trame di livello 2); questo viene risolto con lo **Spanning Tree**, il quale elimina i cicli.



- **Router** → dispositivi di livello 3 (**non trasparenti**), con **separato dominio di broadcast**. Le **VLAN** (Virtual LAN) consentono ad un set di porte di uno switch di simulare che facciano parte di un **dominio di broadcast separato**, ma usando un’unica infrastruttura di rete. Nelle VLAN l’header di livello 2 viene scartato in favore di un header nuovo creato con un differente MAC. Per far parlare le VLAN ci vuole un router con tutte le **sottoreti connesse** [**Il traffico di livello 2 non può attraversare le VLAN perché sono divise logicamente a livello 3; infatti il broadcast non può attraversare VLAN differenti e dunque non è possibile usare**]

ARP (livello 2) per individuare gli indirizzi MAC di un'altra VLAN (infatti a livello 2 le VLAN sembrano ancora una rete unica)]. Un altro modo è connettere il router ad un'unica interfaccia che lavora per entrambe le sottoreti, ottenendo il “**one-arm-router**”.



Per associare un frame ad una VLAN si usa il **tagging** (un campo aggiuntivo di 4 Byte nella trama ethernet contenente il **vlanID**, in modo che questo venga identificato anche negli switch rimanenti) [modifica quindi il MAC già esistente]. Legato a ciò, le porte si dividono in:

- **access** = inviano e ricevono trame “non taggate” (usate per connettere end-stations alla rete)
- **trunk** = inviano e ricevono trame “taggate” (usate nelle connessioni switch-to-switch e per connettere router/server)

4) VPN

Una **VPN (Virtual Private Network)** realizza una connettività tra più sottoreti distinte in modo che possano comunicare come se fossero un'unica **rete privata** (usate infatti per non dover utilizzare cavi per la realizzazione di reti private) [utilizzare una VPN non rende automaticamente una connessione **sicura**, ma servono i protocolli di sicurezza]. Sono composte da:

- **Tunnel** → incapsulano in modo sicuro il traffico in transito sulla rete condivisa
- **VPN gateway** → apre e termina i tunnel

Definiamo alcune **soluzioni**:

- **Site-to-site** = a livello di sottorete (gateway), connette 2 reti remote
- **End-to-end** = a livello di host (terminali), connette 2 terminali remoti
- **Access VPN/Remote VPN/Dial In** = tra terminale e intera sottorete

Dal punto di vista della **distribuzione**:

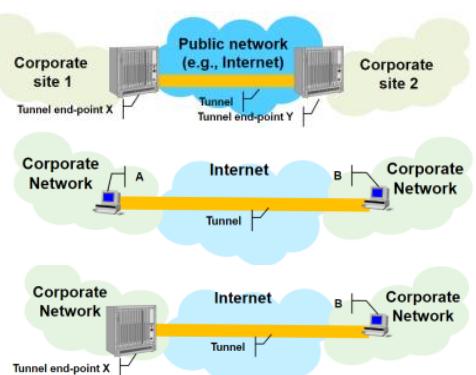
- **Intranet VPN** = interconnette uffici remoti della stessa azienda
- **Extranet VPN** = interconnette aziende diverse; sono presenti alcune limitazioni, in quanto si vuole ridurre l'accesso alle risorse di rete mediante **firewall**, ottenere Overlapping Address Spaces mediante **NAT** e controllare il traffico

L'accesso ad internet può essere:

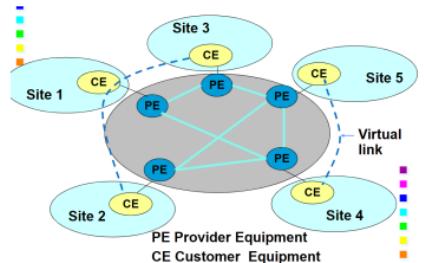
- ✓ **Centralizzato**: gli utenti utilizzano una rete IP pubblica per connettersi
- ✓ **Distribuito**: gli utenti si connettono attraverso la propria rete IP e la VPN è usata solo per il traffico aziendale
- ⚠ Riassumendo, la VPN garantisce separazione dei dati (con tunneling), sicurezza (con crittografia), prevent tempering (integrità), identificazione delle sorgenti e degli endpoint (autenticazione) e confidenzialità dei dati.

Vediamo nel dettaglio le **soluzioni** prima citate:

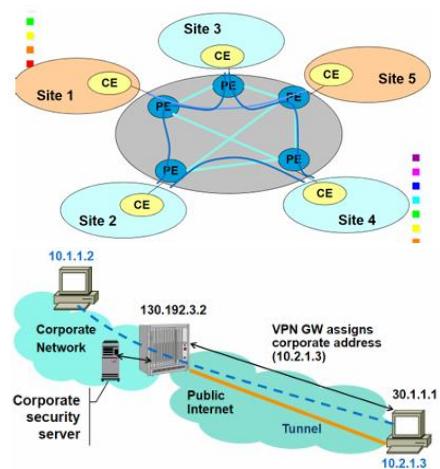
- **Site-to-site (s2s)** = a livello di infrastruttura pubblica
- **End-to-end (e2e)** = danno maggiore sicurezza in quanto il tunnel è diretto tra i 2 host
- **Remote VPN Tunneling** = connette un endpoint con un vpn gateway



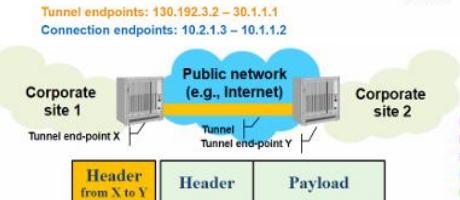
- **Overlay Model** = la rete pubblica non partecipa alla realizzazione della VPN, non sa quale siano le destinazioni e il routing/conessione avviene attraverso VPN gateways; è trasparente. Si può pensare come ad una rete parallela che si sovrappone a quella fornita dall'ISP
- **Peer Model** = ciascun VPN gateway interagisce con i router pubblici. Il routing è migliorato, ma chi realizza la VPN è coinvolto nella comunicazione di rete (non più trasparente). Inoltre i tunnel sono tra i router, compromettendo in parte la sicurezza
- **Customer Provisioned VPN** = il cliente implementa la soluzione VPN. Il network provider non è a conoscenza del fatto che il traffico generato dal cliente sia VPN; i CE (Customer Equipment) sono i terminatori dei tunnel. L'host deve avere 2 indirizzi



- **Provider Provisioned VPN** = il provider implementa la soluzione VPN (quindi sotto il controllo dell'azienda) e i terminatori dei tunnel sono i PE (Provider Equipment). Il remote host deve essere sempre nella VPN (si ha quindi 1 solo indirizzo)



- **Access Customer Provisioned VPN** = sui terminatori della VPN è necessario avere un indirizzo pubblico, costringendo ad avere 2 indirizzi. È però più semplice a livello di Customer Provisioned.

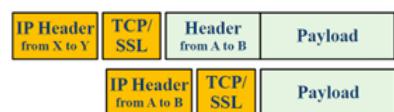


Le VPN hanno **2 topologie** virtuali:

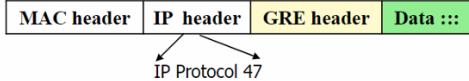
- [Hub & Spoke](#) = ciascun branch comunica direttamente con il quartier generale (sono quindi richiesti pochi tunnel, ma c'è rischio di bottleneck)
- [Mesh](#) = utilizza molti tunnel (difficile la configurazione manuale, ma il routing è ottimizzato)

Una VPN può essere implementata in **più livelli dello stack ISO/OSI**:

- **LIVELLO 2:**
 - [Virtual Private LAN service](#) (emula la LAN e può essere usata per connettere alcuni segmenti LAN)
 - [Virtual Private Wire service](#) (emula connessione cablata; può trasportare ogni protocollo)
 - [IP-only LAN-like service](#) (i CE sono IP routers/hosts; viene usato solo IP)
- **LIVELLO 3:** si usano soluzioni **standard**: i pacchetti sono inviati attraverso la rete pubblica con routing basato su indirizzi di livello 3 (IP) che possono essere **peer** (vpn, corporate, indirizzi cliente) o **overlay** (backbone address). I pacchetti sono trasportati nella rete come pacchetti IP nelle seguenti modalità:
 - pacchetto IP in pacchetto IP ([IP in IP](#)) [nel tunneling basato su IP in IP, il tunneling abilita la comunicazione ma non assicura la sicurezza]
 - frame (livello 2) in pacchetto IP ([IP in frame](#))
- **LIVELLO 4:** le soluzioni VPN di livello 4 provvedono solo alla **sicurezza** e sono **non standard**:
 - [Site-to-site \(s2s\)](#) = VPN costruita usando connessioni TCP, mentre la sicurezza è data con SSL/TSL
 - [End-to-end \(e2e\)](#) = tunnel terminato da un end system



Parliamo ora di **GRE** (**Generic Routing Encapsulation**), un protocollo di livello 3 che si basa su encapsulamento IP in frame, con formato:



L'**IP header** è composto da:

- **C, R, K, S** (flag)
- **s** = strict source routing flag (se il destinatario non è raggiunto quando la source route list termina, il pacchetto viene eliminato)
- **recur** = max n° di volte che il pacchetto può essere encapsulato (deve essere 0)
- **protocol** = id del protocollo per il payload
- **routing** = sequenza di indirizzi dei router IP per ASs o per source routing

⚠ Anche se GRE è di livello 3, può encapsulare qualsiasi protocollo. L'**enhanced GRE** ha anche alcune funzionalità:

- ✓ **Payload Length** (n° di bytes ad eccezione dell'header GRE)
- ✓ **Call ID** (session ID per il pacchetto)
- ✓ **Sequence number** (per ordinare i pacchetti ricevuti, error detection e correction)
- ✓ **Acknowledgement number** (max n° di pacchetti GRE ricevuti in sequenza nella sessione; ACK cumulativo)

La gestione del flusso dati (**flow control**) avviene attraverso una sliding window; i pacchetti non possono essere ricevuti fuori ordine. Ogni volta che un ack viene inviato, i timeout values vengono ricalcolati; viene fatto **controllo della congestione** (in quanto il timeout non causa la ritrasmissione ma è usato per muovere la sliding window).

⚠ Ci sono **protocolli di livello 2 (VPN)** [solo lettura]:

- **LT2P** (Layer 2 Tunneling Protocol) = provider provisioner; indipendente dal protocollo di livello 2 sull'host e la sicurezza è garantita da IPsec
- **PPTP** (Point to Point Tunneling Protocol) = customer provisioner; ha bassa encryption e autenticazione e usa un key management proprietario

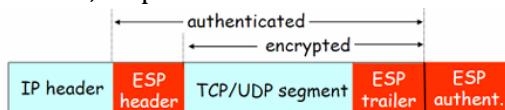
Parliamo ora di **IPsec**, il quale si basa su 2 ulteriori protocolli ovvero **AH** e **ESP**:

- **AH** (**Authentication Header**) = garantisce l'**integrità dei dati**, l'**autenticazione della sorgente**, ma non la **confidenzialità**. L'header è inserito tra l'header IP e il payload, con protocol field pari a **51**.



Inoltre, non si può usare il NAT. Alcuni campi sono:

- **SPI** (Security Parameter Index) = contiene il Session ID e viene usato per verificare la signature
- **Authentication data** = contiene la signature generata dal router di destinazione
- **Next header** = specifica il protocollo usato nel payload (es. TCP, UDP, ICMP...)
- **ESP** (**Encapsulation Security Payload**) = garantisce la **confidenzialità dei dati**, i quali sono criptati insieme al next header nel ESP trailer. Consente l'**autenticazione dell'host** e l'**integrità dei dati**, mediante un'autenticazione simile a quella di AH; ha protocol field **50**.



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

- **AH** = garantisce l'integrità dell'header originario, del payload originario e del nuovo header
- **ESP** = garantisce solo l'integrità dell'header originario, del payload originario, ma **non del nuovo header**

Un **tunnel IPsec** è quindi capace di garantire encapsulazione, autenticazione e cifratura tra 2 VPN gateways; 2 modalità di funzionamento:

- **transport mode** = l'header IP non è completamente protetto ma solo autenticato se si usa AH (si usa un transport layer)
- **tunnel mode** = l'header IP è completamente protetto sia nell'header sia nel payload. Inoltre è previsto che venga criptata l'intestazione IP, l'intestazione TCP/UDP e il payload del pacchetto interno

Le **SA** (Security Association) sono canali logici unidirezionali che rappresentano un “contatto” tra 2 entità coinvolte nella comunicazione; negoziano alcune informazioni prima di cominciare lo scambio di pacchetti IPsec. Il protocollo **IKE** (Internet Key Exchange) viene usato per stabilire e mantenere le SA in IPsec. Per lo scambio sicuro dei dati, vengono usate 1 o più **SA “figlie”**, le quali usano la negoziazione di chiavi tramite IKE SA (potrebbero tutti partire da 1 shared secret). Si usa **ISAKMP** (Internet Security Association Key Management Protocol) per la negoziazione di parametri IKE e dello shared secret (oltre a chiavi pubbliche, certificati e dati firmati/autenticati).

⚠ Ci sono **problematiche** tra l’uso di IPsec e NAT? Sì perché IPsec deve garantire l’autenticazione, che non è possibile se il NAT modifica l’indirizzo IP dei pacchetti.

IPsec VPN ha però **costi elevati, troppe opzioni** (necessita di una configurazione per garantire sicurezza) e **opera a livello kernel** (non sempre va bene). Un’alternativa è **SSL VPN** che ha come **vantaggi**:

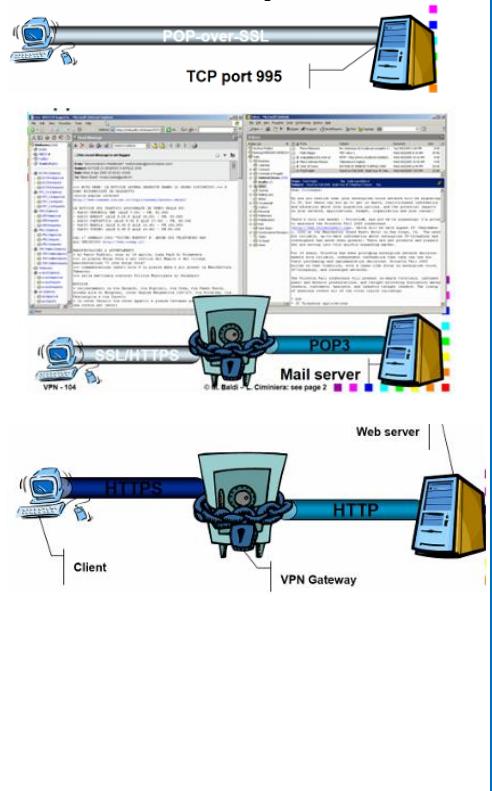
- ✓ Complessità <
- ✓ Non interferisce con il kernel
- ✓ Molto usato
- ✓ Sicurezza > (proprio per SSL)
- ✓ No problemi di attraversamento del NAT

Ha però anche **svantaggi**:

- ✗ **Prestazioni:**
 - IP su TCP → nessuna consegna di pacchetti dopo 1 smarrito (strozzatura del tunnel a causa del controllo della congestione TCP)
 - TCP su TCP → imprevedibile
 - Ampi buffer di trasmissione nei gateway
- ✗ Pacchetti scartati a livello più alto (quindi vulnerabile ad **attacchi DDOS**)
- ✗ **Interoperabilità** (client e server devono installare lo stesso software)
- ✗ Features specifiche del produttore (potrebbe presentare bug a causa delle **soluzioni proprietarie**)
- ✗ **Disponibilità del client sulle specifiche piattaforme**

Le VPN con IPsec connettono reti, host a reti, oppure host a host. Le SSLVPN connettono **utenti a servizi** (per questo vengono dette “**pseudoVPN**”); queste usano tunneling TCP o UDP, forniscono NAT traversal, packet filter traversal, router traversal e usano client universali (web browser). Tra le **PseudoVPN** abbiamo quindi:

- **Protocolli con SSL** (definiti “*secure application protocol*”)
- **Application Translation** = sfrutta protocolli nativi tra il VPN server e l’application server; si usa quindi un modo “standard” e meno protetto, ovvero il protocollo nativo, che verrà richiamato dall’esterno attraverso un’interfaccia sicura SSL
- **Application proxying** = usa VPN gateway per scaricare le webpage attraverso http e le invia tramite https. Consente la compatibilità con server vecchi, in quanto la richiesta viene fatta in modo sicuro dal client al gateway e in modo standard verso il server
- **Port Forwarding**



La **POSIZIONE DEL VPN** comporta **problematiche differenti**:

- **Interno** → nessuna ispezione del traffico VPN oppure VPN gateway protetto da firewall
- **Parallello** → potenziale accesso non controllato

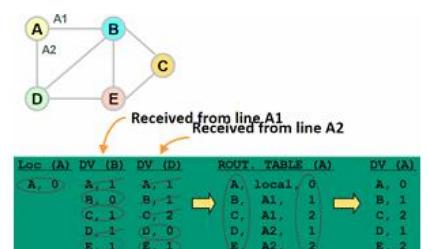
- **Esterno** → VPN gateway potrebbe essere protetto da un access router
 - **Integrato** → max flessibilità
- ⚠ Solitamente vengono messi degli IDS (Intrusion Detection System) all'esterno dei firewall senza controllo del traffico VPN e dopo il VPN gateway.
- ⚠ Alcune **problematiche** sono relative all'uso del **NAT** nella rete; non si può usare PAT/NAPT. Inoltre, in transport mode le porte non sono visibili, mentre in tunnel mode l'indirizzo IP del pacchetto protetto può essere cambiato prima che questo entry nel gateway.
- ⚠ Alcune anomalie sono:
- **Monitorability Anomaly** = quando un nodo del canale "congiunto" può vedere lo scambio dei dati
 - **Skewed Channel Anomaly** = quando si ha una sovrapposizione errata dei tunnel, che rimuove la confidenzialità nella comunicazione [anche se ho più livelli di sicurezza, se configurati male, posso avere un problema di confidenzialità]

5) ROUTING

Distinguiamo:

- **ROUTING** ("proactive routing") = indipendente dal traffico che passa istantaneamente su un nodo; definisce quale percorso è il migliore rispetto ad altri in base ad una metrica scelta [è il normale routing].
- I suoi **ALGORITMI** sono:

- a. **NON-ADAPTIVE** (statici) → controllo di rete dall'admin (non si adatta al cambio di topologia):
 - I. **Fixed Directory routing** (usa static routing e richiede configurazione manuale)
 - II. **Flooding** (e derivati [selective])
- b. **ADAPTIVE** (dinamici)
 - I. **CENTRALIZED** routing = 1 nodo (chiamato **Routing Control Center** [RCC]) si occupa di gestire la rete: deve sapere le informazioni di tutti i nodi, calcola e distribuisce le routing table. Semplifica il troubleshooting (ovvero l'individuazione di problemi), ma porta a bottleneck e single point of failure
 - II. **ISOLATED** routing = ogni nodo è indipendente (no scambio di informazioni tra nodi), quindi non si ha garanzia che il pacchetto venga effettivamente trasmesso [es. in rete lineare]
 - III. **DISTRIBUTED** routing = i router collaborano nello scambiare informazioni sulla connettività (ciascun router decide indipendentemente, ma in modo coerente) [ibrido tra i 2]:
 - i. **Distance Vector** (**Bellman-Ford**): ogni nodo invia/riceve ai/dai **nodi adiacenti** informazioni inerenti alla distanza con gli altri router; ogni nodo ha la **lista completa dei destinatari**. Sono necessari dei router transitori e, visto che ogni nodo comunica con i vicini, è importante tenere conto della distanza dall'announcing router.



L'algoritmo cerca ogni volta la distanza < per raggiungere un determinato nodo, tenendo conto dei **percorsi alternativi** in caso di guasto. All'inizio ogni router ha solo le informazioni in locale ("cold start").

I **vantaggi** sono semplicità di implementazione e nessuna particolare configurazione. Presenta però alcune **problematiche**:

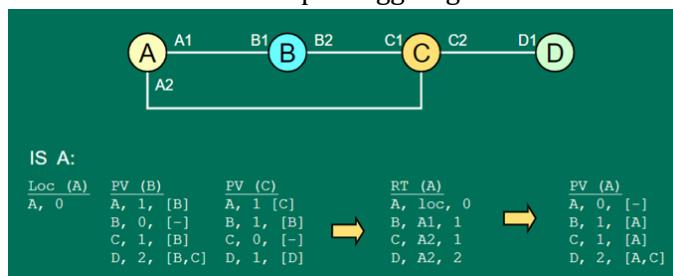
- **Black hole** → se un nodo non risponde più ai messaggi di routing, non si ha più informazioni sulla rete perché queste vengono passate tra nodi adiacenti
- **Count to infinity** → ho 3 nodi A-B-C connessi: se rimuovo l'arco B-C, B segna che non può più raggiungere C, mentre A pensa ancora di poterlo raggiungere con 2 salti e

quindi comunica a B che lo può raggiungere; quindi B si segna che può raggiungere C con 3 salti etc... Questo errore viene propagato all'infinito

- **Bouncing effect** → se un nodo è più vicino ad un altro, ma il percorso è più lungo (+ hop), allora il nodo più vicino non sarà scelto [succede quando ho oscillazioni/fluttuazioni su archi che si accendono e spengono di continuo]
- **Split horizon** → se C raggiunge A mediante B, è inutile per B provare a raggiungere A mediante C (previene cicli tra 2 nodi)
- **Patch hold down** → se un link L fallisce, le destinazioni raggiungibili da L vengono considerate non raggiungibili per un certo tempo ("in quarantena") [quando un link fallisce, il costo viene incrementato fino a che non si arriva al costo max, e qui si ricerca un altro percorso]
- **Route poisoning** → le route scorrette sono inoltrate al posto di essere omesse (es. nel count to infinity)

Il tempo di convergenza nel caso peggiore (*complessità*) è tra $O(n^2)$ e $O(n^3)$, e risulta limitata dai router più lenti, dal set space dei router e dal numero di link

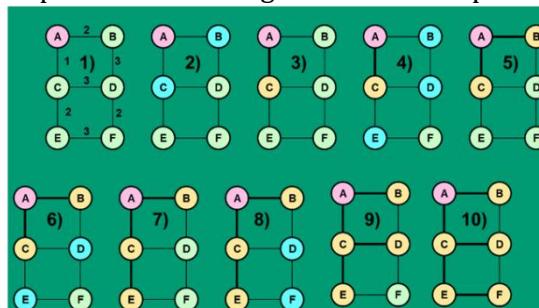
- ii. **Path Vector**: elimina i cicli inoltrando, **in aggiunta alle informazioni sulla distanza**, le informazioni sui **nodi attraversati** per raggiungere una certa destinazione



- iii. **Link State**: vengono inoltrate le informazioni relative a **tutta la rete** (e non solo quelle dei nodi adiacenti), contenenti lo stato di ogni nodo (e non solo di quelli attraversati); in questo modo ciascun nodo può realizzare una mappa locale, inviando le informazioni usando un **selective flooding**.

La **convergenza** è rapida, i **link state** sono piccoli e **raramente genera loop** (può comunque **generarli**), ma è più **complesso** da implementare ed è **sconveniente in reti piccole**. I link state vengono generati quando avvengono cambiamenti topologici, ma nei protocolli attuali si preferisce rigenerarli periodicamente per > affidabilità.

Per calcolare le informazioni della rete si usa l'**algoritmo di Dijkstra**: viene usato per calcolare l'albero di copertura minima di un grafo (ovvero l'**albero di instradamento**) [*complessità* $O(L \log n)$ con $L = \text{n}^{\circ}$ di link, $n = \text{n}^{\circ}$ di nodi]; usa il **shortest path first** (dove il prossimo nodo è il più vicino alla sorgente e il next hop è inserito nella routing table)



- **FORWARDING** ("on the fly routing") = gestisce i pacchetti mediante informazioni locali come routing / forwarding table; è il risultato del routing [è il forwarding]. Avviene uno switching, ovvero il trasferimento verso una porta di output, oltre che di trasmissione. Dipende dall'indirizzamento che si vuole:

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

Parliamo ora di **INTERNET ROUTING ARCHITECTURE**: i protocolli di routing viaggiano tra livello IP (3, rete) e livello TCP (4, trasporto).

Definiamo **Dominio di Routing (routing domain)** l'insieme di router che usano lo stesso protocollo di routing; un router può far parte di più routing domains e può ridistribuire le informazioni imparate con un protocollo mediante un altro protocollo.

Definiamo **Autonomous System [AS]** un gruppo di router e sottoreti raggruppate in base alla topologia o ad un criterio organizzativo, sotto il controllo di 1 autorità. È scalabile, in quanto è il singolo AS a decidere dove far passare i propri dati. È identificato da 2 Byte. All'interno di un AS ci sono dei dispositivi detti **border gateway**, posizionati al confine tra i diversi AS; distinguiamo quindi i protocolli **iBGP** (intra Border Gateway Protocol; comunicazione all'interno dell'AS/della rete) e **eBGP** (exterior Border Gateway Protocol; comunicazioni tra AS). Analogamente si dice **exterior routing** il routing tra diversi AS (non segue il percorso più corto, ma il migliore). Definiamo **NAP (Neutral Access Point)** un punto di accesso neutrale che permette di collegare più AS tra loro, mentre **IXP (Internet eXchange Point)** un punto di scambio di traffico tra AS [un NAP/IXP consiste in una LAN dove router di diversi AS sono connessi; **un NAP/IXP permette quindi a più ISP di comunicare tra loro**]. Inoltre si definisce **private peering** la comunicazione che avviene tra 2 ISP dello stesso tier.

I **PROTOCOLLI DI ROUTING** si distinguono in:

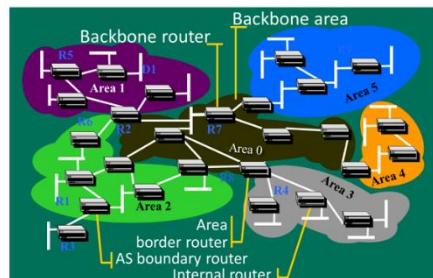
- **IGP (Interior Gateway Protocol)** → le informazioni sono distribuite nella topologia; le routes sono scelte in base alle informazioni della **topologia** (in modo da trovare la route migliore). Gli algoritmi di IGP consentono di usare metriche diverse rispetto all'hop count e consentono il **multipath routing** (usare più percorsi per raggiungere 1 destinazione [può generare loop]):

- **Distance Vector:**

- **RIP** (Routing Information Protocol) = usa l'hop count come metrica ed è imbustato direttamente in UDP; non è usato in reti di grandi dimensioni ed è instabile
- **IGRP** (Interior Gateway Routing Protocol) = supera alcuni dei problemi di RIP, ma è un sistema proprietario di Cisco

- **Link state:**

- **OSPF** = usa routing gerarchico (routing domain diviso in **aree**) e può essere iterato. Ogni area ha una visione completa della sua topologia interna, ma verso l'esterno conosce solo i collegamenti (router di frontiera) per parlare con le altre aree. Distinguiamo:
 - **Strictly hierarchical routing** (i router non hanno informazioni sull'esterno della rete, quindi altamente scalabile)
 - **Loosely hierarchical routing** (i router devono scambiare più informazioni, quindi meno scalabile)

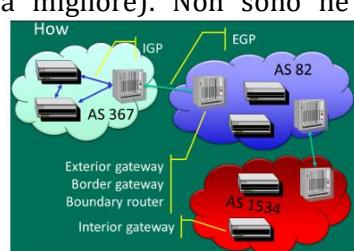


[Per N router avrà N^2 link; la complessità di Dijkstra è lineare nel numero di link]

- **IS-IS** = estensione del protocollo OSI, quindi routing gerarchico con diversi livelli; non è più usato nelle nuove strutture

- **ECP (Exterior Gateway Protocol)** → le informazioni degli AS sono distribuite; le decisioni vengono prese in base alle **policy** (trova quindi la route "preferita", non necessariamente la migliore). Non sono né completamente distance vector né completamente link state:

- **BGP (Border Gateway Protocol)** = usa algoritmi path vector (con vector exchange che avviene su TCP [per maggiore affidabilità] solo dopo un cambiamento), inviando alla destinazione la sequenza di AS attraversati; è possibile configurare la route computation policy



- **IDRP (Inter Domain Routing Protocol)** = usa TCP/IP e rappresenta un'evoluzione di BGP per OSI [doveva essere la soluzione per IPv6, ma non è molto usato]
- **routing statico**

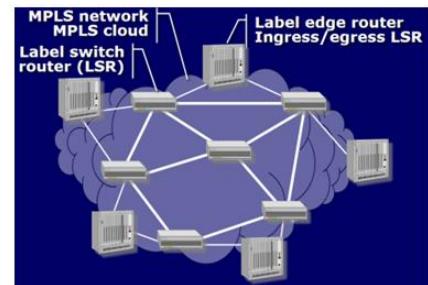
6) MPLS

MPLS consente la realizzazione di un **nuovo tipo di rete pubblica basata su IP** (rete pubblica = rete con traffico di diversi utenti e aziende su cui è possibile vendere dei servizi). MPLS elimina la "struttura a cipolla", usando **1 solo protocollo per la gestione delle comunicazioni tra i nodi e verso l'esterno** (abbattendo i costi degli operatori).

Il forwarding avviene attraverso l'aggiunta di un'**etichetta** (al posto di guardare l'indirizzo IP destinazione; consente quindi l'invio verso multiple destinazioni, usando le etichette come indice) [maggior rapidità, in quanto si skippa il longest prefix matching]. L'obiettivo di MPLS è rendere IP **connection-oriented**, nonostante la rete continui ad essere a **commutazione di pacchetto a circuito virtuale**.

Riguardo all'**ARCHITETTURA DI RETE**, MPLS non usa gli end system e può essere usato in una porzione di una rete (chiamata **MPLS Cloud** [non c'entra con il cloud]):

- **LSR (Label Switching Router)** → fanno la commutazione di pacchetto basata su etichetta
- **LER (Label Edge Router)** → router che non ha altri router MPLS collegati, posti al bordo della rete: aggiungono l'etichetta quando un nodo entra nella rete e la rimuovono prima che esca
- **LSP (Label Switch Path)** → percorso di comunicazione attraverso cui viaggiano i pacchetti
- **LSP tunnel** → entra ed esce dalla MPLS (tra 1 ingress LSR e 1 egress LSR)



L'etichetta viene cambiata ad ogni nodo, in modo da mantenere un'etichetta più corta e poterla riusare senza dover mettere d'accordo i nodi (in quanto, in caso contrario, dovrebbero essere uniche nella rete, in questo modo invece sono **uniche nel nodo**): questa tecnica è il **LABEL SWITCHING** e consente **scalabilità**.

Gli **elementi di MPLS**:

- **Header MPLS** → contiene l'etichetta. È di livello 2 ed è fatto da **più moduli uniti** (perciò detto *shim header*); ogni modulo è composto da:
 - **label** (20 bit) = l'**etichetta** da trasmettere; la lunghezza consente il passaggio di 1 mln di flussi sul link
 - **exp** (3 bit) = **experimental bits** (usati per il Class Of Service)
 - **S** (1 bit) = **bottom of stack**; è posto ad 1 se è l'ultimo modulo dello stack, 0 altrimenti
 - **TTL** (8 bit) = **time to live**
- ⚠ Nel caso di ATM e frame relay erano già presenti dei campi per la label, e quindi sono stati riusati (VIC/VPI in ATM e DLCI in frame relay) [gli switch ATM/frame relay diventano degli switch MPLS]
- **Protocolli per la distribuzione delle etichette** → definisco **FEC** (Forwarding Equivalence Class) un insieme di pacchetti che hanno lo stesso destinatario; dunque un **LSP** è un percorso di comunicazione che viene usato per trasportare un FEC.

Quando viene creato un LSP, sono necessarie **3 operazioni da parte degli LSR**:

- I. **LABEL BINDING** = **associazione di un'etichetta ad alcuni pacchetti di una FEC**; viene effettuato in modalità **downstream** (ovvero, tra 2 nodi che si trovano ai capi di un link, il binding è eseguito da quello a valle). Per sapere di dover usare tale etichetta, l'LSR a monte (*upstream node*) deve essere notificato e può avvenire in 2 modi:
 - a. **unsolicited** (senza richiesta diretta)
 - b. **on-demand** (in seguito ad una richiesta diretta)

Distinguiamo inoltre:

- ✓ binding **statico** = avviene attraverso un gestore di rete che stabilisce e impone l'etichetta ai nodi della rete; non è scalabile ed è impossibile avere un LSP che attraversa più operatori
- ✓ binding **dinamico** = scatenato in 2 modi:
 - **data/traffic driven** (innescato dall'arrivo di un pacchetto)
 - **control driven** (innescato dai messaggi di controllo) → la creazione di LSP control driven origina 2 tipi di LSP:
 - topology based = il router scopre che esiste una destinazione in base alla topologia della rete; gli LSR creano degli LSP per le destinazioni
 - creazione esplicita degli LSP (inizializzata dai LER; avviene on-demand)

- II. **LABEL MAPPING** = creazione della riga nella tabella di forwarding (tra ingresso e uscita); esegue l'**associazione tra un'etichetta di ingresso** (scelta dal LSR considerato) e **un'etichetta di uscita** (scelta dal downstream LSR), per riuscire a raggiungere il next hop in base al routing
 - III. **LABEL DISTRIBUTION** = l'**etichetta scelta deve essere comunicata ad 1 o più nodi vicini**; i protocolli usati (non compatibili tra loro) sono:
 - a. **BGP** (Border Gateway Protocol) = utilizzo di 1 protocollo di routing, solo topology based
 - b. **LDP** (Label Distribution Protocol) = poco usato perché sistema proprietario
 - c. **RSVP** (Resource reSerVation Protocol) = usato per l'allocazione di servizi integrati nelle reti
- **Protocolli di routing** (migliorati) → **determinano il percorso che sarà compiuto da un LSP**. Si usano **protocolli già esistenti** (OSPF, IS-IS, BGP-4), **ma modificati** per trasportare informazioni riguardo alle scelte di routing e porre dei **vincoli** (come capacità dei link, utilizzo dei link e dipendenza tra i link). Tali vincoli si usano per eseguire il **Constraint based routing**, ovvero un protocollo di routing di controllo che permette di scegliere il percorso più adatto in base ai vincoli imposti; per fare ciò si modificano i protocolli di routing sopra citati, ma queste **modifiche sono solo legate al constraint based routing** (quindi versioni **TE**, ovvero **modificate per il Traffic Engineering**). Le modalità di routing sono 2:
- **Hop by hop routing** = ciascun LSR decide il prossimo hop (ovvero il prossimo LSR) del percorso LSP [quindi come il routing IP tradizionale]; si procede così:
 - viene scelta una label per l'upstream link (label binding)
 - la label viene mappata all'indirizzo dell'interfaccia del prossimo LSR (next hop)
 - la label viene annunciata dal LSR che segue
 - **Explicit constraint based routing** = 1 switch sceglie il percorso per l'intero LSP, oltre al FEC. La scelta del percorso avviene usando il constraint based routing, ma non c'è 1 unico criterio per scegliere il percorso e possono esserci vincoli in conflitto. I **protocolli di label distribution** (distribuzione etichette) sono modificati per supportare informazioni su qual è il percorso scelto e si usano **CR-LDP** e **RSVP-TE** (usati con OSPF-TE e IS-IS-TE). Le modifiche permettono:
 - **Traffic Engineering** → consente la **ridistribuzione del traffico** non in base alla destinazione, ma in modo **omogeneo** (evitando la congestione). In MPLS, l'inoltro delle 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 vengono inoltrati in base alle **forwarding table** (e non alle routing table) realizzate con il mapping [anche se la routing table viene aggiornata, la forwarding table non cambia; quindi il traffico inviato inizialmente continua a seguire il percorso originale, solo quello nuovo segue il percorso nuovo (ciò accade perché il piano di controllo lavora sugli indirizzi IP, mentre il piano dati sulle etichette)].
MPLS è **IP-aware** (c'è un 1 solo control plan operativo su topologia fisica, quindi più scalabile); inoltre ci sono alcune estensioni:
 - **MP&S** (MPLS control plan su rete ottica)
 - **GMPLS** (Generalized MPLS, supporta vari tipi di rete)
 - **QoS** (qualità del servizio) → garantisce bandwidth, delay e burst size; consente una rete unificata che supporta tutti i tipi di servizi. Molte reti multiservizio usano un paradigma "ships in the night" (dove i protocolli ATM sono per servizi di ATM e l'MPLS control plan è usato per i servizi IP).
 - ⚠ Il supporto per QoS e i servizi real time su IP non sono ancora pronti.
 - **CoS** (classe di servizio) → insieme di parametri che descrivono il servizio richiesto (consente priorità relativa tra FEC diversi; supporta modello DiffServ, class traffic engineering, EF e AF)
 - **Fast fault recovery** (rapido recupero dei guasti) → una volta creato un LSP, se un link si rompe, il protocollo di routing se ne accorge e ricalcola la route, ma il piano dati continua a inviare pacchetti in base alla tabella di forwarding (che non cambia). Si può quindi creare a priori un'ulteriore LSP che fa da backup del link, usata al momento della rottura del link. Ci sono 2 modalità d'uso di questa LSP secondaria:
 - **Edge-to-edge re-routing** → se il link si rompe, si sostituisce **tutta la LSP originale**
 - **Link re-routing** → se il link si rompe, si sostituisce **solo il link rotto**

⚠️ Usare più etichette aiuta a generare una **gerarchia tra reti MPLS diverse**, che contribuisce alla scalabilità della rete. Questo è utile per instradare pacchetti in punti geografici molto lontani, quando un ISP non ha connettività diretta verso la destinazione.

Parliamo di **PHP (Penultimate Hop Popping)**: il penultimo nodo esegue il pop della label dal LSP, in modo da non doverlo fare il nodo di destinazione. Qui il LER (Label Edge Router) indirizza il pacchetto in base all'indirizzo IP. Il PHP può essere:

- **Implicito** = l'etichetta più esterna viene rimossa
- **Explicito** = l'etichetta non viene rimossa, ma ci si scrive "0" all'interno

Per qualsiasi router, sull'ultimo hop avviene lo swap sull'etichetta 0.

Parliamo di **MPLS VPN**: uno dei problemi fondamentali che le VPN devono risolvere nella connessione tra 2 reti private è quello di **usare reti private** (ovvero lo stesso range di indirizzi privati). Vediamo delle soluzioni:

- **PWE3 (Pseudo Wire Emulation End-to-End)** → consente l'uso di **VPN basate su MPLS al livello 2**, andando a creare LSP tra tutti i BR (router di confine) della rete MPLS che devono fornire connettività aziendale (è come unire 2 aziende mediante un cavo virtuale per l'inoltro dei pacchetti). Si usano 2 etichette: 1 interna (per il multiplexing di molti utenti/servizi sullo stesso AP) e 1 esterna (per identificare l'AP nella rete)
- **MPLS-based Layer 3 VPNs** → alternative di **livello 3** che consentono di **creare gli LSP automaticamente** tra i vari **PE router** (Provider Edge router) e poi crearne **ulteriori nella rete per il collegamento delle reti private** (connesse a tali PE) [funziona solo per il trasposto di pacchetti IP]. Per rendere automatico il processo, bisogna che i **CE router** (Customer Edge router) scambino informazioni di routing con i router PE.
Queste soluzioni sono **Provider Provisioned** con il vantaggio di non richiedere esperienza da parte dell'utente.
Ci sono 2 alternative:
 - **BGP** (Border Gateway Protocol) [inizialmente di Cisco, ora è il più usato] = permette di **trasportare informazioni su indirizzi che non sono indirizzi IP**. Con BGP, la MPLS VPN ha lo scambio di informazioni di routing tra gli edge e compie **Router Filtering** (ovvero ciascun PE determina quali route installare nella VRF). Qui vengono definiti **indirizzi VPN-IPv4**: ha indirizzi IP da 32 bit + i **Route Distinguisher** (64 bit per differenziare i percorsi dei diversi clienti nell'infrastruttura condivisa)
 - **Virtual Router** (VR) = i **PE** funzionano non come singolo router, ma come **tanti router virtuali diversi**

Le **componenti principali di un MPLS VPN** sono quindi:

- **CE router** = crea collegamenti con PE router; usa static routing o IGP (Interior Gateway Protocol)
- **P router** = ha route solo verso PE router; fa il setup degli LSP
- **PE router** = usato per scambiare le informazioni di routing; usa I-BGP (in soluzioni BGP) o IGP (in soluzioni VR). I PE hanno tabelle denominate VRF (VPN Routing & Forwarding Table) che consentono di creare delle tabelle di routing separate.

⚠️ I **vantaggi** di usare MPLS VPN sono:

- ✓ **Non necessita cifratura**, in quanto il traffico va sulla rete del service provider (di cui ci fidiamo)
- ✓ **Non ci sono vincoli sugli indirizzi**, ovvero possiamo anche usare indirizzi privati perché non viene visto l'indirizzo sul backbone [oltre all'indirizzo si specifica la VPN a cui appartiene (per questo si usa BGP, in quanto permette di trasportare informazioni su indirizzi che non sono indirizzi IP)]
- ✓ I CE non scambiano direttamente informazioni tra di loro
- ✓ I **customer non gestiscono il backbone** (infrastruttura di rete)
- ✓ Il **provider non ha un backbone virtuale per ogni cliente**
- ✓ **La VPN può passare tra provider diversi**

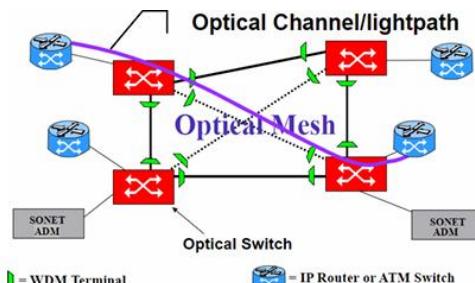
⚠️ Dato che **MPLS usa IPv4**, per **garantire il dual-stack** (IPv4 e IPv6) senza cambiare tutta l'architettura c'è **6PE**

7) RETE OTTICA

Con RETE OTTICA non si fa riferimento al trasferimento su un cavo di fibra ottica, ma alla **COMUNICAZIONE OTTICA** (ovvero reti dove i nodi sono collegati da fibre ottiche). Il concetto dietro alle reti ottiche è il **WDM** (Wavelength Division Multiplexing), cioè la trasmissione di frequenze diverse sullo stesso mezzo; 2 tipi:

- **DWDM** (Dense WDM) = trasmissione di 10/100 segnali sulla stessa fibra (aumento di capacità)
- **CWDM** (Coarse WDM) = più finestre (frequenze diverse) per la trasmissione; - lunghezze d'onda, + economico

Avendo tanti segnali ottici si può costruire un **optical switch** (commutatore ottico che riconosce i vari canali e li inoltra su fibre differenti) con l'idea di commutare canali ottici da una fibra di ingresso ad una fibra di uscita. Il commutatore ottico è semplice e ha permesso di abbassare i costi e di aumentare le prestazioni. Si può aggiungere o rimuovere multiplexing con **OADM** (Optical Add-Drop Multiplexer) nelle topologie ad anello. Ci sono diversi tipi di **optical switch**:



- OTTICI o ELETTRONICI:

- **Optical Core** → usa le **proprietà fisiche** di alcuni materiali per deflettere la luce da una fibra di ingresso verso una fibra di uscita; tipologie:
 - **Tilting mirrors (MEMS)** = apparati meccanici sensibili agli eventi esterni [+ usati]
 - **Superfici riflettenti olografiche** = usano il voltaggio
 - Materiali che cambiano **proprietà riflettenti** (in base a calore, pressione e voltaggio/corrente)Sono poco costosi, hanno **bitrate e segnale indipendenti** (alta scalabilità) e basso consumo di energia; hanno però alti prezzi di produzione e alta attenuazione (non c'è rigenerazione del segnale)
- **Electrical Core** → dispositivi elettronici che convertono un segnale ottico in elettrico effettuando lo switch dei bit ricevuti. Hanno **prezzi minori** rispetto ai MEMS, ma si perdono tutte le proprietà dei componenti ottici in quanto non si ha indipendenza tra bitrate e segnale

- CROSS CONNECT o SWITCH:

- **Cross connect** → smista alte **capacità di traffico** tra le varie parti di una rete, usando core ottici (MEMS) in configurazioni statiche (fixed) [usati nelle dorsali geografiche]
- **Fiber cross connect** → tutti i **segnali** da una fibra di ingresso vengono trasmessi su una fibra di uscita; vengono usati dei MEMS, ma richiedono molto tempo per la configurazione
- **Wavelength cross connect** → 1 o più **lunghezze d'onda** vengono mandate da una fibra di ingresso verso una fibra di uscita; c'è un WDM de-multiplexer + MEMS a separare le wavelength (prisma). Per la rigenerazione del segnale, la conversione **OEO** (optical-electrical-optical) consente di rigenerare elettronicamente

- WAVELENGTH CONVERSION:

complessa, realizzata con **EOE** (electrical-optical-electrical), richiede un **costo elevato** ed è poco matura. Non è trasparente per i dati, dunque **poco scalabile**. Dal punto di vista fisico, si comporta come una cassa di risonanza. Non richiede la stessa wavelength end-to-end.

⚠ Alcune **combinazioni** comuni sono:

- **Wavelength cross connect + Wavelength conversion** = 1 o più lunghezze d'onda da una fibra di ingresso verso 1 o più fibre di uscita. Si può usare un electrical core; il FEC (forward error correction) del segnale è più semplice e consente di ridurre il BER (bit error ratio). Altrimenti si potrebbe usare un optical core con OEO per la conversione, consentendo anche la rigenerazione del segnale
- **Dynamic Optical Switching** = ci può essere conversione di lunghezza d'onda e si può usare un optical core con OEO per la rigenerazione e per la wavelength conversion, oppure un electrical core con SONET/SDH con la possibilità di inserire multipli OEO per la rigenerazione. Si chiama dynamic perché la configurazione dello switch cambia dinamicamente, in base ad alcuni criteri (es. configurazione, ora del giorno, etc...)

Le reti ottiche consentono di ottenere **provisioning** (gestione delle risorse di rete per soddisfare i requisiti di un servizio/connessione) e **protezione dai guasti** (permettendo di trovare un percorso alternativo), ottenendo in questo modo una **DISTRIBUZIONE conveniente e flessibile** dei reti.

Di cosa hanno bisogno gli switch ottici?

- **Resource discovery** (sapere la **topologia**)
- Gestione della connessione/**signaling** (creazione dei **lightpath** e la loro modifica)
- **Routing distribuito**
- **Protezione e recupero** (per reti a maglia o anello)
- Servizio di **protezione per le classi**

Cosa è necessario garantire agli utenti?

- **Resource discovery** (gli indirizzi degli utenti devono essere **raggiungibili** con una rete ottica)
- **Gestione lightpath (signaling)**
- **Negoziare protection service classes**

L'unica soluzione per l'optical network **control plane** è **MPλS** (Multi-Protocol Lambda Switching), mentre si usa:

- OSPF, IS-IS, BG per **Resource Discovery**
- RSVP/LDP per **Signaling**

Parlando di **ROUTING**, nelle reti ottiche **gli utenti sono i router**, mediante i modelli:

- **Overlay** → la rete ottica fornisce connettività mediante i router, che vedono la rete come una scatola nera
- **Peer** → i router e gli switch partecipano con gli stessi protocolli di routing. I router conoscono la topologia della rete ottica e possono scegliere dei path preferiti rispetto ad altri

8) QoS (Quality of Service)

Le **applicazioni multimediali** potrebbero necessitare di alcuni requisiti di **QoS (QUALITÀ DEL SERVIZIO)** [es. latenza, banda o perdita di pacchetti] perché il flusso di dati è continuo ed il profilo generato deve essere = a quello ricevuto. Vediamo i **requisiti** legati ai vari tipi di servizio:

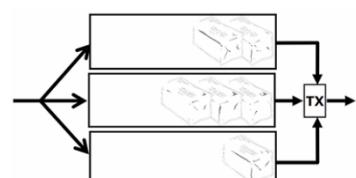
- **Streaming** → il flusso di dati deve essere continuo, è concessa tolleranza nella perdita di pacchetti, ma è necessario un **ritardo nullo/costante**
- **Interattività** → basso tempo di risposta
- **Trasmissione su larga banda** → quantità elevata di risorse quindi **capacità elevata** e molta memoria nei nodi

Il vero problema da risolvere è il **RITARDO (latenza)**: ai nodi potrebbe esserci congestione, quindi, se ci sono molti pacchetti che vogliono uscire tutti dallo stesso link, questi vengono messi in un **buffer**. Questo porta ad un **ritardo variabile a seconda del carico del nodo** (grave problema) [il ritardo di attraversamento dei nodi dipende non solo dalla quantità, ma anche dalla tipologia del traffico].

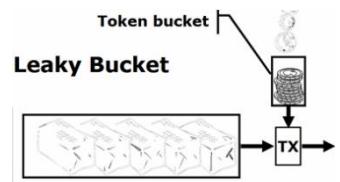
Sono necessari alta **capacità trasmissiva, buffer più grandi** nei nodi e capacità di commutazione (**switching**). A livello di pacchetto bisogna applicare **traffic shaping** (gestione della congestione), mentre a livello di flusso bisogna fare delle segnalazioni per riservare le risorse attraverso **RSVP** (Resource reSerVation Protocol) per IP e UNI per ATM. Inoltre bisogna applicare a priori il **network engineering** (dimensionando la rete in base al traffico previsto e limitando il numero di utenti nella rete) e il **traffic engineering** (controllare la distribuzione del traffico sulla rete) per dimensionare il caso peggiore.

Le **CONTROMISURE** che si possono realizzare nella rete sono:

1. **Classificazione del traffico** → identificare i pacchetti che necessitano QoS che appartengono ad una determinata comunicazione. È necessario un componente **hardware**, cioè un **ASIC** con della logica oppure le memorie **CAM** (dove si può dare un indirizzo e ricevere in risposta una **quintupla** [IP sorgente, IP destinazione, protocollo di trasporto, porta destinazione, porta sorgente] **per determinare il tipo di QoS**)
2. **Algoritmi di scheduling** → scegliere quali pacchetti prendere dal buffer per inviarli all'esterno. Bisogna analizzare tutti i pacchetti e inserirli in **code multiple** servite in base alla priorità, usando **algoritmi di scheduling** come Priority Queuing, Round Robin, CBQ, WFQ e Deadline Queuing



3. **Controllare il traffico che entra nella rete** → se i pacchetti in arrivo sono molti, bisogna **scartare dei pacchetti** (senza garantire QoS). Bisogna stabilire se il traffico in ingresso nella rete ha il **profilo adatto** per entrare; in particolare, si può adottare la tecnica del **leaky bucket** per controllare il flusso di ingresso



⚠ La Call Admission Control (CAC) consente di fare signaling e effettua reservation. Le scelte di routing sono prese in base alla disponibilità delle risorse e non solo in base alla topologia.

Esistono **2 standard per supportare il QoS**:

- **IntServ** → **garantisce il QoS**: effettua la prenotazione delle risorse (**RSVP**) garantendo QoS ad ogni singolo flusso; è **complesso e poco scalabile**, per questo non viene usato (nonostante sia implementato nei router)
- **DiffServ** → **non garantisce il QoS**: distingue il traffico in **classi** (identificate dal DS field). Se usato con network/traffic engineering e accesso controllato, si può limitare il n° di pacchetti nel buffer. **Non efficiente**, ma è **semplice e scalabile** (**sempre più usato**)

9) SDN

La domanda (cloud, big data, traffico mobile, IoT) e l'offerta di rete sta aumentando e il problema principale è sui pattern del traffico di rete. **QoS** (qualità del servizio) e **QoE** (qualità dell'esperienza) si stanno ampliando e quindi il carico di traffico va gestito in un modo più sofisticato. **L'architettura tradizionale** è basata su TCP/IP e si basa su indirizzamento a 2 livelli, routing basato sulla destinazione e controllo autonomo+distribuito; ha **limitazioni**:

- Architettura statica e complessa
- Politiche incoerenti
- Difficoltà di scalabilità
- Dipendenza dal fornitore

Un dispositivo di rete può essere diviso **logicamente** in **control plane** (dynamic configuration of the data plane tables [routing, OSPF]), **management plane** (user/admin configuration) e **data plane (forwarding)**.

Si introducono quindi le **SDN** (Software-Defined Networking), un nuovo paradigma di rete che separa **fisicamente** il **control plane** dal **forwarding plane** (data). Il **control plane** (**1 controller centrale**) gestisce più dispositivi **di rete** (**router più semplici, economici e veloci**) [quindi data plane semplice]. È presente **context-based forwarding** (decisioni di routing dinamiche). Vediamo quindi i **4 pilastri di SDN** [a dx architettura SDN]:

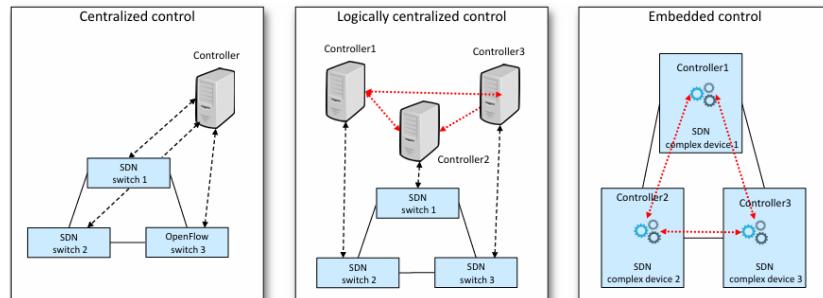
- **Separazione control plane e data plane** → routing arbitrario affidato ad un'applicazione software che runna su un **controller esterno** (separato dal piano dati). Il controller offre un'interfaccia aperta che consente a varie applicazioni di interagire con gli switch mediante un'**interfaccia standard** (all'inizio **OpenFlow**)

⚠ Il controllo centralizzato permette di cambiare facilmente la logica di routing usando nuove applicazioni e rende gli switch semplici e facilmente sostituibili. Inoltre l'interfaccia standard permette l'interoperabilità tra dispositivi di diversi fornitori

- **Piano dati semplice (“plumbing”)** → svolge principalmente forwarding, basato su una logica **match/azione** (con azione che è principalmente “forward to” o “drop”). Questo approccio è molto efficiente e permette di configurare la rete come un **insieme di “tubi” (plumbing)** attraverso cui fluisce il traffico (< **complessità**)

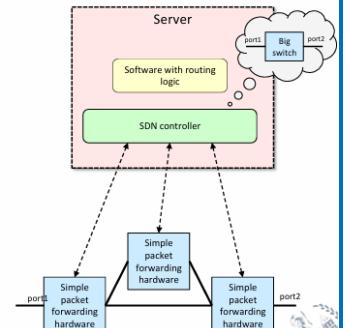
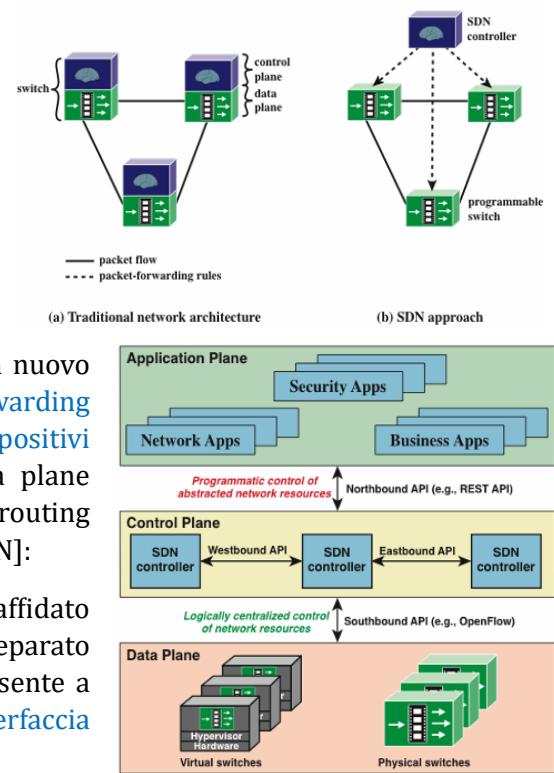
- **Controllo centralizzato** → il **controller SDN**, avendo una visione completa della rete, la gestisce in modo più semplice rispetto al controllo distribuito (routing tradizionale opera su ogni nodo, SDN ha gestione unificata). Il controller può rappresentare la rete anche come un **“big switch”** (nascondendo i dettagli interni e consentendo all'admin di configurare i nodi con comandi ad alto livello).

Ci sono diverse **implementazioni di controllo centralizzato**:



Non è necessario avere un controller fisico centralizzato in quanto con SDN può essere **logico** (esegue la logica di controllo in modo distribuito su server dedicati o sui dispositivi stessi).

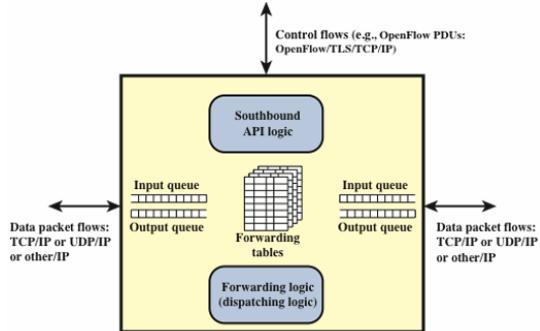
Il **controllo fisicamente distribuito, ma logicamente centralizzato** è più **robusto, veloce e scalabile** (perché alcune decisioni possono essere prese localmente), ma è **complesso** e ci sono dei **limiti posti dal teorema CAP** (Consistency, Availability, Partition tolerance) [le implementazioni attuali garantiscono Consistency]. Questa



architettura richiede **sincronizzazione** tra i dati tra le diverse istanze del controller [chi programma deve definire cosa debba essere condiviso tra le istanze].

- **Context-based forwarding** → all'inizio non presente, consente di **decidere dinamicamente il percorso del traffico in base al contesto del momento** ("run-time decisions") (es. un pacchetto può essere inoltrato al controller per stabilire la modalità di gestione del traffico in base al carico attuale di rete).

Qui possiamo vedere un **data plane network device**:



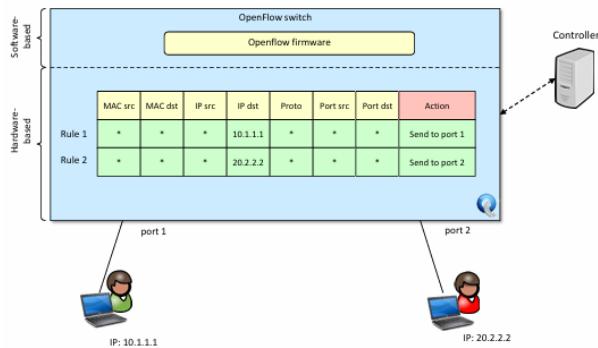
Parliamo di **OPENFLOW** (**OF**): è un protocollo che manda le regole di forwarding allo switch, riceve dallo switch i pacchetti che non matchano nessuna regola, ha statistiche sugli switch (facilitando il controllo centralizzato). È la 1^a implementazione di SDN e supporta i 3 pilastri di SDN prima citati (+ eventualmente può anche supportare il 4^o pilastro, ovvero il context-based forwarding). Riguardo al **context-based control path** (ricorda che abbiamo forwarding dinamico e adattabile al traffico; gli switch fanno da "cache" per il forwarding), OF consente:

- **Configurazione reattiva** = inserisce le regole quando necessarie, riducendo l'uso della tabella di flusso, ma introducendo un ritardo iniziale. I pacchetti che non matchano nessuna regola vengono mandati al controller e vengono inserite nuove entry nella tabella di flusso; se si perde la connessione di controllo, va in caciara
 - **Configurazione proattiva** = popola la tabella in anticipo, eliminando i ritardi ma richiedendo regole aggregate

OF permette **Routing**:

- Dettagliato (“fine-grained”) = ogni flusso gestito singolarmente (la tabella di flusso ha 1 entry per ogni flusso)
 - Aggregato (“aggregated”) = utile con più flussi di traffico (1 entry per gruppo di flussi)

Esempio di tabella di flusso:



Si definisce:

- **Flow Rule** = astrazione di un flusso di traffico (usa i campi di OF, può connettere 2 porte di switch OF, può attraversare più switch OF, è isolata dalle altre flow rules; può essere 1 entry di una flow table di uno switch)
 - **Flowmod** = messaggio OF per creare, modificare o cancellare una flow rule da un singolo switch

Vediamo la **struttura delle entry** di una flow table (ovvero la **struttura di una singola flow rule**):

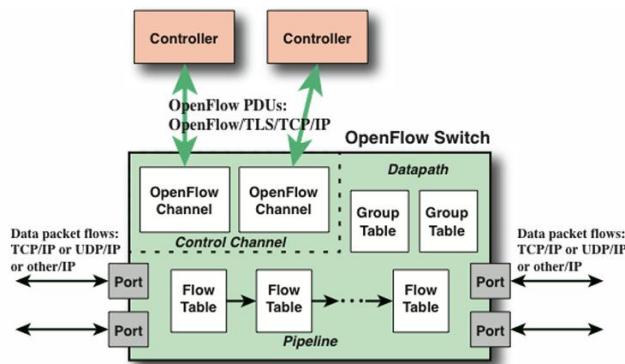
Match	Priority	Action(s)	Counters	Timeouts	Cookies	Flags
<p>Matching precedence of the flow entry.</p> <p>List of fields (with netmask) used to match against packets. These consist of the ingress port and packet headers, and optionally other pipeline fields such as metadata specified by a previous table.</p>		<p>List of actions that are applied to the packet in case of matching, which modify the action set or pipeline processing.</p>	<p>Statistics (packet/byte counters), updated in case of match.</p>	<p>Maximum amount of time or idle time before flow is expired by the switch.</p>	<p>Opaque data value chosen by the controller. May be used by the controller to filter flow entries affected by flow statistics, flow modification and flow deletion requests. Not used when processing packets.</p>	<p>Flags alter the way flow entries are managed, for example the flag OFPFF_SEND_FLOW_REM triggers flow removed messages for that flow entry.</p>

Riguardo al **MATCHING**, OF permette di verificare diversi header per decidere il routing (fa match sui livelli 2 [collegamento] – 4 [trasporto], ma non supporta il livello 7 [applicazione]). Ogni regola ha un valore di **priorità** che ne determina l'ordine di applicazione: si parte dalla priorità massima e, se non ci sono match, vengono esaminate le regole con priorità inferiore finché non si trova un match.

Le **azioni comuni** per le entry della tabella di flusso sono:

- Forward = inoltro alla porta, incapsulo + mando al controller oppure normal flow
- Drop (scarto)
- Overwrite header field
- Push/Pop fields

Vediamo un **OF switch** [dx]:



Parliamo ora del **Controller SDN**, una piattaforma software complessa che permette di controllare un'intera rete; funziona come un **sistema operativo di rete**, ispirato ai NMS (Network Management Systems) e agli OF controller. Rende la **rete programmabile** e permette l'estensibilità attraverso software addizionali (attraverso interfacce aperte e modulari, consente a sviluppatori terzi di aggiungere nuove funzioni e servizi di rete). Supporta dispositivi di rete eterogenei e domini di rete eterogenei; la sua architettura è composta da più livelli:

- ❖ Southbound (interfaccia sud) = gestisce interazione con gli switch
- ❖ SAL (Service Abstraction Layer)
- ❖ Core network services (es. routing)
- ❖ Northbound (interfaccia nord) = permette l'integrazione di servizi esterni (es. interfacce utente)

⚠ La virtualizzazione delle funzioni di rete (NFV) e l'approccio delle SDN stanno rendendo l'HW una commodity. (mentre il software open-source sta diventando competitivo)

⚠ L'approccio SDN ha come requisiti per una gestione flessibile e automatizzata della rete:

- ✓ **Adattabilità** (le reti devono rispondere dinamicamente alle esigenze delle applicazioni)
- ✓ **Automazione** (modifiche propagate automaticamente per ridurre errori)
- ✓ **Manutenibilità** (introduzione di nuove features deve avvenire senza interruzioni del servizio)
- ✓ **Gestione a livello di modello** (e non di singolo dispositivo)
- ✓ **Mobilità** (degli utenti e dei server virtuali)
- ✓ **Sicurezza integrata**
- ✓ **Scalabilità su richiesta**

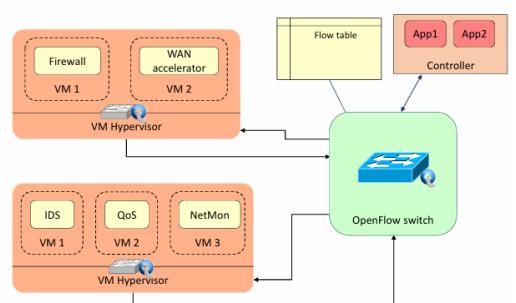
Vediamo ora la **NFV (Network Functions Virtualization)** [poter runnare funzioni di rete su standard hardware]. Con la **SFC** (Service Function Chain) si devono collegare vari **dispositivi HW dedicati** per fornire servizi aggiuntivi, utili particolarmente al margine della rete (edge); questo porta uso inefficiente dell'HW, interruzione del servizio (se aggiungo o tolgo dispositivi) e difficoltà nella differenziazione dei servizi.

Questi problemi si possono attenuare usando un OF switch per collegare tutti i dispositivi HW (**SFC + SDN**), facilitando l'instradamento ora gestibile via software e permettendo agilità nel provisioning dei nuovi servizi, manutenzione + affidabilità e personalizzazione dei percorsi per diversi clienti. Nonostante ciò, SDN presenta limiti nelle SFC a causa della **difficile partizione/condivisione di risorse HW** tra utenti.

NFV si basa su:

- **HW standard veloce COTS** (Commercial-Off-The-Shelf) [< costi]
- **Funzioni di rete basate su SW** (su server standard)
- **Virtualizzazione** [> flessibilità, > efficienza]
- **Standard API**

L'architettura **SFC+NFV** [dx img] consente di distribuire diverse funzioni di rete su macchine virtuali (VM) gestite da un **hypervisor**.

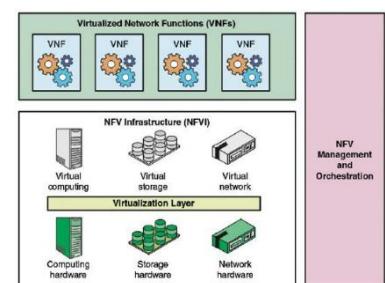


NFV permette la creazione di **SFG** (Service Function Graph → percorsi logici rappresentati con diverse funzioni di rete, personalizzabili in base alle esigenze).

⚠ I vantaggi di NFV sono:

- ✓ Virtualizzazione (uso ottimizzato delle risorse)
- ✓ Orchestrazione (gestione centralizzata di molti devices)
- ✓ Programmabilità (modifica dinamica delle funzioni di rete)
- ✓ Scalabilità dinamica (adatta la capacità in base al carico)
- ✓ Automazione (riduce l'intervento umano)
- ✓ Visibilità (monitoraggio delle risorse)
- ✓ Multi-tenancy (supporto per più utenti e servizi)
- ✓ Openness (completa scelta dei moduli del servizio)

⚠ ETSI NFV ISG (Industry Specification Group) = definisce i requisiti, i framework e le interfacce per NFV. Definisce quindi gli standard NFV e gestisce i gruppi di lavoro per garantire affidabilità, resilienza e sicurezza nelle implementazioni NFV



Il **framework NFV ad alto livello** comprende:

- **VNF** (Virtual Network Function) = funzioni di rete virtualizzate eseguite su HW generico
- **NFVI** (NFV Infrastructure) = HW e SW che consentono l'esecuzione delle VNF
- **MANO** (Management & Orchestration) = gestisce il ciclo di vita delle VNF e l'orchestrazione delle risorse

Proprio legato a ciò definiamo:

- **NF** (Network Function) = componente di rete con comportamento funzionale ben definito
- **VNF** (vedi sopra) = implementazione software di una NF su infrastruttura virtualizzata
- **User Service** = servizi offerti agli utenti finali
- **NFVI PoP** (Point of Presence) = posizione fisica dov'è collocata la NFVI
- **NFVI-PoP Network** = rete interna all'infrastruttura PoP
- **Transport Network** = connette i PoP tra loro e con le reti esterne
- **VNF Manager** = gestisce il ciclo di vita delle VNF (parte di MANO)
- **VIM** (Virtualized Infrastructure Manager) = gestisce le risorse virtuali (es. calcolo, archiviazione, rete...)
- **Network Service** = composizione di diverse funzioni di rete per fornire un servizio completo
- **NFV Service** = servizi di rete che usa NF con almeno 1 VNF
- **Deployment Behavior** = risorse necessarie per una VNF (VM, memoria, banda...)
- **Operational Behavior** = operazioni di ciclo di vita come avvio, stop, migrazione
- **VNF Descriptor** = descrive il comportamento di deployment + quello operativo
- **NFV Orchestrator** = automazione del deployment, gestione e coordinamento di VNF e NFVI (parte di MANO)

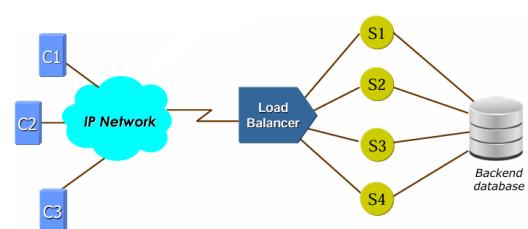
⚠ NFV può operare su 1 o più domini (consente gestione centralizzata di risorse distribuite su diversi center)

Ultimo punto è l'integrazione **NFV+SDN** (virtualizzazione delle funzioni di rete + controllo programmabile della rete); i vantaggi di questo approccio combinato sono:

- ✓ Automazione
- ✓ Flessibilità
- ✓ Efficienza (+ scalabilità)
- ✓ Provisioning rapido dei servizi
- ✓ Utilizzo migliore delle risorse

10) CDN

Come si possono **distribuire server farm efficienti**? Usando bilanciatori di carico (**load balancer**) front-end per distribuire le richieste sui server, con tutti i server che accedono allo stesso **database** di backend.



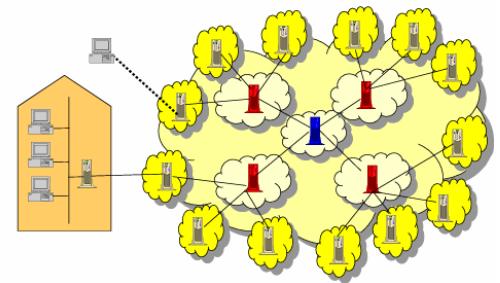
In passato si usavano **web caches** per memorizzare una copia locale degli oggetti HTTP richiesti più di recente (funzionavano come proxy server [tempo di risposta al client <]]). I vantaggi dell'uso di cache sono < traffico su Internet (< costi per i provider), > qualità dell'esperienza utente (< latenza, > velocità throughput) e supporto alla distribuzione di **contenuti localizzati** [es. pubblicità locali]. HTTP include “**caching information**” negli header (HTTP/1.0 usa “expires” e “pragma”; HTTP/1.1 ha “cache-control” e “e-tag”) e se il contenuto è:

- **expired** (scaduto) → usare la GET request per prendere il nuovo contenuto (con intestazione “if-modified-since”) ricevendo in risposta i codici (304 = “not modified”, 200 = “ok”)
- **not expired** → usare il contenuto in cache

⚠ All'inizio c'erano i **web proxy** che andavano configurati dall'utente, poi i **proxy trasparenti** (non richiedevano configurazione, ma avevano problemi con il traffico SSL rendendo ciechi gli ISP [necessità di accordi esplicativi tra i content providers e ISP])

Questo approccio però ha dei problemi a causa del **limiti delle server farm** (non risolvono la congestione della rete né riducono la latenza), **delle proxy cache** (usate solo dagli utenti del proprio ISP e non da tutta Internet) e dei **problemi di contabilità** (difficile per i content provider tracciare le visualizzazioni) [oltre al fatto che gran parte degli oggetti HTTP sono “**un-cacheable**”].

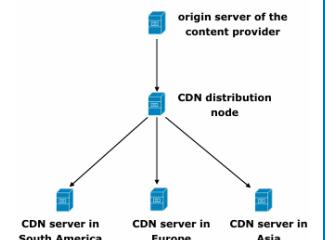
Per questo si introducono le **CDN (Content Delivery Networks)**, un **overlay network su server di cache distribuiti** (repliche o nodi di consegna) che migliora le prestazioni dei siti web tramite **riduzione della latenza** (bandwidth), **availability** e **bilanciamento del carico**. Le CDN vengono usate per hosting distribuiti, video on-demand, live streaming scalabile, contenuti dinamici e pubblicità custom.



⚠ CDN vs Caching proxies:

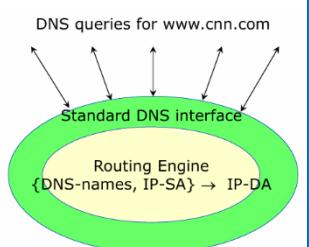
- Le cache sono usate dagli ISP, mentre le CDN dai content provider
- Le cache sono reattive, le CDN proattive
- Le CDN offrono controllo completo sui contenuti ai provider

Le aziende di CDN installano moltissimi server su Internet (in grandi data center o vicino agli utenti) perciò c'è bisogno di **Content Replication** tra server in **aree diverse**; quando il **content provider aggiorna il contenuto**, la **CDN aggiorna i server** (il server di origine riscrive le pagine per distribuire i nuovi contenuti mediante la CDN).

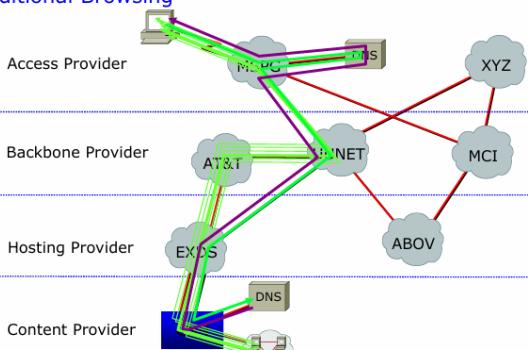


L'idea delle CDN è quindi la replicazione dei contenuti, ma come replicare, dove replicare e come indirizzare i client ai contenuti replicati? Intanto bisogna fare la **scelta del server migliore** per rispondere alle richieste (basata su carico, geografia, RTT o throughput; garantisce carico minimo, performance migliori e fault tolerance). Il **client viene diretto al server migliore** con **routing IP anycast** (trasparente per i client, complesso), con **HTTP redirect** (controllo granulare, ma aumento di carico) e con il **naming DNS** (riduce latenza, ma ha limiti nella granularità).

Viene quindi introdotta una **nuova DNS Server Architecture** che funziona come **motore di routing**: funziona usando una traduzione DNS dei nomi (www.ciao.com) in indirizzi IP ottimizzati. Le **DNS-based CDN** usano i nomi host per **reindirizzare il traffico ai "server-replica" migliori** (selezione della replica avviene nella traduzione nome → indirizzo IP); i server DNS diventano dei **“content routers”** che usano metriche (come RTT, carico, response time...) per calcolare la **replica routing table** con struttura **{DNS-name, IP-Source Address} → IP-Destination Address}**; tuttavia, la misurazione delle metriche non è semplice e non sempre gli indicatori di livello 3 (rete) sono utili.

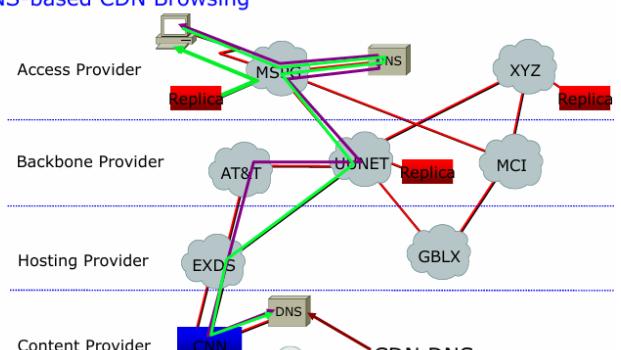


Traditional Browsing



VS

DNS-based CDN Browsing



⚠ Le CDN basate su DNS hanno come **limiti**:

- ✗ La granularità del reindirizzamento è **basata sul nome host, non sull'URL**
- ✗ I contenuti di grandi siti web non possono essere facilmente suddivisi in più cache
- ✗ È complesso usare lo stesso nome host per contenuti sia statici sia dinamici

Perciò si introduce **Akamai** (usa URL personalizzati per migliorare la distribuzione dei contenuti [parti statiche come le immagini]): crea **nuovi domini per ogni content provider** e i server DNS di Akamai sono **autoritativi** per questi nuovi domini; il content provider modifica i propri URL in modo che puntino ai domini Akamai (es. <http://cnn.com/a.gif> → <http://a128.g.akamai.net/7/23/cnn.com/agif>). Usare più domini consente ad Akamai di dividere ulteriormente il contenuto, ottimizzando la distribuzione del contenuto in base alla **geolocalizzazione**.

⚠ Se il contenuto non è cachato (**Content Not Cached**) il client fa la richiesta DNS, la richiesta passa attraverso i vari livelli di server DNS (root, high-level, low-level) di Akamai e poi viene indirizzata al server di Akamai più vicino per recuperare il contenuto; se invece il contenuto è cachato (**Content Already Cached**), la richiesta del client viene soddisfatta dal server Akamai locale (senza coinvolgere il server di origine).

Le DNS-based CDN possono essere **estese** usando **HTTP redirect** (> granularità): il reindirizzamento si può fare con una **Server Farm** (che fa il redirect) oppure con **SLB** (Server Load Balancer, che gestisce i redirect); richiede **sempre 2 richieste per ogni oggetto** (la 1^a al sito originale, la 2^a all'URL dell'oggetto). Si introduce però una latenza aggiuntiva.

L'evoluzione successiva delle CDN prevede l'uso di **URL per il routing**: richiede una terminazione TCP (complessa e costosa), introduce ritardo aggiuntivo (a causa dell'unico punto di terminazione TCP) e la terminazione può essere vicino al client o vicino al server.

→ Seminario 5G:

Il 5G deve supportare **3 scenari possibili** (amplia quindi il concetto di mobile broadband):

- eMBB (enhanced Mobile Broadband) = connessioni ad alta velocità per streaming e applicazioni a larga banda
- mMTC (massive Machine Type Communications) = tanti dispositivi IoT con bassa velocità ma alta efficienza
- uRLLC (ultra Reliable & Low Latency Communications) = comunicazioni con latenza bassa e alta affidabilità (es. guida autonoma e automazione industriale)

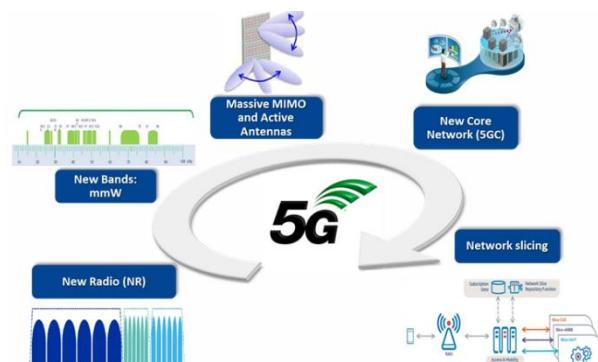
I principali **KPI** (Key Performance Indicators) per il 5G sono **data rate** (fino a 10-20 Gbps), **latenza** (< 1ms per uRLLC), **densità** di connessione (fino a 1 mln device/km²), **efficienza energetica** (100 volte migliore) e **mobilità** (fino a 500 km/h, utile per treni ad alta velocità).

La **Roadmap 3GPP** per il 5G è divisa in fasi:

- Rel-15 (fase 1 [2018]) = focus su eMBB e uRLLC. La fase iniziale (Early Drop) è stata completata nel 2017 e ha introdotto una soluzione NSA (Non StandAlone) dove il 4G LTE faceva da rete principale, mentre il 5G NR come supporto per la capacità aggiuntiva; è diventata StandAlone con Rel-15
- Rel-16 (fase 2 [2020]) = potenziamento delle funzionalità e preparazione alla standardizzazione IMT-2020

Parlando dei **Key Technological Enablers**, il 5G è abilitato da tecnologie come:

- ODFM filtrato (> efficienza spettrale) [detto **NR = New Radio**]
- Nuove bande di frequenza (incluse le mmWave [onde millimetriche]) → incluse 2 gamme di frequenza (FR1 = [450 MHz, 6 GHz]; FR2 = [24.25 GHz, 52.6 GHz]); le **mmWaves** hanno:
 - **pathloss** = a frequenze >, si hanno > perdite di potenza, ma usando antenne più grandi si compensa
 - **bloccaggio** = sono sensibili agli ostacoli (es. edifici, corpi umani), quindi necessaria > densità di rete
- MIMO massivo e beamforming (> copertura e > capacità) = usa array di antenne con elementi attivi che permettono il beamforming (orientamento del segnale) sia in orizzontale sia in verticale (> efficienza)
- 5GC = nuovo **Core di rete “virtualizzato”**, ovvero si ha transazione verso un’infrastruttura software (es. 5GC) per > flessibilità e > scalabilità
- Network slicing = creare reti virtuali dedicate (**slice**) all’interno di un’infrastruttura condivisa; ogni slice si può ottimizzare per specifiche esigenze (es. servizi per smartphone, applicazioni per auto o dispositivi IoT)



L'**Edge Computing** permette di eseguire servizi più vicini agli utenti finali, migliorando i tempi di risposta delle applicazioni, la sicurezza e le prestazioni; quindi i vantaggi sono **ottimizzazione della latenza** (riduce i tempi di interazione tra utente e rete, > performance), **nuovi servizi digitali** (abilitati dalla prossimità e dalla sicurezza), **elaborazione locale dei dati** (> affidabilità [utile per la sorveglianza video]).

Vediamo alcuni **esempi di utilizzo del 5G**:

- realtà aumentata (AR) e virtuale (VR) per manutenzione (operatori usano visori AR per assistenza sul campo)
- visite virtuali (es. musei con visori VR) e eventi immersivi a 360° da casa (sempre con VR)
- bodycam indossabili della polizia (trasmettono la registrazione live ad una sala di controllo)
- monitoraggio delle infrastrutture con IoT (sensori wireless)
- smart industry con manutenzione avanzata (IoT, analisi predittiva e AR) e AGV (Automated Guided Vehicles)

⚠ Come ci muoveremo verso il **6G**? TIM propone un approccio interdisciplinare con focus su efficienza energetica, sostenibilità, inclusione digitale, resilienza, sicurezza e esperienze estreme (sistemi cibernetici). Il 6G non sarà solo un miglioramento, ma risponderà alle esigenze della società futura, con l’obiettivo di connettere l’intelligenza distribuita. Si parla quindi di **AI/ML integrati** (automazione delle reti), **comunicazioni sub-THz** (> capacità di banda), **MIMO distribuito + nuove forme d’onda** (> efficienza spettrale) e **dispositivi a energia zero** (> sostenibilità). **Xexxa-X** è un’iniziativa europea per definire la visione del 6G (tratta 6 sfide: connessione dell’intelligenza, rete di reti, sostenibilità, copertura globale, esperienze estreme, affidabilità+sicurezza).