

</>

3. Livello di Rete

3.1 Servizi del livello di rete

3.1.1 Protocolli e servizi

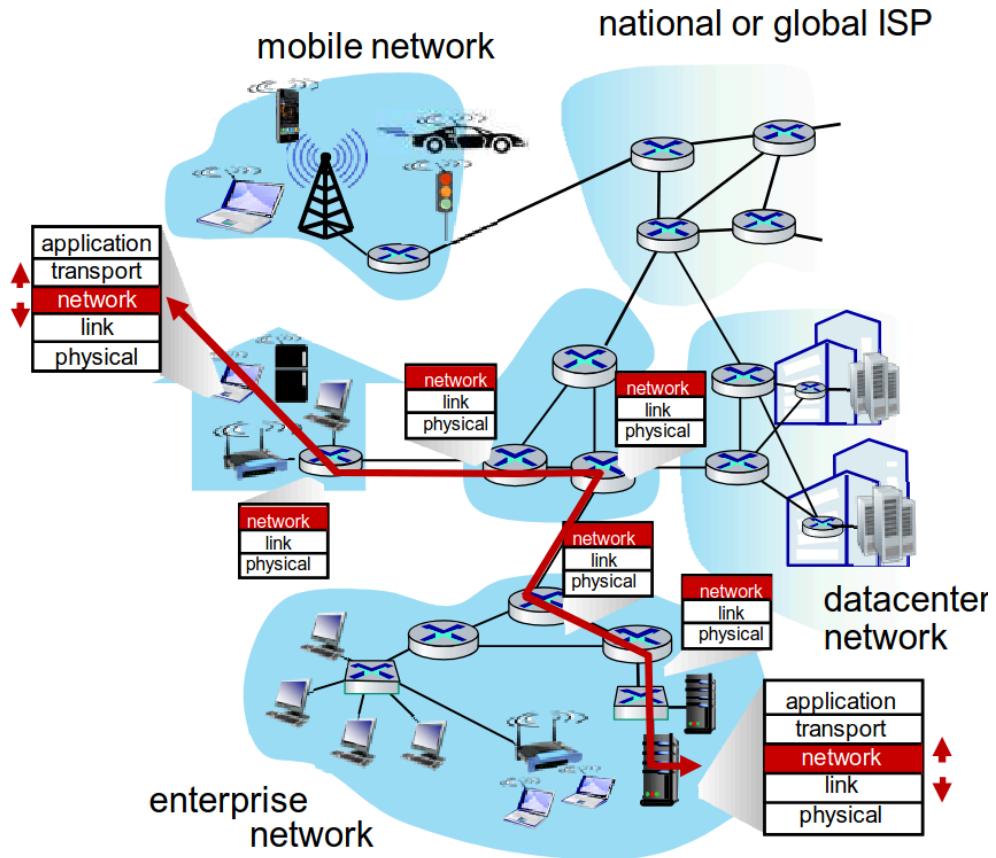
Il **livello di rete** ha il compito di fornire un servizio di comunicazione logica **da host a host**, gestendo il trasporto di pacchetti attraverso una rete costituita da router e link eterogenei. Questo avviene attraverso l'uso di **protocolli** specifici e la definizione di un **modello di servizio** offerto agli host.

Protocolli del livello di rete

I principali **protocolli del livello di rete** sono:

- **IP (Internet Protocol)**: è il protocollo cardine, responsabile dell'instradamento e del trasporto dei datagrammi da un host sorgente a uno di destinazione, attraverso uno o più router. Esistono due versioni: IPv4 (attualmente dominante) e IPv6 (progressivamente in adozione).
- **ICMP (Internet Control Message Protocol)**: utilizzato per la diagnostica e la gestione degli errori (es. `ping` e `traceroute`).
- **Protocolli di routing**: servono a determinare i percorsi ottimali per i datagrammi (es. RIP, OSPF, BGP), agendo nel **control plane**.
- **ARP, DHCP, NAT**: protocolli di supporto che operano a cavallo tra livello di rete e livelli inferiori o superiori.

Questi protocolli lavorano insieme per assicurare che i pacchetti possano essere consegnati correttamente, anche in reti complesse e in continua evoluzione.



Servizi offerti dal livello di rete

Il **modello di servizio** definisce il tipo di comunicazione logica fornita dalla rete agli host. In teoria, un livello di rete potrebbe offrire:

- **consegna affidabile** (no perdite),
- **ordinamento dei pacchetti**,
- **garanzia di ritardo massimo/minimo**,
- **garanzia di larghezza di banda**.

Tuttavia, **Internet adotta un servizio best-effort**, che **non garantisce nessuna di queste proprietà**:

- i pacchetti possono **essere persi o ritardati**,
- possono **arrivare fuori ordine**,
- il throughput può variare nel tempo.

Questa scelta progettuale è coerente con l'idea di **rete semplice e scalabile**, dove la complessità viene spostata ai livelli superiori, in particolare al livello di trasporto (es. TCP si occupa di affidabilità e ritrasmissione).

Routing e Forwarding: differenze fondamentali

Nel livello di rete è importante distinguere tra due funzioni chiave: **routing** e **forwarding**. Anche se collegate, svolgono ruoli concettualmente diversi:

- **Forwarding** (inoltro) è l'operazione di tipo **locale** svolta da ciascun router: prende un pacchetto in ingresso e lo inoltra sull'interfaccia di uscita corretta. Si tratta di un processo veloce e meccanico, eseguito per ogni pacchetto che attraversa il router.

- **Routing**, invece, è un'operazione **globale e strategica**: consiste nel determinare i **percorsi** che i pacchetti dovranno seguire attraverso la rete. Il routing genera le **tabelle di instradamento** usate poi dal forwarding.

Una metafora utile è quella del viaggio in auto:

- **Routing** è il processo di pianificare l'intero tragitto sulla mappa.
- **Forwarding** è la decisione, a ogni incrocio, su quale direzione prendere.



forwarding



routing

In una rete IP, il routing è realizzato da **protocolli di routing** (come OSPF, BGP), che popolano la **tabella di forwarding**.

3.1.2 Data Plane e Control Plane

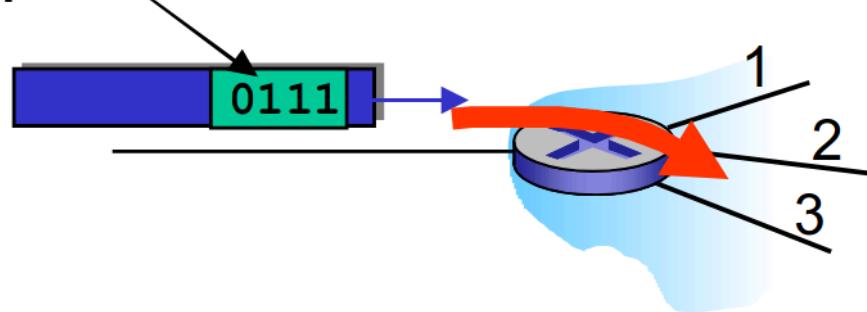
Il funzionamento del livello di rete in un router moderno è organizzato logicamente in due componenti distinte ma complementari: il **data plane** e il **control plane**.

Data Plane

Il **data plane** è responsabile del trattamento dei pacchetti in tempo reale. Quando un pacchetto arriva su un'interfaccia del router, è il data plane che decide **dove inoltrarlo**, consultando la **tabella di forwarding**. Questo processo è locale e ad alte prestazioni: deve essere rapido, efficiente e capace di gestire grandi volumi di traffico.

In molti router, il data plane è implementato direttamente in **hardware dedicato** (come ASIC – Application-Specific Integrated Circuit), proprio per garantire velocità elevate e ridurre la latenza.

values in arriving packet header



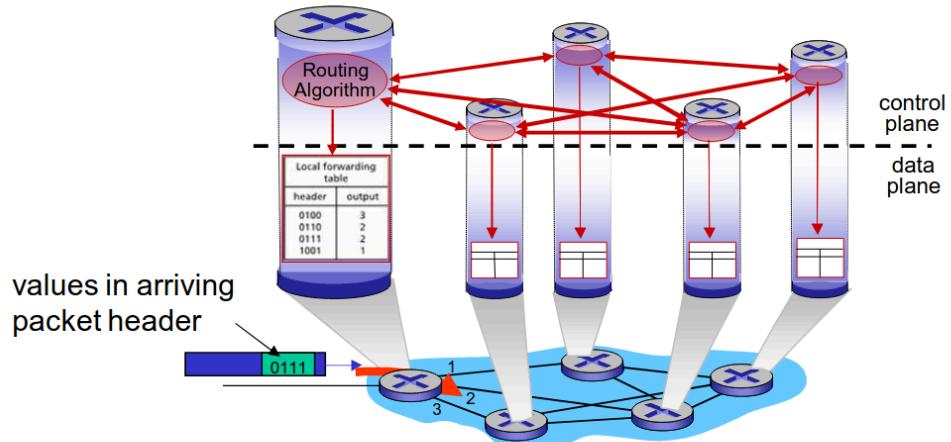
Control Plane

Il **control plane** ha invece una funzione più strategica e globale: si occupa di **costruire e aggiornare le tabelle di forwarding**, determinando il **percorso ottimale** che un pacchetto deve seguire nella rete. Questa funzione viene realizzata tramite i **protocolli di routing** (come RIP, OSPF, BGP) che comunicano tra i router per diffondere informazioni sulla topologia e le condizioni della rete.

Il control plane è tradizionalmente implementato in **software**, nel **processore di routing** del router, ed è separato logicamente (e spesso fisicamente) dal data plane.

Pre-router Control Plane

In architetture tradizionali, il control plane è **distribuito**: ogni router esegue un'istanza del protocollo di routing e prende decisioni in autonomia, collaborando con gli altri router per costruire una visione coerente della rete. Questo approccio è robusto e scalabile, ma introduce complessità nella gestione della rete, specialmente in scenari dinamici o su larga scala.



Software Defined Networking (SDN)

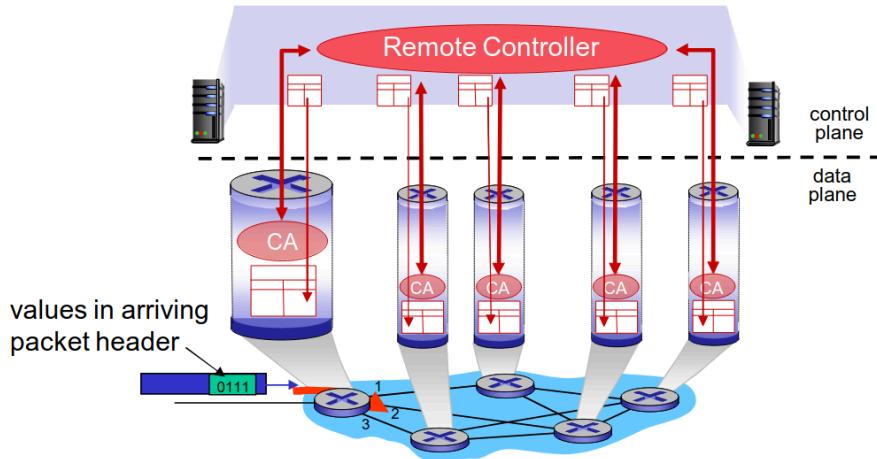
Con l'avvento delle **Software Defined Networks (SDN)**, si è introdotto un **modello centralizzato** di controllo. In questo paradigma:

- il control plane viene **estratto dai router e accentratato in un controller SDN**,

- i router diventano semplici "forwarding devices" che obbediscono alle istruzioni del controller.

Il controller ha **una visione globale** della rete e può programmare dinamicamente i percorsi, semplificando la gestione e abilitando funzionalità avanzate come il bilanciamento del carico, la sicurezza adattiva e l'orchestrazione automatica.

In SDN, la comunicazione tra il controller e i dispositivi di forwarding avviene tramite **protocolli come OpenFlow**, che permettono di modificare in tempo reale le regole di inoltro.

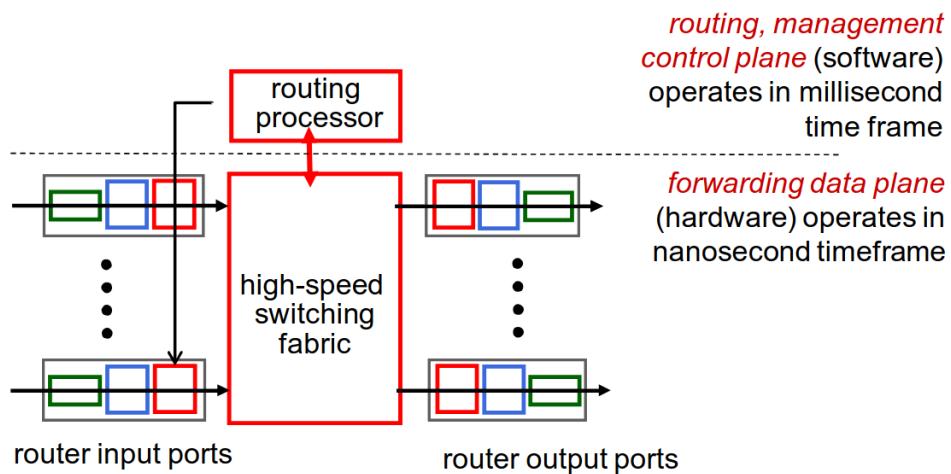


3.2 Router

3.2.1 Componenti del router

Un **router** è un dispositivo centrale del livello di rete, responsabile dell'**inoltro dei pacchetti** tra i nodi di una rete. Per svolgere correttamente questa funzione, il router è organizzato attorno a **quattro componenti principali**, ciascuno con un ruolo specifico nel processo di instradamento:

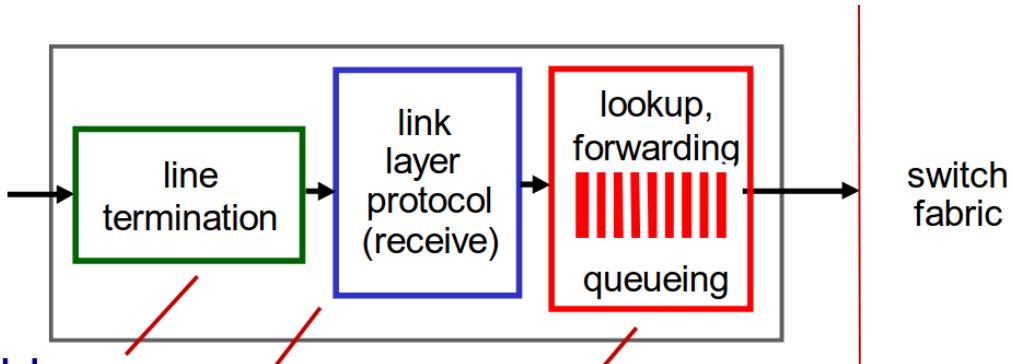
1. **Le porte di ingresso (input ports)**: sono il punto di ricezione dei pacchetti. Oltre a svolgere funzioni di livello fisico e di collegamento (come la decodifica dei bit ricevuti), iniziano anche l'elaborazione IP del pacchetto, determinando l'interfaccia di uscita tramite la tabella di forwarding.
2. **La fabric di commutazione (switching fabric)**: è il "cuore" del router, ovvero il sistema che trasferisce fisicamente i pacchetti dalla porta di ingresso alla porta di uscita corretta. La sua architettura determina in gran parte le prestazioni complessive del router.
3. **Le porte di uscita (output ports)**: ricevono i pacchetti dalla fabric e li preparano per l'invio sul link successivo. Qui può avvenire una temporanea memorizzazione (buffering), oltre all'aggiunta di eventuali header di livello data-link.
4. **Il processore di routing**: è la componente che gestisce le funzioni di controllo del router, come l'esecuzione dei protocolli di routing, la costruzione della tabella di forwarding e la gestione della configurazione e del monitoraggio.



Insieme, questi quattro blocchi lavorano in modo coordinato per permettere al router di eseguire il **forwarding efficiente e corretto** dei pacchetti. Le loro prestazioni e il modo in cui sono implementati (in hardware o software) influenzano direttamente la capacità di throughput, la latenza e l'affidabilità del dispositivo.

3.2.2 Porte di ingresso

Le **porte di ingresso** (input ports) sono il primo punto di contatto tra un pacchetto in arrivo e il router. La loro funzione è più complessa di quanto sembri a prima vista, perché integrano sia componenti dei **livelli inferiori** (fisico e data-link), sia funzionalità proprie del **livello di rete**.



Funzioni principali

Una porta di ingresso esegue le seguenti operazioni:

1. **Ricezione a livello fisico**: decodifica i segnali ricevuti dal collegamento in ingresso, traducendoli in una sequenza di bit.
2. **Elaborazione a livello data-link**: verifica l'integrità dei dati (es. controllo CRC), rimuove l'intestazione del livello 2 e isola il pacchetto IP.
3. **Lookup e forwarding**: analizza l'indirizzo IP di destinazione e consulta la **tavella di forwarding** per determinare a quale interfaccia di uscita inviare il pacchetto.
4. **Inoltro del pacchetto alla switching fabric**, in base alla decisione presa.

In alcuni router, il **lookup della tabella di forwarding** avviene direttamente nella porta di ingresso (approccio **decentralizzato**), mentre in altri viene eseguito dal **processore centrale** (approccio **centralizzato**). I router ad alte prestazioni privilegiano il primo metodo, in quanto più veloce e parallelo.

Accodamento in ingresso e congestione

Un aspetto critico delle porte di ingresso riguarda il **buffering**. Quando la **switching fabric è occupata** o più pacchetti devono accedere alla stessa uscita, i pacchetti in ingresso non possono essere trasferiti immediatamente, quindi devono essere **accodati temporaneamente** nella porta di ingresso.

Questo fenomeno, noto come **input queuing**, può portare a congestione e perdita di pacchetti se il buffer si satura.

Un caso limite è quello del **head-of-line blocking (HOL blocking)**: quando il primo pacchetto nella coda attende l'uscita, ma i successivi potrebbero essere inoltrati verso porte libere. Tuttavia, l'intera coda è bloccata dal pacchetto in testa, causando una **riduzione significativa del throughput**.

Longest Prefix Matching

Durante il lookup, il router utilizza una tecnica chiamata **longest prefix matching**, che confronta l'indirizzo IP di destinazione con i **prefissi CIDR** presenti nella tabella di forwarding.

Si seleziona il prefisso più lungo (cioè quello più specifico) che matcha con l'indirizzo IP.

Esempio

Supponiamo di avere tre voci nella tabella:

- 11001000 00010111 ***** (prefisso /16)
- 11001000 00010111 0001***** (prefisso /20)
- 11001000 00010111 00010000 ***** (prefisso /24)

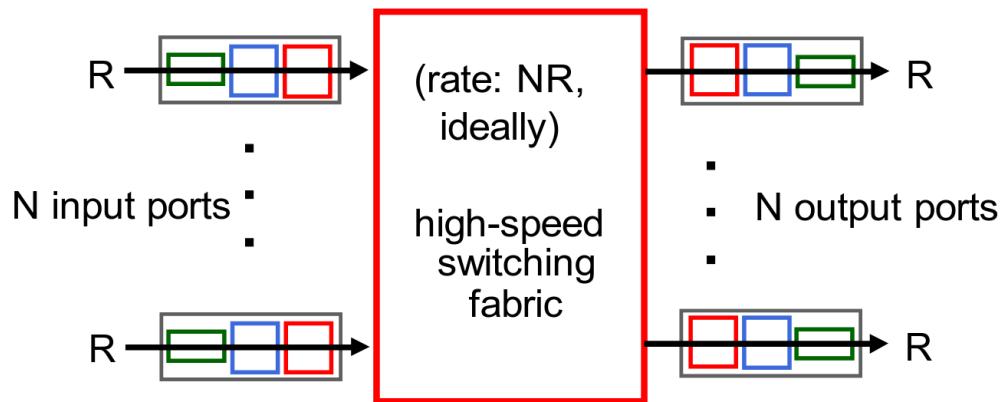
Se l'indirizzo IP del pacchetto è `11001000 00010111 00010000 10101010`, tutte le voci matchano, ma viene selezionata quella **/24**, la più specifica.

Nella realtà, questo metodo è utilizzato poco perché si utilizzando delle memorie particolari chiamate **TCMAs**, **Ternary content addressable memories** che estraggono l'indirizzo di destinazione in un ciclo di clock, indipendentemente dalla dimensione della tabella.

3.2.3 Switching Fabric

La **switching fabric** è il cuore interno del router, il componente che collega logicamente ogni **porta di ingresso** a ogni **porta di uscita**. Il suo compito è prendere il pacchetto ricevuto da un input port, una volta determinata la sua interfaccia di uscita, e **trasferirlo fisicamente** alla destinazione interna corretta.

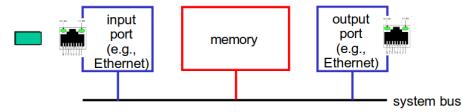
Una switching fabric veloce è fondamentale per garantire alte prestazioni di throughput nel router. Le prestazioni del router sono spesso limitate non dalla velocità dei link esterni, ma dalla **capacità interna della fabric** di trasferire pacchetti.



Architetture della switching fabric

Basata su memoria

E' l'architettura host più semplice e comune. Utilizzata dai computer tradizionale con lo switching sotto il controllo diretto della CPU. Tutto è gestito dalla memoria centrale della macchina.

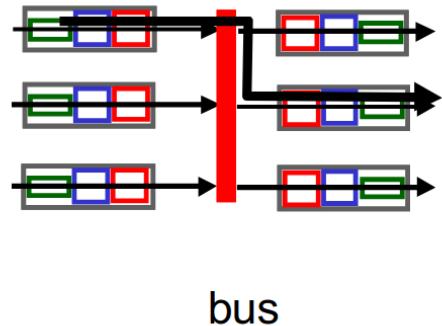


- **Pro:** implementazione semplice e basso costo
- **Contro:** velocità limitata dalla banda della memoria

Basata su bus

È l'architettura più semplice e meno costosa. Tutte le porte sono collegate a un **bus interno condiviso**. I pacchetti viaggiano uno alla volta sul bus. Un controller regola l'accesso, evitando collisioni.

- **Pro:** implementazione semplice, basso costo.
- **Contro:** throughput limitato a un pacchetto per volta; non scala bene con molti input/output.

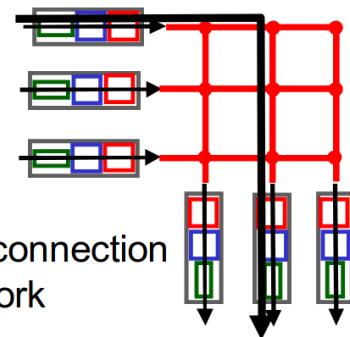


Il massimo throughput teorico è **pari a R** (dove R è la velocità di un singolo link).

Basata su interconnessione a commutazione (crossbar)

Questa architettura collega direttamente ciascuna porta di ingresso con ciascuna porta di uscita attraverso una **matrice di interconnessione**. Diversi pacchetti possono essere trasferiti simultaneamente, purché non puntino alla stessa uscita.

- **Pro:** più trasferimenti paralleli → maggiore velocità.
- **Contro:** complessità hardware superiore, specialmente per molti ingressi/uscite.

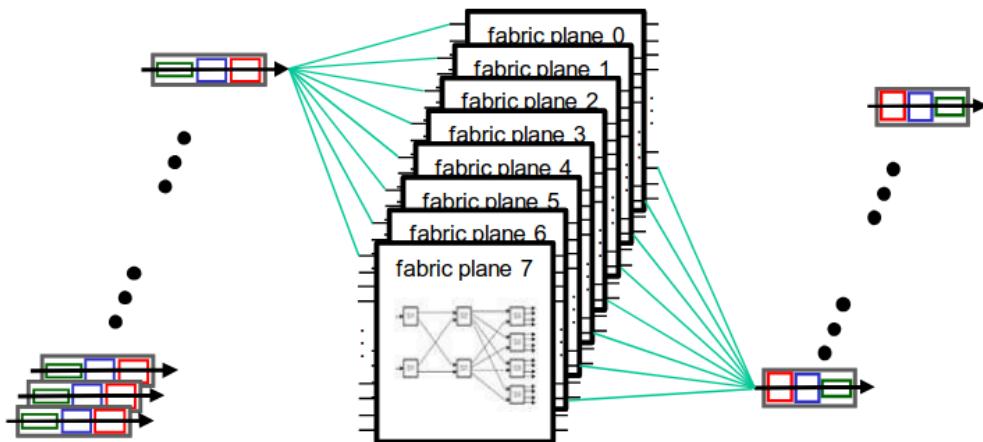


Con questa architettura, il throughput può arrivare fino a $N \cdot R$, dove N è il numero di porte.

Architetture multistadio (es. Clos network)

Utilizzate nei router di fascia alta, le reti multistadio usano una combinazione di switch più piccoli collegati tra loro, garantendo **scalabilità e parallelismo**. Permettono trasferimenti simultanei anche in presenza di carichi elevati, con logiche intelligenti di scheduling.

- **Pro:** altamente scalabili, tolleranti ai guasti.
- **Contro:** progettazione e controllo più complessi.



Gestione del trasferimento

Nel trasferimento del pacchetto da input a output port, è essenziale evitare:

- **conflitti**, quando più ingressi puntano alla stessa uscita,
- **ritardi**, dovuti a congestione interna,
- **blocchi**, soprattutto in presenza di accodamenti non ottimizzati.

Alcune architetture introducono anche **buffer interni alla switching fabric**, per migliorare l'efficienza.

3.2.4 Porte di uscita

Le **porte di uscita** (output ports) rappresentano l'ultima tappa nel percorso interno di un pacchetto dentro il router, subito prima della trasmissione sul link verso il prossimo nodo della rete. Dopo essere stati trasferiti

dalla switching fabric, i pacchetti arrivano alla porta di uscita dove possono essere **bufferizzati** in attesa che il canale fisico sia libero, quindi trasmessi.

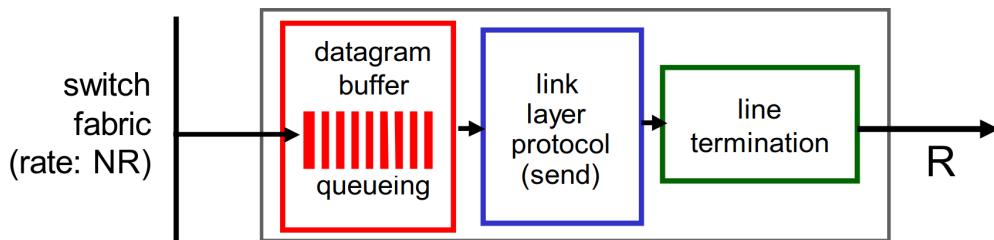
Funzioni principali

Una porta di uscita svolge tre funzioni fondamentali:

1. **Accodamento (queueing)**: se più pacchetti sono destinati alla stessa uscita, vengono temporaneamente memorizzati in una coda. Questo accade tipicamente quando il ritmo di arrivo supera la capacità di trasmissione sul link.
2. **Elaborazione di livello 2**: viene costruito un nuovo frame di livello data-link, con i relativi header (MAC, CRC, ecc.), in base al protocollo utilizzato sul link di uscita (Ethernet, PPP, ecc.).
3. **Trasmissione sul link**: una volta che il pacchetto è pronto e la coda lo consente, viene inviato attraverso il canale fisico verso il prossimo router o host.

Output Queueing

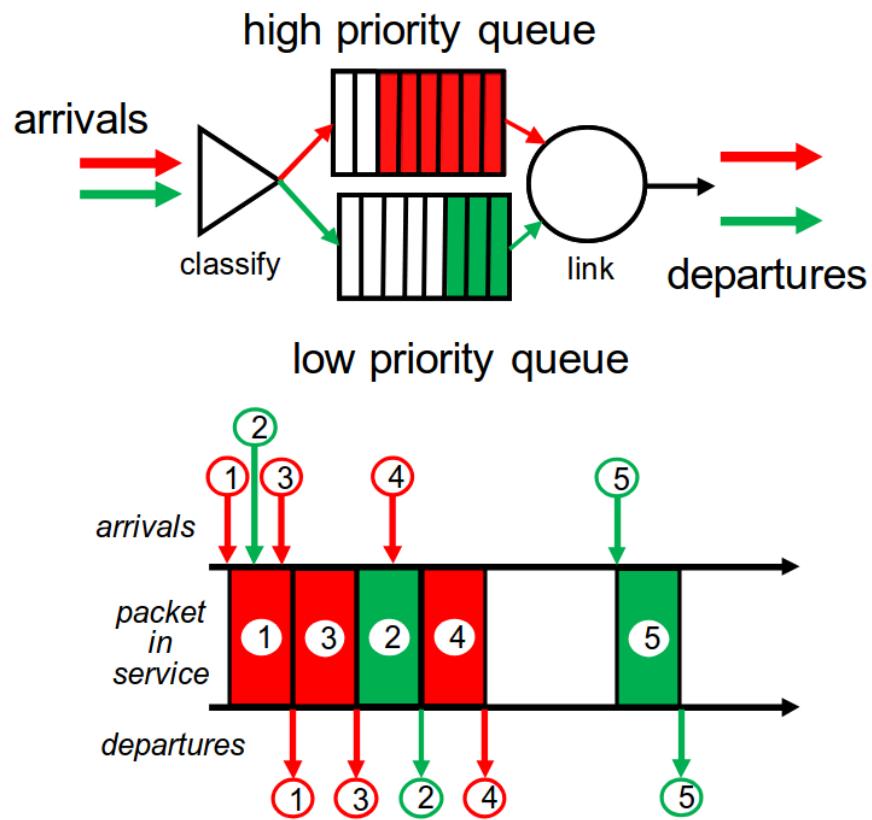
L'accodamento alla porta di uscita può portare a **congestione**, soprattutto nei router ad alta velocità dove più pacchetti possono essere instradati contemporaneamente verso la stessa interfaccia.



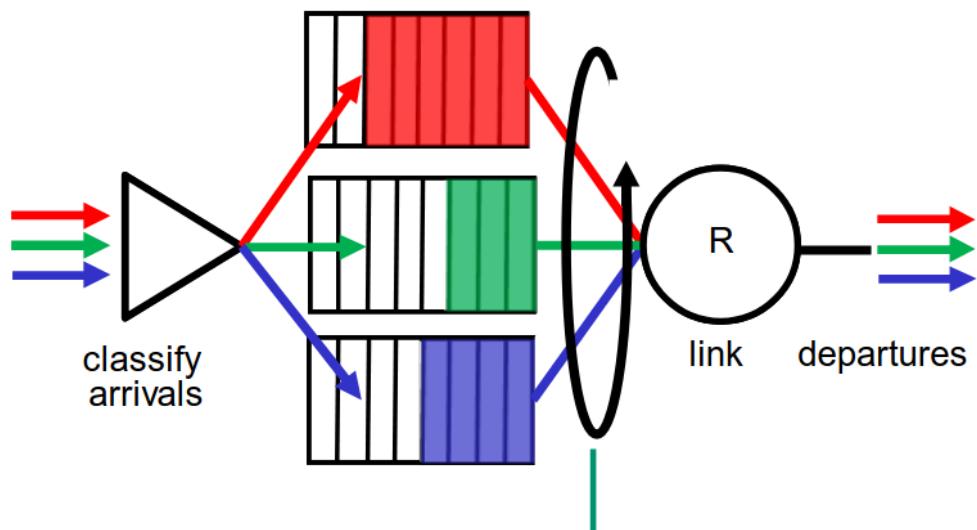
In scenari ad alta contesa, la coda può diventare lunga e il buffer può saturarsi. Quando ciò accade, i pacchetti in arrivo vengono **scartati** (drop), causando perdita.

La gestione della coda può avvenire con diverse **strategie di scheduling**, tra cui:

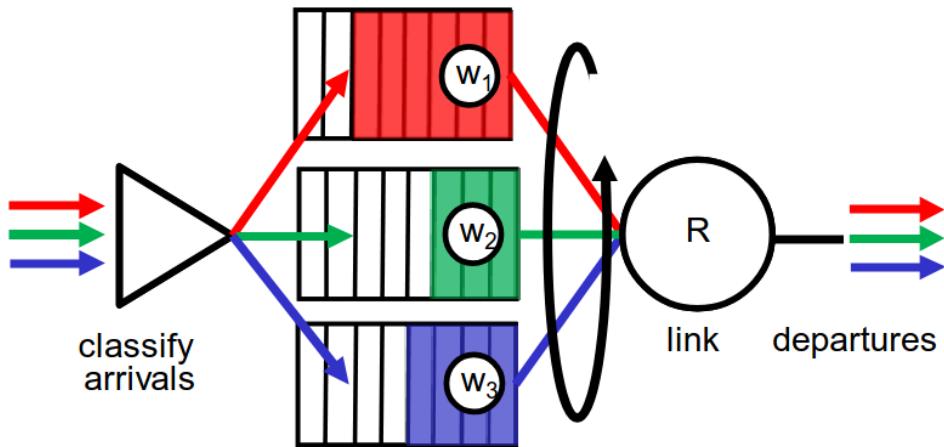
- **FIFO (First In First Out)**: semplice, ma può penalizzare i flussi lenti.
- **Priority queueing**: alcuni pacchetti (es. VoIP) hanno precedenza su altri.



- **Round-Robin:** algoritmo di scheduling dove ogni coda (un flusso di dati da una porta) riceve lo stesso trattamento.



- **Weighted Fair Queueing (WFQ):** Evoluzione dei round robin dove ogni coda riceve un peso associato alla sua priorità o necessità di banda.



- **RED (Random Early Detection)**: più che un algoritmo di scheduling qui si parla di una tecnica di gestione anticipata della congestione. Il router monitor la lunghezza media della coda, quando questa supera una soglia minima inizia a scartare pacchetti casualmente e man mano che la coda cresce il drop dei pacchetti diventa più frequente. Quest'idea funziona se viene utilizzato TCP, in modo che i mittenti si accorgano in anticipo della congestione.

Blocchi critici e congestione interna

Anche se il buffering avviene nelle porte di uscita, il problema può **propagarsi a monte**, bloccando la fabric e limitando le prestazioni complessive del router. Per esempio:

- se l'accodamento rallenta la trasmissione,
- e la fabric non può svuotare le porte di ingresso abbastanza in fretta,
- si ha una **retroazione negativa**, con rallentamento generale.

Questo tipo di blocco, noto come **backpressure**, può ridurre drasticamente il throughput del router.

3.3 IPv4

3.3.1 Assegnazione degli indirizzi IP

Ogni dispositivo collegato a una rete IP ha un **indirizzo IP univoco**, che lo identifica all'interno del sistema globale. Questo indirizzo è un **numero di 32 bit** (per IPv4) e viene solitamente rappresentato in **notazione decimale puntata** (es. 192.168.1.5), dove ogni gruppo rappresenta un byte.

Parte di rete e parte di host

Un indirizzo IP è concettualmente diviso in due sezioni:

- la **parte di rete** identifica la sottorete a cui il dispositivo appartiene;
- la **parte di host** identifica un dispositivo specifico all'interno di quella rete.

Questa distinzione è logica, non visibile direttamente dall'indirizzo, e dipende dalla maschera di rete (subnet mask) o dal prefisso CIDR (es. /24).

Ad esempio:

- L'indirizzo 200.23.16.8/23 indica che i **primi 23 bit** formano la parte di rete;
- i **rimanenti 9 bit** identificano i singoli host all'interno di quella rete.

In questo caso, la rete include tutti gli indirizzi da `200.23.16.0` a `200.23.17.255`.

Indirizzi speciali

Nel contesto dell'assegnazione degli indirizzi IP, esistono valori riservati:

- **Tutti 0 nella parte host** → identifica l'**indirizzo di rete**.
- **Tutti 1 nella parte host** → indica l'**indirizzo di broadcast locale**, usato per inviare un pacchetto a **tutti gli host della rete**.

Esempio:

- `223.1.1.0/24` è l'indirizzo della rete.
- `223.1.1.255` è il broadcast.
- Indirizzi validi per gli host: da `223.1.1.1` a `223.1.1.254`.

Importanza del prefisso

La conoscenza della **lunghezza del prefisso (CIDR)** è cruciale:

- determina **quali indirizzi appartengono alla stessa rete**;
- consente al router di **decidere in modo efficiente** il forwarding dei pacchetti (con il **longest prefix matching**, visto in precedenza).

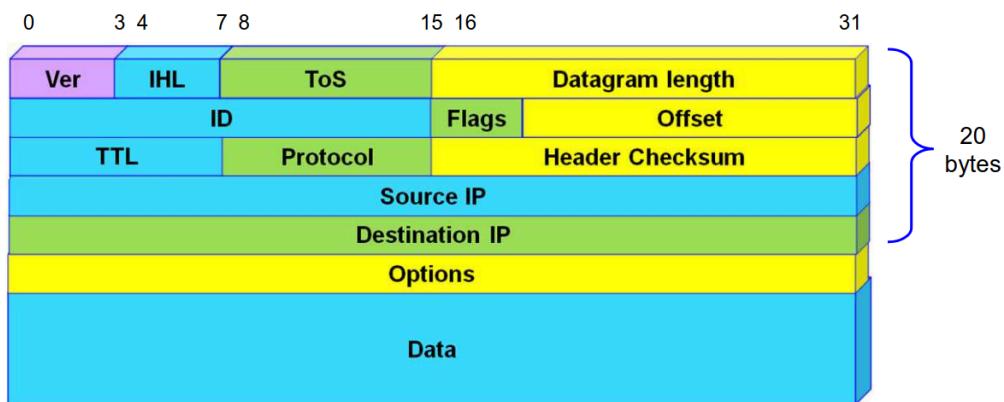
3.3.2 Datagramma

Il **datagramma IP** è l'unità fondamentale con cui il livello di rete trasferisce dati da un host sorgente a un host destinatario. Ogni pacchetto trasportato a livello di rete ha il formato di un datagramma IP, che incapsula dati provenienti dal livello di trasporto (come un segmento TCP o un datagramma UDP) e li arricchisce con informazioni necessarie per l'instradamento.

Struttura del datagramma

Il datagramma è composto da due sezioni:

- un'**intestazione (header)**, che contiene metadati per la consegna del pacchetto;
- un **campo dati**, che trasporta il payload vero e proprio.



La **struttura dell'intestazione IPv4**, ha una dimensione tipica di 20 byte (può estendersi in presenza del campo opzionale). Di seguito i campi più rilevanti:

- **Version (4 bit)**: identifica la versione del protocollo IP. Per IPv4, è 4.
- **Header Length (4 bit)**: lunghezza dell'intestazione in parole da 32 bit.

- **Type of Service (8 bit)**: campo utilizzato per indicare la priorità o il tipo di servizio desiderato (oggi spesso usato per QoS con il campo Differentiated Services).
- **Total Length (16 bit)**: lunghezza totale del datagramma (header + dati).
- **Identification (16 bit)**: usato per distinguere datagrammi in contesti in cui possono esserci più frammenti.
- **Flags (3 bit)**: includono controlli come il bit "Don't Fragment".
- **Fragment Offset (13 bit)**: indica dove un frammento si colloca nel datagramma originale (non usato se non c'è frammentazione).
- **Time to Live – TTL (8 bit)**: contatore che si riduce a ogni passaggio in un router; serve per evitare che i pacchetti si perdano in loop infiniti.
- **Protocol (8 bit)**: indica quale protocollo di livello superiore è encapsulato (es. TCP = 6, UDP = 17).
- **Header Checksum (16 bit)**: consente di rilevare errori nell'intestazione.
- **Source Address (32 bit)**: indirizzo IP dell'host mittente.
- **Destination Address (32 bit)**: indirizzo IP dell'host destinatario.
- **Options + Padding**: campo opzionale, raramente usato, serve per funzioni aggiuntive.

Incapsulamento

Il datagramma IP non viaggia da solo. Viene incapsulato in una **trama di livello data-link** (ad esempio, Ethernet) per la trasmissione fisica tra due nodi. Ogni passaggio da un router a un altro implica un re-incapsulamento, ma il **datagramma IP rimane invariato** tra mittente e destinatario.

3.3.3 Frammentazione

Uno dei limiti fondamentali del livello di rete è che i pacchetti IP non possono sempre essere trasmessi interamente su tutti i collegamenti. Ogni tipo di collegamento (Ethernet, Wi-Fi, PPP, ecc.) ha infatti una dimensione massima per la trasmissione: la **MTU (Maximum Transmission Unit)**. Se un datagramma IP è **più grande della MTU del collegamento**, allora deve essere **frammentato**.

Perché serve la frammentazione?

IP è un protocollo progettato per funzionare su reti eterogenee. Può capitare che un pacchetto parta da un collegamento con MTU ampia e debba attraversarne uno con MTU più piccola (es. Ethernet: 1500 byte). In questi casi, il router **non può inviare tutto il datagramma insieme** e lo **spezza in frammenti più piccoli**.

Ogni frammento diventa un datagramma IP a sé stante, con la sua intestazione.

Riassemblaggio

- La **frammentazione** avviene nei **router**, lungo il percorso verso la destinazione.
- Il **riassemblaggio** dei frammenti avviene **solo nell'host di destinazione**. I router intermedi non riassemblano mai.

Campi usati nell'intestazione IP

Nella frammentazione vengono usati tre campi chiave dell'intestazione IP:

1. **Identification**: stesso valore per tutti i frammenti del datagramma originale, per identificarli come appartenenti allo stesso gruppo.
2. **Flags**:
 - **MF** (More Fragments): se è **1**, indica che ci sono altri frammenti dopo questo.
 - **DF** (Don't Fragment): se è **1**, vieta la frammentazione. In quel caso, il pacchetto viene **scartato** se troppo grande.

3. **Fragment Offset:** specifica **dove si colloca** il frammento nel datagramma originale. È espresso in **unità da 8 byte**.

Esempio pratico

Ipotizzando un datagramma IP da **4000 byte**, che deve passare su un link con MTU di **1500 byte**.

- L'header IP è lungo 20 byte, quindi restano 1480 byte disponibili per i dati.
- Il datagramma viene diviso in 3 frammenti:
 - **Frammento 1:** offset 0, lunghezza dati 1480, MF=1
 - **Frammento 2:** offset 185 (1480/8), lunghezza 1480, MF=1
 - **Frammento 3:** offset 370 (2960/8), lunghezza 1020, MF=0

Tutti e tre portano lo **stesso Identification** e verranno ricomposti nell'host destinatario una volta ricevuti tutti.

3.3.4 Indirizzamento

Nel contesto del livello di rete, l'indirizzamento serve a identificare in modo univoco le interfacce dei dispositivi connessi a Internet o a una rete locale. Gli indirizzi IPv4, composti da 32 bit, sono utilizzati per assegnare a ciascuna interfaccia un identificatore logico, fondamentale per l'instradamento dei datagrammi IP.

Struttura dell'indirizzo IPv4

Un indirizzo IPv4 viene suddiviso logicamente in due parti:

- **Net ID (o parte di rete):** identifica la rete a cui l'host appartiene;
- **Host ID (o parte di host):** identifica l'interfaccia specifica all'interno di quella rete.

La distinzione tra Net ID e Host ID **non è fissa**, ma determinata dalla **subnet mask**, spesso indicata in **notazione CIDR** (es. /24, che indica che i primi 24 bit appartengono alla parte di rete).

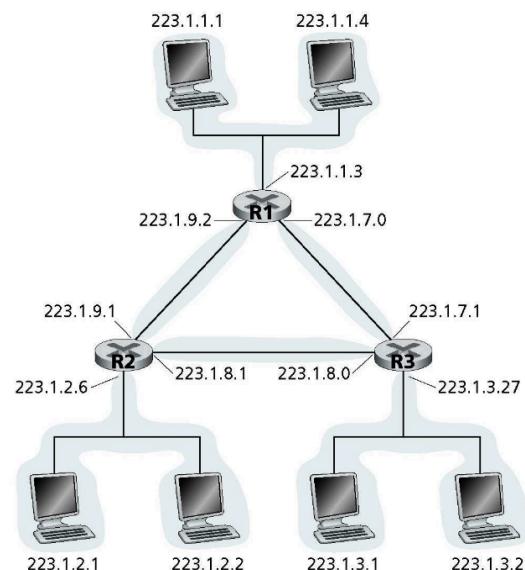


Comunicazione tra host nella stessa rete

Quando due dispositivi condividono lo stesso Net ID, possono comunicare direttamente tra loro tramite **livello data-link** (es. Ethernet), senza necessità di passare attraverso un router.

Comunicazione tra reti diverse

Se invece due dispositivi appartengono a **sottoreti differenti**, la comunicazione richiede l'intervento di un **router**. In questo caso, anche se i dispositivi sono vicini fisicamente, la loro appartenenza a reti diverse obbliga il pacchetto IP a passare attraverso un nodo intermedio che svolge il forwarding tra sottoreti.



Logica dell'indirizzamento

L'idea centrale dell'indirizzamento IP è **non identificare un host, ma un'interfaccia di rete**. Questo è importante perché un dispositivo può avere più interfacce (es. una Wi-Fi e una Ethernet), ognuna con un proprio indirizzo IP.

Inoltre, l'indirizzamento è strettamente legato al **meccanismo di routing**: l'instradamento dei pacchetti avviene sulla base della parte di rete dell'indirizzo, mentre la consegna all'interno della rete avviene sulla base della parte host.

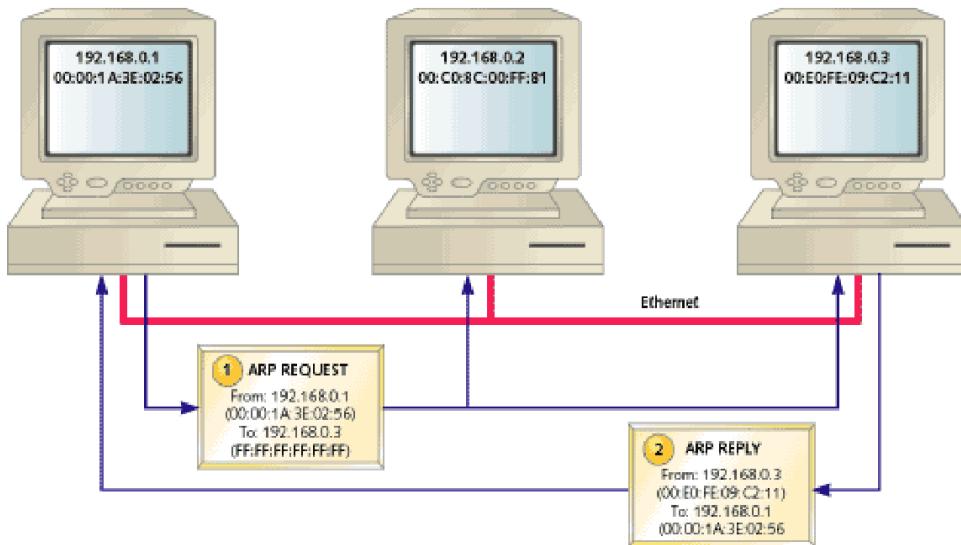
3.3.5 Il protocollo ARP

Nel modello di rete, quando un host vuole inviare un datagramma IP verso un altro host sulla **stessa LAN**, deve incapsularlo all'interno di un frame di livello link (es. Ethernet). Tuttavia, l'indirizzo IP **non basta** per trasmettere fisicamente il pacchetto: serve conoscere il **MAC address** (indirizzo hardware) dell'host di destinazione.

Qui entra in gioco il **protocollo ARP (Address Resolution Protocol)**, che ha il compito di **mappare un indirizzo IP su un MAC address** all'interno della stessa rete locale.

Funzionamento di ARP

Supponiamo che un host A voglia inviare un pacchetto IP a un host B con indirizzo IP noto (es. 192.168.1.15), ma **non conosce il suo indirizzo MAC**.



I passaggi sono:

1. ARP Request

L'host A invia un **messaggio broadcast** ARP sulla rete locale (MAC destination: **FF:FF:FF:FF:FF:FF**) chiedendo:

"Chi ha l'indirizzo IP 192.168.1.15? Rispondi con il tuo MAC."

2. ARP Reply

Se un host B sulla rete ha quell'indirizzo IP, risponde con un **messaggio unicast** ad A, fornendo il proprio indirizzo MAC:

"Io sono 192.168.1.15. Il mio MAC è 00:1C:42:2B:60:4A."

3. Caching

L'host A salva l'associazione tra IP e MAC in una **tavella ARP (ARP cache)** per evitare di rifare la richiesta in futuro.

| Interface | IP Address | Physical Address | Type | Expiry |
|-----------|--------------|-------------------|---------|--------|
| eth0 | 192.168.1.1 | 00:21:29:CC:A7:31 | Static | - |
| eth0 | 192.168.1.24 | 00:17:31:BA:5C:A4 | Dynamic | 2m 27s |
| eth1 | 10.1.2.66 | 00:1A:70:3C:A6:3D | Dynamic | 35s |
| wlan0 | 210.210.3.1 | 90:6C:AC:2D:DE:9A | Dynamic | 1m 14s |

Caratteristiche importanti

- ARP **non richiede una risposta da parte del livello 3 (IP)**, perché opera direttamente tra i livelli **network** e **data link**.
- Ogni nodo mantiene una **cache ARP**, con timeout per le voci.
- È un protocollo **stateless**: non mantiene lo storico delle richieste effettuate, ma solo l'ultima associazione.

- Può essere sfruttato per attacchi se non viene autenticato.

3.3.6 Altri protocolli del livello di rete

Il funzionamento del livello di rete non si limita all'indirizzamento e all'instradamento dei datagrammi IP. Esistono diversi protocolli ausiliari che svolgono **funzioni fondamentali per la gestione, il controllo e la configurazione della rete**, specialmente in ambienti dinamici o complessi. Tra questi, alcuni sono ormai storici ma didatticamente rilevanti, mentre altri sono tuttora in uso quotidiano nei sistemi moderni.

RARP (Reverse Address Resolution Protocol)

Il **RARP** è un protocollo ideato per permettere a un host di scoprire il **proprio indirizzo IP** conoscendo soltanto il proprio **indirizzo MAC**. Era utile in scenari con **dispositivi diskless**, i quali, appena avviati, non conoscevano il proprio IP. Tuttavia, RARP soffriva di limitazioni:

- richiedeva un **server RARP nella stessa subnet**,
- non era in grado di fornire altre informazioni fondamentali per il boot o la configurazione di rete.

Proprio per queste limitazioni, è stato sostituito da protocolli più versatili come **BOOTP** e, successivamente, **DHCP**.

BOOTP (Bootstrap Protocol)

Il **BOOTP** nasce come evoluzione di RARP, permettendo a un host non solo di ottenere il proprio **indirizzo IP**, ma anche:

- l'indirizzo del **router predefinito** (default gateway),
- il **server TFTP** per il boot di immagini di sistema,
- uno o più indirizzi di **server DNS**.

BOOTP utilizza UDP ed è progettato per facilitare l'avvio remoto di client (ad esempio terminali o stampanti di rete). Nonostante la maggiore potenza rispetto a RARP, BOOTP è comunque rigido: gli indirizzi IP assegnati sono statici e associati a MAC address specifici.

ICMP (Internet Control Message Protocol)

L'**ICMP** non è un protocollo di trasporto, ma agisce a fianco di IP per gestire **segnalazioni di errore e diagnostica**. È usato per comunicare informazioni tra router e host, per esempio:

- errore di destinazione irraggiungibile,
- TTL scaduto (es. nei traceroute),
- messaggi di echo (ping).

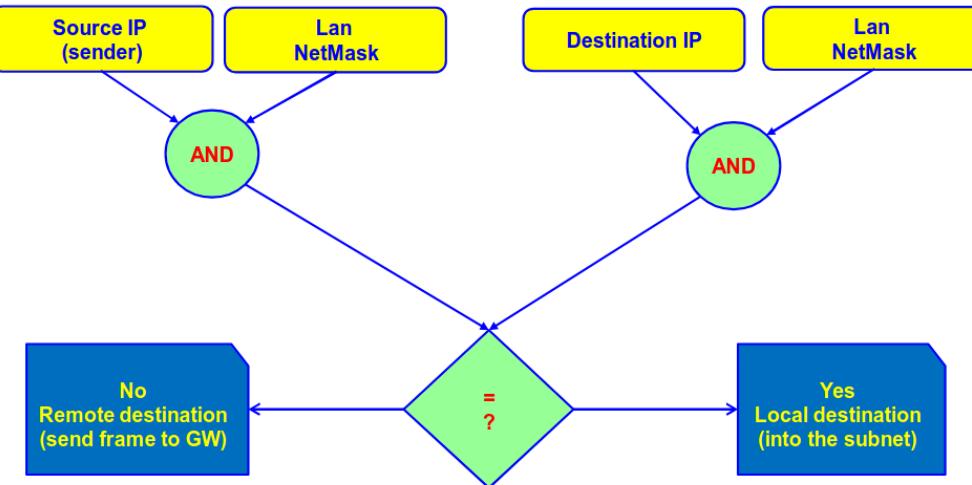
ICMP utilizza direttamente IP come mezzo di trasporto e i suoi messaggi sono fondamentali per la **gestione e il debugging della rete**. È importante notare che ICMP **non è affidabile e non garantisce consegna**, ma è spesso sufficiente per il controllo e il feedback di rete.

3.3.7 Maschera di rete

Nel contesto dell'indirizzamento IPv4, la **maschera di rete** (subnet mask) è un elemento fondamentale per determinare a quale rete appartiene un host. Essa consente di separare un indirizzo IP nelle sue due componenti principali:

- **Net ID**: identifica la rete;
- **Host ID**: identifica un host specifico all'interno di quella rete.

Per capire a quale rete appartiene un certo indirizzo IP, il dispositivo effettua un'**operazione logica AND** tra l'indirizzo IP e la subnet mask. Questo permette di ottenere il **Network Address**, cioè l'indirizzo della rete di riferimento.



Notazione

È un numero a 32 bit che indica **quanti bit dell'indirizzo IP fanno parte della rete**. Viene spesso rappresentata in due modi:

- **Forma decimale puntata:** es. 255.255.255.0
- **Notazione CIDR (Classless Inter-Domain Routing):** es. /24

Entrambe rappresentano la stessa informazione: i **primi 24 bit** dell'indirizzo sono riservati al **Net ID**, e i rimanenti 8 bit sono destinati agli host.

Esempio

Ipotizzando:

- **IP:** 192.168.1.10
- **Subnet Mask:** 255.255.255.0 (/24)

Si fa l'AND tra le due maschere:

| | |
|---------------|---|
| 192.168.1.10 | → 11000000.10101000.00000001.00001010 |
| 255.255.255.0 | → 11111111.11111111.11111111.00000000 |
| <hr/> | |
| Net ID | → 11000000.10101000.00000001.00000000 → 192.168.1.0 |

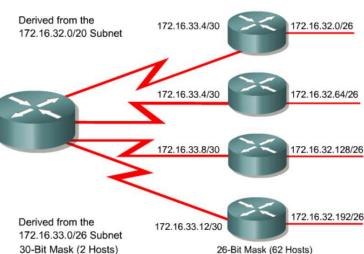
VLSM: Variable Length Subnet Mask

Il VLSM è una tecnica di subnetting che permette di utilizzare, **all'interno dello stesso blocco di indirizzi IP, maschere di rete di lunghezza differente** in base alle necessità di ciascuna sottorete.

Per utilizzarlo:

1. Si parte da un blocco principale.
2. Si suddivide in sottoreti e a ogni sottorete si assegna la maschera con il numero minimo di indirizzi necessari.

A Working VLSM Example



Vantaggi

1. Efficienza nell'uso degli indirizzi

Assegna a ogni sottorete la maschera più "stretta" possibile, minimizzando gli indirizzi inutilizzati e ritardando l'esaurimento dello spazio IP.

2. Flessibilità di progettazione

Permette di creare reti di dimensioni diverse (da poche decine fino a migliaia di host) all'interno dello stesso blocco gerarchico, senza dover riallocare o conservare grandi spazi vuoti.

3. Possibilità di aggregazione selettiva (supernetting)

Pur avendo sottoreti di lunghezze diverse, è spesso possibile riassumere percorsi contigui in un solo prefisso ("route summarization"), riducendo la dimensione delle tabelle di routing.

Svantaggi

1. Maggiore complessità di pianificazione

La progettazione richiede un'analisi attenta delle esigenze di ogni rete: errori di calcolo possono portare a collisioni di indirizzi o a spazi non più aggregabili.

2. Configurazione e manutenzione più laboriose

Gli indirizzi non uniformi (maschere variabili) aumentano il rischio di errori manuali in access list, route map e filtri, e complicano l'automazione dei template di configurazione.

3. Rischio di frammentazione del blocco

Se nel tempo si aggiungono nuove sottoreti non pianificate o con requisiti disparati, lo spazio IP rimasto può diventare "spezzettato" e più difficile da riutilizzare o aggregare.

Esempio

Ipotizzando di voler distribuire 3 blocchi di indirizzi IP a 3 sottoreti:

- A : 80
- B : 30
- C : 10

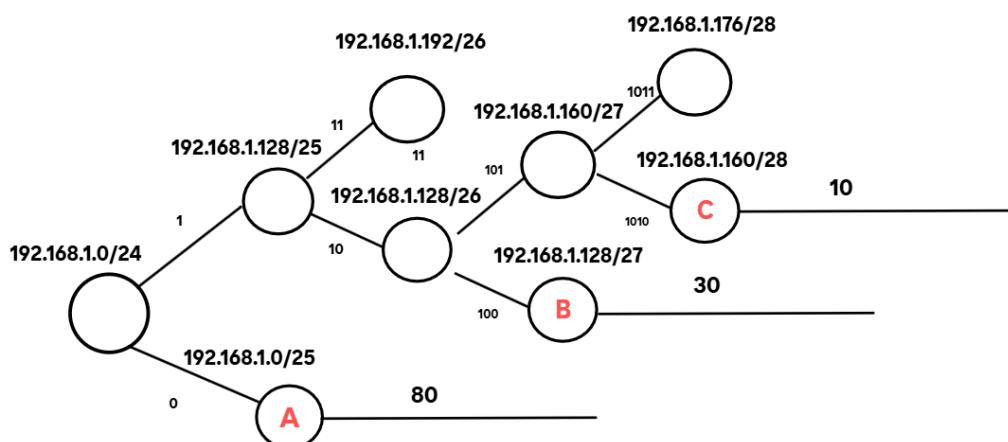
Sappiamo che il blocco di indirizzi da suddividere è **192.168.1.0/24**, e che le reti devono essere il più piccole possibili.

Per prima cosa assegniamo ad ogni sottorete il numero di indirizzi in base alla più vicina potenza di 2, ovvero:

- A : $80 \rightarrow 2^7 = 128$
- B : $30 \rightarrow 2^5 = 32$
- C : $10 \rightarrow 2^4 = 16$

Bisogna sempre tenere in considerazione che il numero minimo di indirizzi disponibile deve tener conto **dell'indirizzo di broadcast e dell'indirizzo di rete**.

Dopo aver scelto la lunghezza del blocco di indirizzi da assegnare, dividiamo gli indirizzi in un albero binario, assegnando i vari bit dal 25-esimo bit della maschera al 28-esimo.



3.3.8 Tabelle di host e tabelle di routing

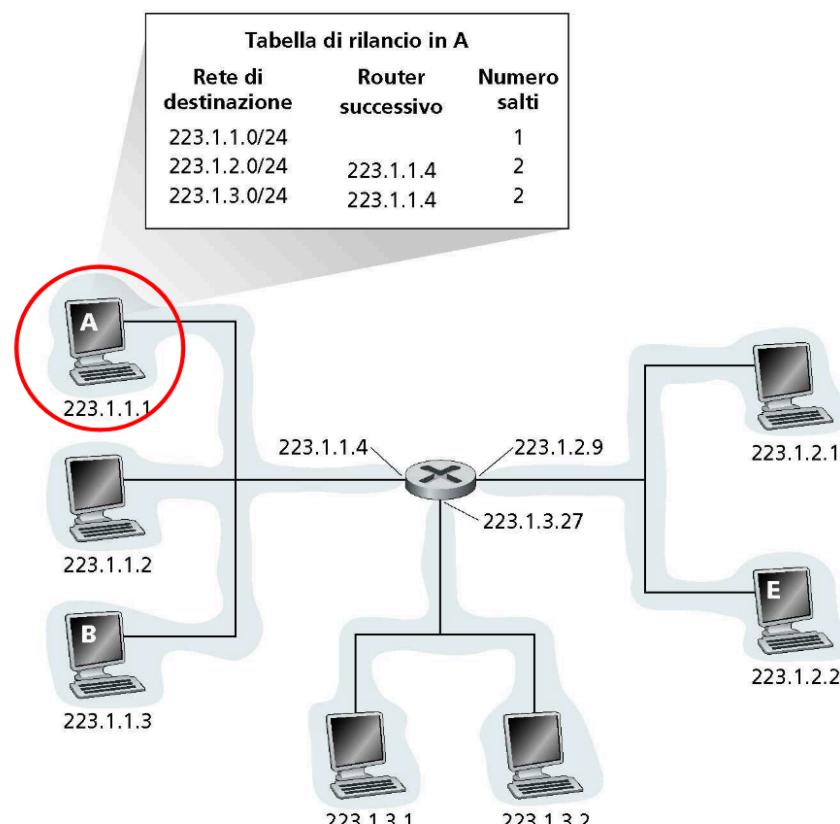
Quando un host genera un datagramma IP, il suo stack di rete deve decidere **a quale interfaccia inviarlo**. Questo processo non avviene "alla cieca", ma si basa su due elementi fondamentali: la **tabella dell'host** (host table) e la **tabella di routing** (routing table).

Queste tabelle rappresentano la "mappa di consegna" che ogni sistema utilizza per sapere **come inoltrare i pacchetti**, sia all'interno della propria LAN, sia verso reti esterne.

Tabella dell'host (Host Table)

La **host table** è usata da un **endpoint (PC, server, ecc.)** per decidere come gestire un pacchetto in uscita. Si tratta di una tabella generalmente semplice, che contiene:

- Le **interfacce disponibili** sul dispositivo (es. `wlan0`, `eth0`)
- Il **gateway predefinito**
- Le **route locali** (per la propria subnet)
- Eventuali **regole statiche** definite manualmente



Esempio

| Rete di destinazione | Gateway | Interfaccia |
|----------------------|-------------|-------------|
| 192.168.1.0/24 | * | wlan0 |
| default | 192.168.1.1 | wlan0 |

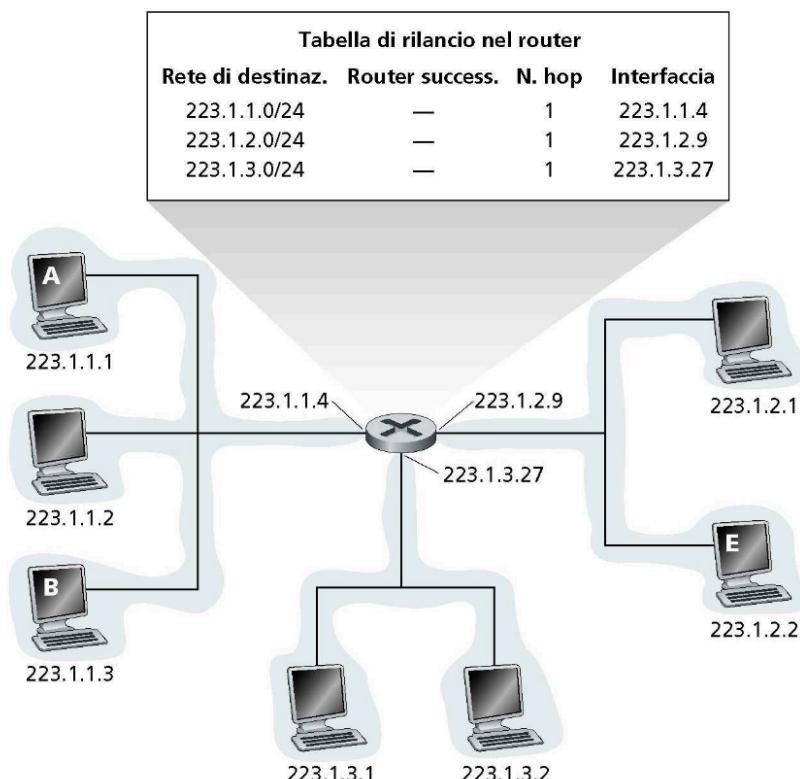
Questo significa che tutti i pacchetti diretti alla rete `192.168.1.0/24` vanno inviati direttamente via `wlan0`, tutto il resto va al router `192.168.1.1`.

Tabella di routing (Routing Table)

Nei **router**, la tabella di routing è molto più complessa. Serve a **decidere a quale interfaccia inoltrare un datagramma**, a seconda della rete di destinazione. A differenza dell'host, il router gestisce **più interfacce e collega reti diverse**.

Ogni voce nella tabella contiene:

- Una **rete di destinazione** (es. `10.0.0.0/8`)
- Una **next hop** (il router verso cui inoltrare il pacchetto)
- L'**interfaccia di uscita** (es. `eth1`)
- Una **metrica**, cioè un costo relativo (hop count, delay, larghezza di banda, ecc.)



Regole di routing: longest prefix matching

Il router sceglie la rotta che **corrisponde al prefisso più lungo** dell'indirizzo di destinazione. Ad esempio:

```
10.11.0.0/16 → eth0  
10.11.12.0/24 → eth1
```

Un pacchetto per `10.11.12.7` corrisponde a **entrambe le rotte**, ma `/24` è il match più lungo → quindi viene inoltrato su `eth1`.

3.3.9 DHCPv4

Il **Dynamic Host Configuration Protocol versione 4 (DHCPv4)** è il protocollo di riferimento per l'assegnazione dinamica delle configurazioni IP in reti IPv4. È stato progettato per automatizzare il processo di configurazione degli host su una rete, evitando la necessità di assegnare manualmente indirizzi IP, subnet mask e gateway a ogni dispositivo.

DHCPv4 rappresenta un'evoluzione di protocolli precedenti come **RARP** e **BOOTP**, superandone i limiti grazie alla **gestione dinamica degli indirizzi IP**, la possibilità di **rilasciare automaticamente le configurazioni di rete**, e l'inclusione di **molti parametri** aggiuntivi come gateway, server DNS, tempo di lease e altro ancora.

Il ciclo DHCP

Il processo di assegnazione IP tramite DHCPv4 avviene attraverso **quattro messaggi principali**, tutti trasmessi inizialmente in **broadcast**:

1. DHCP Discover

Il client, che non ha ancora un indirizzo IP, invia un messaggio broadcast ($0.0.0 \rightarrow 255.255.255.255$) per cercare un server DHCP disponibile.

2. DHCP Offer

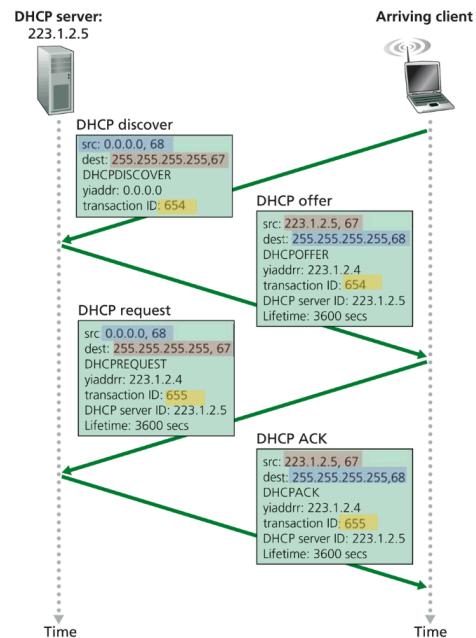
Uno o più server DHCP rispondono proponendo un indirizzo IP e altre opzioni (subnet mask, DNS, ecc.).

3. DHCP Request

Il client seleziona una delle offerte ricevute e invia una richiesta formale per ottenere quell'indirizzo.

4. DHCP ACK

Il server conferma l'assegnazione e il client può ora utilizzare l'indirizzo IP per un **tempo determinato** (lease).



Durante il lease, il client può **rinnovare** l'indirizzo inviando un nuovo DHCP Request. Alla scadenza, l'indirizzo torna disponibile nel pool del server.

Parametri forniti da DHCP

Un server DHCP può assegnare molteplici configurazioni in un colpo solo, tra cui:

- Indirizzo IP
- Subnet mask
- Gateway predefinito
- Server DNS
- Tempo di lease
- Server WINS, dominio di ricerca, e altre opzioni opzionali

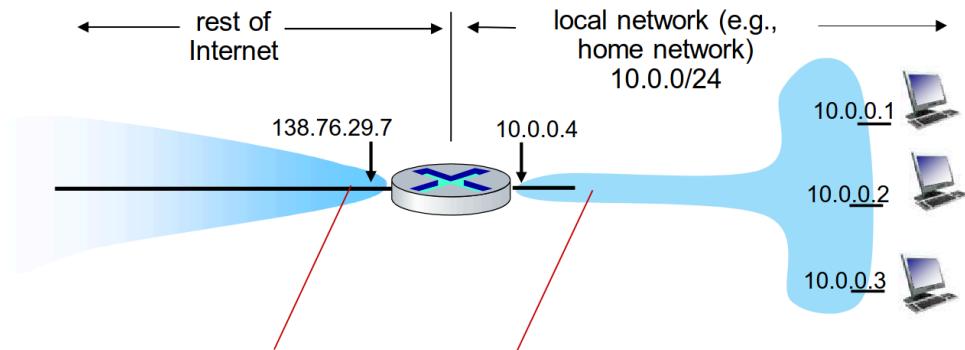
Questo rende il protocollo fondamentale per ambienti dinamici, come le reti aziendali e domestiche, dove dispositivi mobili (laptop, smartphone) si connettono e disconnettono frequentemente.

3.3.10 NAT

Il **Network Address Translation (NAT)** è una tecnica fondamentale nel moderno networking IPv4, progettata per affrontare la carenza di indirizzi IP pubblici. Il NAT consente a una rete privata di accedere a Internet utilizzando un **singolo indirizzo IP pubblico**, traducendo dinamicamente gli indirizzi IP e le porte delle connessioni in uscita e in ingresso.

Il router NAT agisce come intermediario: riceve pacchetti da host interni con IP privati (es. $192.168.x.x$) e li inoltra su Internet utilizzando il proprio **indirizzo IP pubblico**. Quando le risposte tornano, il router le inoltra

correttamente all'host interno, sfruttando una **tavella di associazione temporanea** tra IP/porta interna e IP/porta esterna.



Funzionamento base del NAT

Supponiamo che un host locale 192.168.1.10 apra una connessione TCP verso un server web su Internet:

- Porta sorgente locale: 49152
- IP e porta pubblica del NAT: 93.40.22.158:60001

Il router NAT sostituisce il mittente nel datagramma IP:

192.168.1.10:49152 → 93.40.22.158:60001

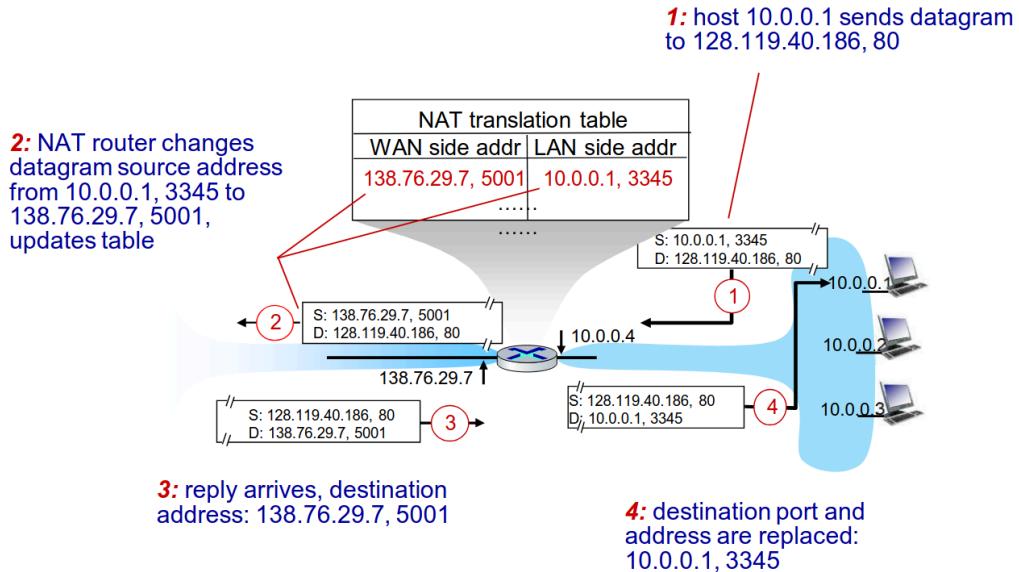
La risposta dal server Internet sarà indirizzata a 93.40.22.158:60001. Il router NAT consulterà la sua **NAT table** per reinstradare il pacchetto a:

192.168.1.10:49152

NAT Table

Il NAT mantiene una tabella temporanea con tutte le connessioni attive:

| IP Privato | Porta Privata | IP Pubblico | Porta Pubblica |
|--------------|---------------|--------------|----------------|
| 192.168.1.10 | 49152 | 93.40.22.158 | 60001 |
| 192.168.1.11 | 49153 | 93.40.22.158 | 60002 |



3.4 IPv6

3.4.1 Limitazioni di IPv4

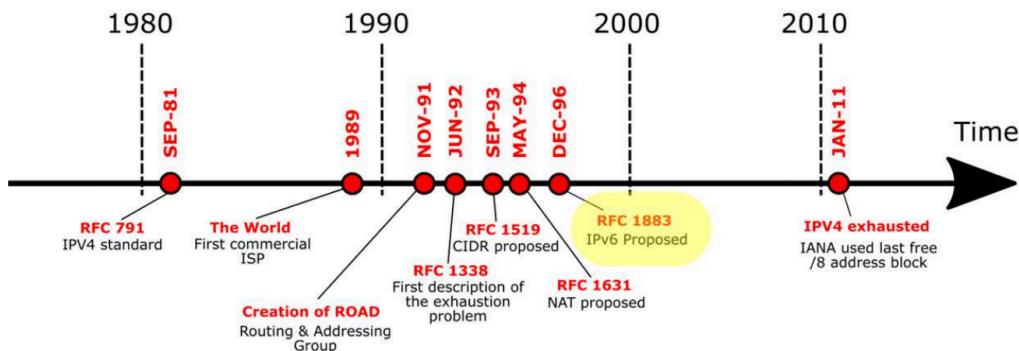
Il protocollo **IPv4**, definito negli anni '80, è stato il cuore dell'Internet moderno per decenni. Tuttavia, col passare del tempo e l'espansione globale della rete, sono emersi limiti strutturali sempre più evidenti, che hanno spinto verso l'adozione di una nuova versione del protocollo: **IPv6**.

Esaurimento degli indirizzi

Uno dei problemi più critici di IPv4 è lo **spazio limitato degli indirizzi**: con i suoi **32 bit**, il protocollo può rappresentare **circa 4,3 miliardi di indirizzi unici**. Questo numero, sebbene sembri elevato, è diventato insufficiente con:

- la crescita esponenziale dei dispositivi connessi (smartphone, IoT),
- l'espansione dei servizi Internet nei paesi in via di sviluppo,
- l'allocazione inefficiente degli indirizzi storici (blocchi A, B, C).

Questo limite ha spinto all'uso intensivo di **NAT**, una soluzione temporanea che ha prolungato la vita di IPv4 ma ne ha anche compromesso la trasparenza.



Rigidità nell'header

L'header IPv4 è **complesso e poco flessibile**, con campi opzionali raramente usati ma presenti. Questo lo rende **poco scalabile** per le esigenze delle reti moderne, che richiedono:

- estensibilità del protocollo,
- gestione efficiente della qualità del servizio (QoS),
- supporto per nuove funzionalità di mobilità e sicurezza.

IPv4, nella sua struttura base, **non supporta crittografia né autenticazione**.

Mancanza di supporto integrato per mobilità e autoconfigurazione

IPv4 **non supporta nativamente la mobilità** degli host (mobile IP) né un meccanismo di **autoconfigurazione plug-and-play**. Per assegnare indirizzi IP dinamici è necessario ricorrere a protocolli esterni come DHCP.

In contesti con dispositivi mobili o reti dinamiche, questa mancanza si traduce in maggiore complessità nella gestione.

Sicurezza non incorporata

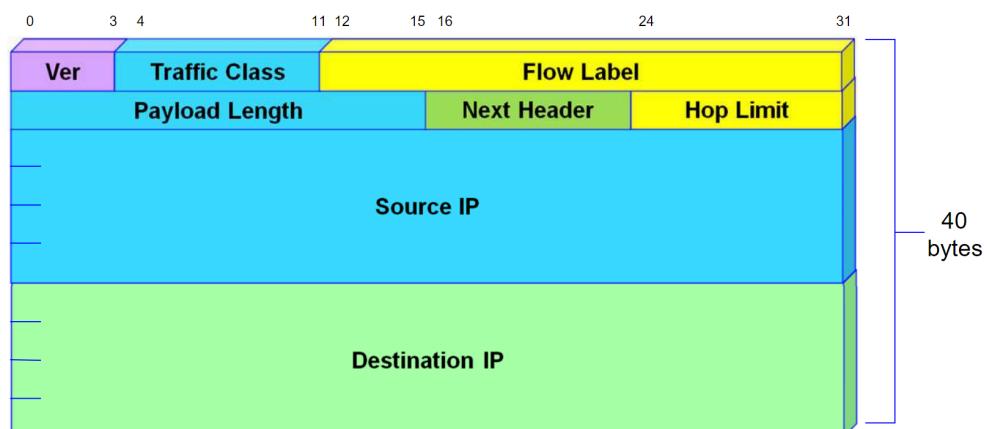
La sicurezza nel protocollo IPv4 **non è parte integrante dello standard**. Non sono previsti meccanismi nativi per:

- autenticare l'origine del pacchetto,
- garantire l'integrità del messaggio,
- proteggere la confidenzialità.

Questo ha reso necessario l'uso di soluzioni parallele come **IPsec**, implementate su base opzionale e non sempre interoperabili.

3.4.2 Datagramma

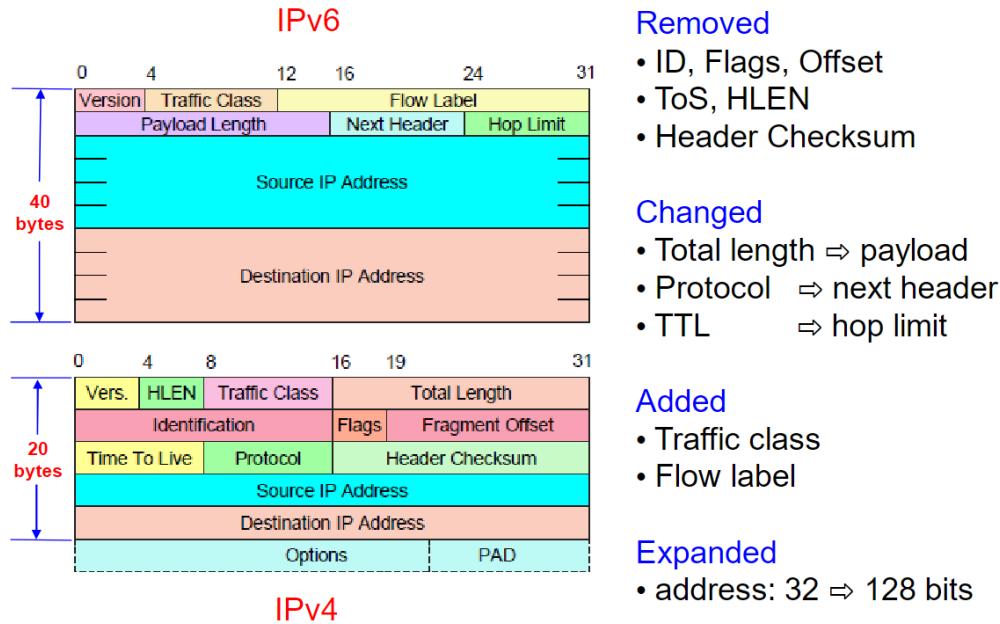
Uno degli obiettivi principali del protocollo IPv6 è stato **semplificare il formato dell'header IP**, rendendolo più efficiente nel forwarding e più adatto alle esigenze delle reti moderne. Il **datagramma IPv6** è quindi strutturalmente diverso da quello IPv4, pur mantenendo le funzionalità di base.



Confronto tra header IPv4 e IPv6

| Caratteristica | IPv4 | IPv6 |
|-------------------|---------------------------------|---------------------------------------|
| Dimensione header | Variabile (20–60 byte) | Fissa (40 byte) |
| Indirizzi IP | 32 bit | 128 bit |
| Checksum | Presente (verifica dell'header) | Rimosso (affidato a strati superiori) |

| Caratteristica | IPv4 | IPv6 |
|------------------------|------------------------------------|--|
| Fragmentation | Campo ID, Offset, MF | Gestita solo dal sender via header esteso |
| Options | Campo opzionale (lungo, complesso) | Header di estensione dedicati |
| TTL | Time To Live | Hop Limit |
| Protocol | Protocol (next-level) | Next Header |
| Type of Service | ToS/DSCP | Traffic Class + Flow Label |
| Header Length | Campo dedicato | Eliminato (lunghezza fissa) |



Miglioramenti introdotti da IPv6

- Header semplificato:** i router possono analizzare e inoltrare i pacchetti più velocemente grazie a un header a dimensione fissa e privo di checksum.
- Estensioni modulari:** funzionalità avanzate (es. IPsec, frammentazione) non appesantiscono l'header base, ma vengono aggiunte tramite *extension headers* solo quando servono.
- Assenza di frammentazione intermedia:** solo il mittente può frammentare un pacchetto, riducendo il carico computazionale sui router.
- Indirizzamento esteso:** i 128 bit permettono una gerarchia più efficiente, un indirizzamento globale, e l'unicità degli host anche senza NAT.

3.4.3 Indirizzamento

Uno dei principali cambiamenti introdotti con IPv6 è il nuovo schema di **indirizzamento a 128 bit**, pensato per superare definitivamente la scarsità di indirizzi che affligge IPv4. Con uno spazio teorico pari a 3.4×10^{38} indirizzi, IPv6 consente l'identificazione unica di ogni dispositivo connesso alla rete, senza bisogno di NAT o di assegnazioni conservative.

Rappresentazione degli indirizzi IPv6

Un indirizzo IPv6 è composto da **8 gruppi di 4 cifre esadecimale**, separati da due punti (:), ad esempio:

2001:0db8:0000:0042:0000:8a2e:0370:7334

Per semplicità, si applicano le seguenti regole:

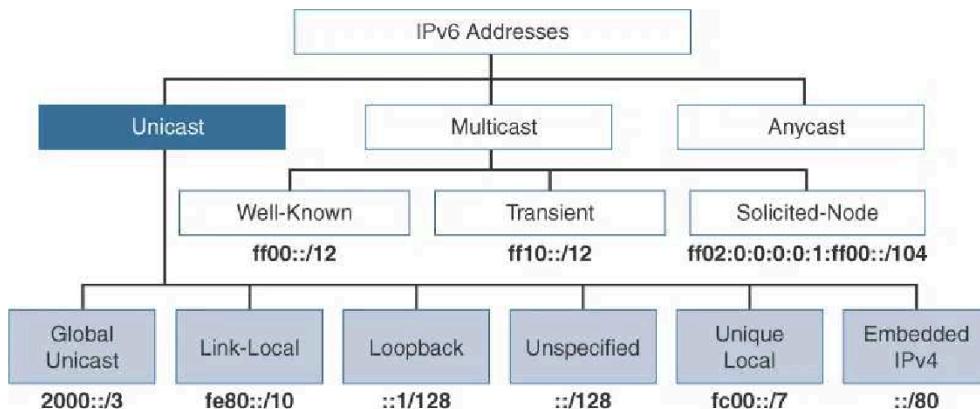
- Gli **zeri iniziali** in ogni blocco possono essere omessi.
- Una sequenza continua di blocchi **0000** può essere abbreviata con **::** (ma solo una volta per indirizzo).

2001:db8::42:0:8a2e:370:7334

Classi di indirizzi IPv6

In IPv6 **non esistono classi come in IPv4** (A, B, C), ma l'indirizzo è suddiviso in:

- **Prefisso di rete** (Net ID), indicato da **/n** (es. **/64**),
- **Interface Identifier (Host ID)**, ovvero l'identificatore unico dell'host.



I principali tipi di indirizzo sono:

Unicast

Indirizzo che identifica **un solo host**. I pacchetti destinati a un unicast arrivano a **un'interfaccia specifica**.

- Esempio: **2001:db8::1**
- Il **Global Unicast Address** sostituisce gli indirizzi pubblici IPv4.

Multicast

Un singolo pacchetto viene consegnato a **più destinatari**, appartenenti a un gruppo.

- Inizia con **ffxx::...**
- Sostituisce il broadcast (non esiste in IPv6).

Anycast

Indirizzo assegnato a **più dispositivi**, ma il pacchetto viene consegnato **solo al nodo più vicino (in termini di routing)**.

- Utilizzato per bilanciamento del carico e servizi distribuiti (es. DNS root servers).

Indirizzi locali

Link-local

- Usati per comunicazioni locali (nella stessa rete fisica).
- Generati automaticamente su ogni interfaccia.
- Prefisso: **fe80::/10**

Unique Local Addresses (ULA)

- Equivalenti ai privati IPv4 (come **192.168.x.x**)
- Non sono instradati su Internet.
- Prefisso: **fc00::/7**

- Necessari per protocolli come **NDP** o **routing locale**.

Autoconfigurazione

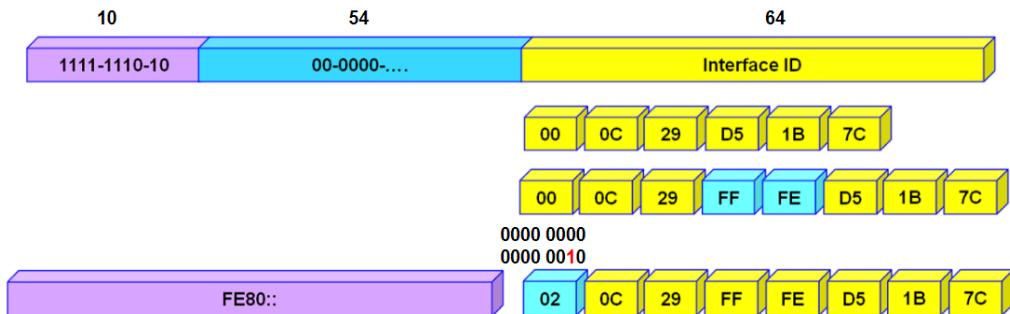
IPv6 prevede **meccanismi automatici di configurazione**, sia stateless (senza DHCP) sia stateful (con DHCPv6). Con lo stateless address autoconfiguration (SLAAC), l'host costruisce il proprio indirizzo combinando:

- il **prefisso ricevuto dal router** (es. /64)
- con l'**identificativo dell'interfaccia** (es. MAC address elaborato in formato EUI-64)

Questo elimina la necessità di server DHCP per ambienti semplici e dinamici.

Standard EUI-64

Nello standard IPv6, l'**identificatore dell'interfaccia** (Interface Identifier) può essere generato automaticamente partendo dall'**indirizzo MAC** dell'interfaccia di rete, usando lo **schema EUI-64**. Questo processo è parte del meccanismo di **Stateless Address Autoconfiguration (SLAAC)**.



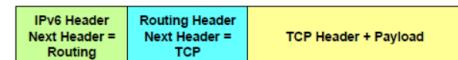
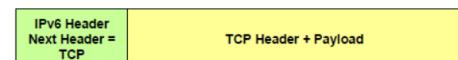
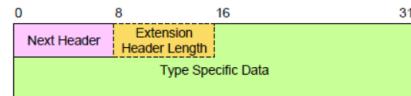
```
[root@LabRete ~]# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0C:29:D5:1B:7C
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fed5:1b7c/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:11 errors:0 dropped:0 overruns:0 frame:0
             TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:1421 (1.3 KiB)  TX bytes:1382 (1.3 KiB)
             Interrupt:19 Base address:0x2000
```

Funzionamento

- Si prende il **MAC address a 48 bit** dell'interfaccia (es. 00:1A:2B:3C:4D:5E).
- Lo si divide in due metà e si inserisce **FF:FE** nel mezzo → 00:1A:2B:**FF:FE**:3C:4D:5E .
- Si inverte il **settimo bit** (Universal/Local bit) del primo byte per distinguere l'indirizzo come generato localmente.

3.4.4 Header opzionali

Una delle innovazioni fondamentali introdotte da IPv6 riguarda la gestione delle **funzionalità opzionali** attraverso un sistema di **header di estensione (extension headers)**. A differenza di IPv4, dove l'header base poteva contenere campi opzionali con una struttura poco chiara e variabile, in IPv6 questi elementi sono spostati in **blocchi separati**, concatenati in una catena di intestazioni ordinate.



Architettura modulare

Nel datagramma IPv6, l'**header principale è fisso e semplificato** (lunghezza costante di 40 byte), ma può essere seguito da una sequenza di **header opzionali**, specificati attraverso il campo **Next Header**. Questo campo agisce come un puntatore, indicando il tipo del successivo blocco di intestazione (proprio come accade con il campo **Protocol** in IPv4).

Gli header opzionali sono letti **solo se necessario**, rendendo più efficiente il forwarding dei pacchetti da parte dei router intermedi, che possono ignorare le informazioni non rilevanti.

Principali header di estensione

1. Hop-by-Hop Options Header

Contiene opzioni che devono essere analizzate da **ogni router** attraversato dal pacchetto. È l'unico header che, se presente, **dove trovarsi subito dopo l'header IPv6**.

2. Routing Header

Specifica un percorso alternativo che il pacchetto dovrebbe seguire, simile al source routing di IPv4 (raramente usato per motivi di sicurezza).

3. Fragment Header

Viene usato quando un pacchetto è frammentato dal **mittente** (nei router IPv6 non è permessa la frammentazione). Sostituisce i campi di frammentazione presenti in IPv4.

4. Destination Options Header

Contiene opzioni destinate solo al **nodo finale** (o a ogni nodo se usato insieme a un Routing Header).

5. Authentication Header (AH) e Encapsulating Security Payload (ESP)

Utilizzati da IPsec per **autenticare e/o cifrare** i pacchetti IPv6, integrando funzionalità di sicurezza direttamente nel livello di rete.

3.4.5 Frammentazione

Nel protocollo IPv4, la frammentazione poteva avvenire **in qualsiasi nodo della rete**, incluso ogni router intermedio. Questo comportamento, sebbene flessibile, imponeva un **carico computazionale elevato ai router**, che dovevano spezzare i pacchetti, aggiornare gli header e ricomporre le unità finali all'arrivo.

Con IPv6, questo approccio è stato **radicalmente ripensato**. La frammentazione è gestita **esclusivamente dall'host sorgente**, che deve preventivamente scoprire il valore minimo della **MTU (Maximum Transmission Unit)** lungo il percorso e frammentare il pacchetto in modo compatibile.

Niente frammentazione nei router

I router IPv6 non frammentano più i pacchetti. Se un pacchetto supera la MTU e non è frammentato in anticipo dal mittente, il router lo scarta e invia al mittente un messaggio **ICMPv6 "Packet Too Big"**. Questo comportamento fa parte della procedura di:

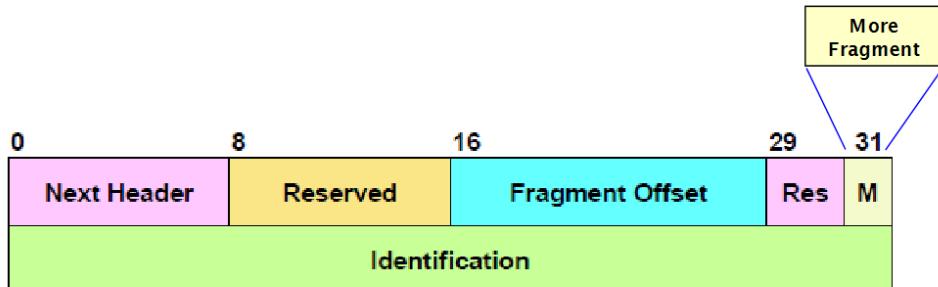
Path MTU Discovery

L'host scopre dinamicamente la MTU minima lungo il percorso verso la destinazione. Se invia un pacchetto troppo grande, riceve un errore ICMPv6 e si adatta inviando frammenti più piccoli.

Header di frammentazione

Se l'host decide di frammentare il pacchetto, aggiunge un **header di estensione specifico per la frammentazione**, che contiene:

- **Identification**: un numero univoco per riassemblare i frammenti.
- **Fragment Offset**: l'offset (in multipli di 8 byte) rispetto all'inizio del pacchetto originale.
- **M flag (More Fragments)**: indica se ci sono frammenti successivi.



Solo l'host di destinazione ha il compito di **ricomporre il pacchetto**.

Differenze rispetto a IPv4

| Caratteristica | IPv4 | IPv6 |
|--------------------------|-----------------------------|--------------------------------------|
| Chi frammenta? | Mittente e router | Solo il mittente |
| Header di frammentazione | Parte dell'header IPv4 | Header di estensione separato |
| Riassemblaggio | Solo al destinatario finale | Solo al destinatario finale |
| Notifica errore | ICMP Fragmentation needed | ICMPv6 Packet Too Big |

3.4.6 Sicurezza

Uno dei miglioramenti strutturali introdotti con IPv6 è il **supporto integrato alla sicurezza a livello di rete**, tramite l'inclusione nativa di **IPsec (Internet Protocol Security)**. A differenza di IPv4, dove l'uso di IPsec è opzionale e richiede configurazioni aggiuntive, IPv6 lo incorpora come **componente obbligatorio del protocollo base**, anche se nella pratica la sua attivazione dipende comunque dalle scelte implementative del sistema operativo o dell'amministratore di rete.

Cosa offre IPsec in IPv6?

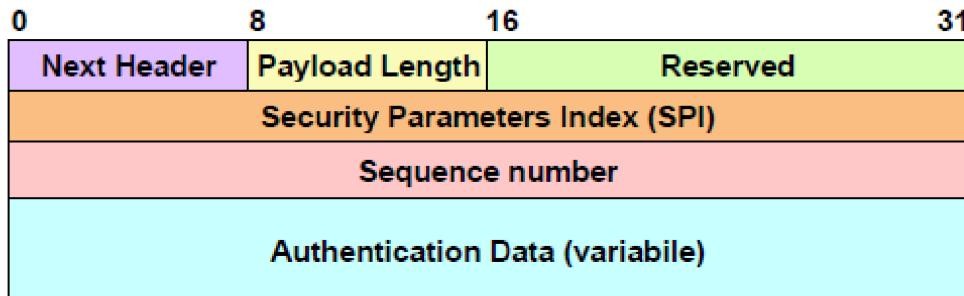
IPsec consente di realizzare tre obiettivi fondamentali di sicurezza:

1. **Autenticazione dell'origine del pacchetto** – verifica che il mittente sia chi afferma di essere;
2. **Integrità dei dati** – protegge il contenuto del pacchetto da modifiche accidentali o malevoli;

3. **Cifratura del contenuto** – impedisce a terzi non autorizzati di leggere i dati in transito.

Queste funzionalità sono ottenute mediante due tipi di header IPv6 opzionali (extension headers):

- **AH (Authentication Header)**: per l'autenticazione e l'integrità.
- **ESP (Encapsulating Security Payload)**: per la cifratura e, optionalmente, anche autenticazione.



Confronto con IPv4

Nel mondo IPv4, IPsec è disponibile ma **non fa parte del protocollo base** e non è garantito che tutti i dispositivi lo supportino. IPv6, al contrario, è stato progettato sin dall'inizio con la sicurezza in mente, anche se nella pratica **non tutti i dispositivi abilitano IPsec di default**, specialmente nelle reti consumer.

3.4.7 Altri protocolli

Il passaggio da IPv4 a IPv6 ha richiesto la ridefinizione di diversi protocolli ausiliari per supportare le nuove funzionalità e mantenere la compatibilità. Tra questi spiccano **ICMPv6**, **DHCPv6** e i **meccanismi di transizione** che permettono ai due protocolli di coesistere nella rete attuale.

ICMPv6

Il protocollo **ICMPv6** (Internet Control Message Protocol versione 6) svolge un ruolo cruciale nel funzionamento di IPv6, molto più ampio rispetto al suo predecessore in IPv4. Le sue funzioni includono:

- **Gestione degli errori**: pacchetti troppo grandi, indirizzi irraggiungibili, hop limit superato, ecc.
- **Diagnostica**: come `ping6`, che invia messaggi Echo Request/Reply.
- **Neighbor Discovery Protocol (NDP)**: utilizzato per
 - rilevare i router disponibili,
 - ottenere prefissi di rete,
 - determinare l'indirizzo MAC dei vicini (ARP non esiste più in IPv6),
 - gestire la duplicazione degli indirizzi (DAD – Duplicate Address Detection).

ICMPv6 è **indispensabile** in IPv6: un pacchetto IPv6 senza il corretto trattamento dei messaggi ICMPv6 può causare malfunzionamenti o perdita di connettività.

DHCPv6: configurazione dinamica per IPv6

Anche se IPv6 supporta l'**autoconfigurazione stateless (SLAAC)**, molte reti adottano anche o in alternativa il protocollo **DHCPv6** per gestire l'assegnazione centralizzata di parametri IP. DHCPv6 permette di:

- Assegnare indirizzi IPv6 dinamici,
- Distribuire informazioni aggiuntive (DNS, lease, gateway),
- Operare in modalità **stateful** (con server che memorizza lo stato del client) o **stateless** (senza assegnare IP, ma distribuendo altri parametri).

Nelle reti aziendali o ISP, DHCPv6 è spesso preferito per la **gestione centralizzata** e la possibilità di tenere traccia degli indirizzi assegnati.

Coesistenza IPv4/IPv6

Nonostante IPv6 sia stato standardizzato da anni, la transizione da IPv4 è ancora **in corso** e procede lentamente. Le cause principali sono:

- l'enorme infrastruttura esistente basata su IPv4,
- la necessità di garantire compatibilità,
- la difficoltà di aggiornamento nei dispositivi legacy.

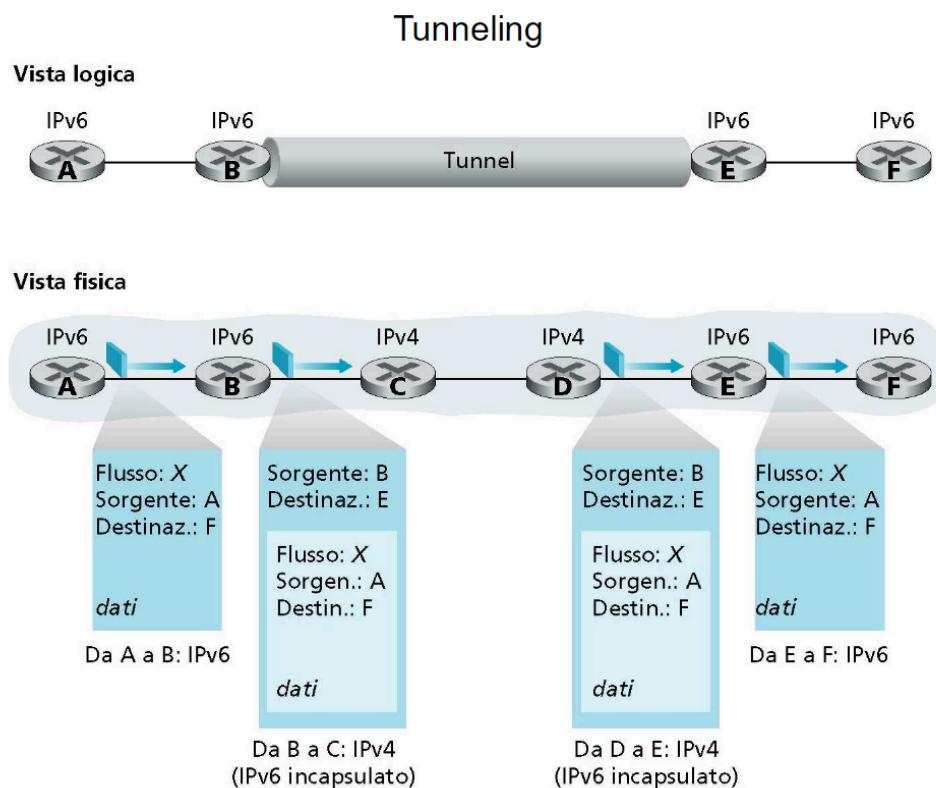
Attualmente molte reti adottano configurazioni **dual stack**, in cui i dispositivi sono in grado di operare **sia con IPv4 che con IPv6**. In alternativa, esistono soluzioni di **incapsulamento e traduzione**.

Incapsulamento IPv6 su IPv4

L'**incapsulamento** è una tecnica che consente di **trasportare pacchetti IPv6 all'interno di pacchetti IPv4**, utile quando due nodi IPv6 devono comunicare attraverso un'infrastruttura che non supporta ancora IPv6 nativamente.

Un esempio comune è **6to4 tunneling**:

- Il pacchetto IPv6 viene incapsulato come payload di un datagramma IPv4,
- Gli header IPv4 sono interpretati da router intermedi compatibili con il tunneling,
- All'arrivo, il pacchetto viene decapsulato e instradato normalmente su rete IPv6.



Questo approccio consente una **transizione graduale**, anche se introduce latenza e complicazioni di routing.

3.5 Cenni di firewall

I firewall sono una componente fondamentale della sicurezza informatica moderna. Nel contesto delle reti, un firewall è un sistema (software, hardware o misto) che **filtrà il traffico in ingresso e in uscita** da una rete, **applicando delle regole** per determinare quali pacchetti sono autorizzati a passare e quali devono essere bloccati.

L'obiettivo principale è **proteggere i dispositivi e i dati** da accessi non autorizzati, attacchi o attività indesiderate, mantenendo al contempo aperte le comunicazioni legittime.

3.5.1 Cos'è il firewall

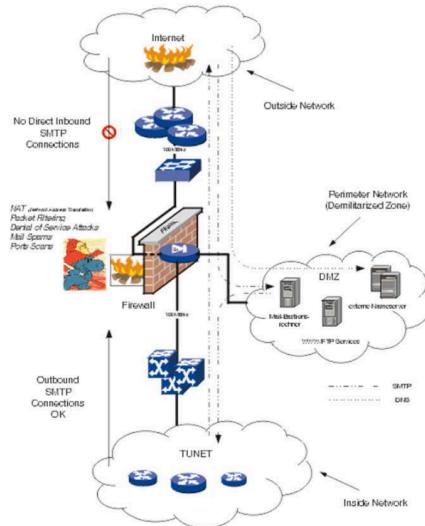
Un firewall è un **sistema di difesa perimetrale** progettato per monitorare e controllare il traffico di rete secondo criteri di sicurezza predefiniti. Agisce come una **barriera** tra una rete affidabile (es. una LAN aziendale o domestica) e una rete non affidabile (es. Internet).

Il funzionamento di base si basa su **regole**, che analizzano:

- l'indirizzo IP sorgente e destinazione,
- la porta sorgente e destinazione,
- il protocollo (TCP, UDP, ICMP...),
- lo stato della connessione (es. nuova, stabilita, relazionata).

In base a queste regole, il firewall può:

- **consentire (ACCEPT)** il pacchetto,
- **bloccarlo (DROP)** senza notifica,
- o **rifiutarlo (REJECT)** con un messaggio di errore.



Due direzioni di controllo

- **Ingress filtering**: controlla ciò che entra nella rete.
- **Egress filtering**: controlla ciò che esce dalla rete.

3.5.2 Firewall software

Un **firewall software** è un'applicazione che gira direttamente su un host (es. un computer o server) e ne controlla il traffico di rete.

Caratteristiche:

- Facile da installare e configurare.
- Consente regole personalizzate per ogni processo o programma.
- Ideale per ambienti **client o postazioni individuali**.

3.5.3 Firewall fisico

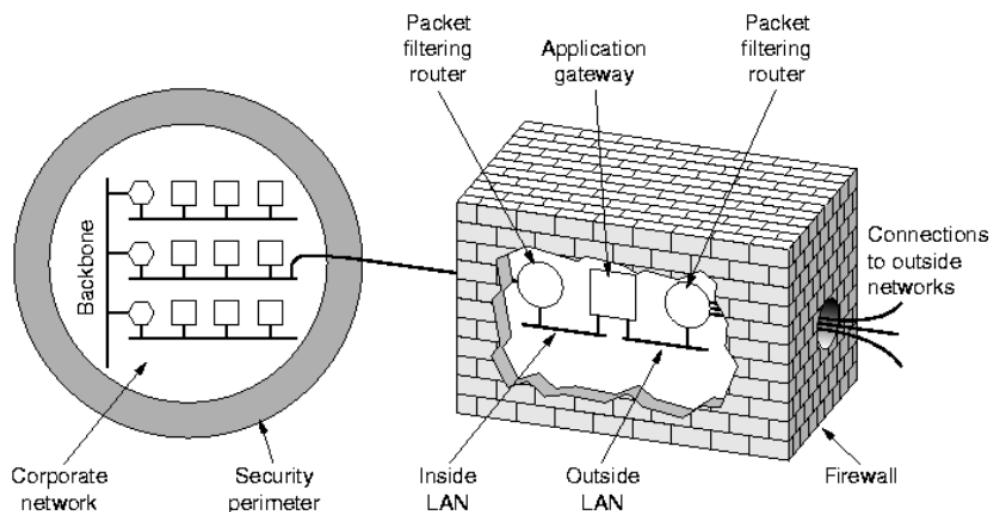
Un **firewall fisico** (o hardware) è un dispositivo dedicato (standalone o integrato in un router) progettato per filtrare il traffico **di un'intera rete**. È molto usato in ambienti aziendali e infrastrutture critiche.

Posizionamento

Viene posto **tra la rete interna e il gateway di accesso a Internet**, agendo come punto di passaggio obbligato per tutto il traffico.

Vantaggi:

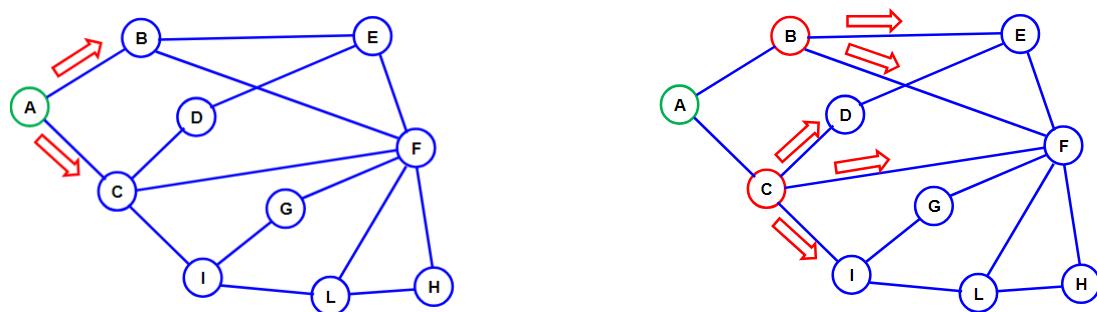
- **Protegge più dispositivi contemporaneamente.**
- Offre prestazioni elevate (grazie all'hardware dedicato).
- Spesso include funzioni aggiuntive: NAT, VPN, load balancing.



3.6 Algoritmi di Routing

3.6.1 Flooding

Il **flooding** è un algoritmo di routing molto semplice, dove ogni router inoltra una copia di ogni pacchetto ricevuto su **tutte le interfacce tranne quella di arrivo**. In questo modo, il pacchetto si diffonde rapidamente attraverso tutta la rete, raggiungendo tutti i nodi.



Funzionamento

Quando un router riceve un pacchetto:

1. Verifica da quale interfaccia è arrivato.
2. Inoltra una copia del pacchetto **su tutte le altre interfacce**.

3. I router successivi fanno lo stesso, generando una crescita esponenziale del numero di pacchetti nella rete.

Per evitare **loop infiniti** e congestione, si introducono **tecniche di controllo**:

- **Hop count limit:** In IPv4 con TTL, in IPv6 Max Hop.
- **Duplicate suppression:** ogni router tiene traccia dei pacchetti già inoltrati (basandosi su ID univoci) per evitare di inviarli più volte.

Vantaggi

- **Semplicità:** non richiede informazioni sulla topologia o sullo stato dei link.
- **Affidabilità:** garantisce che il pacchetto raggiunga tutti i nodi raggiungibili, utile in scenari di emergenza o inizializzazione.
- **Ridondanza:** se esistono più percorsi, il pacchetto seguirà tutti, aumentando le possibilità di consegna in reti instabili.

Svantaggi

- **Efficienza estremamente bassa:** ogni nodo può ricevere lo stesso pacchetto molte volte, causando un'esplosione del traffico.
- **Sovraccarico della rete:** la quantità di pacchetti cresce rapidamente con il numero di nodi e link.
- **Non adatto a reti grandi:** in ambienti complessi, flooding può causare congestione grave e comportamenti imprevedibili.

Contesto d'uso

Il flooding, nonostante i suoi limiti, viene ancora utilizzato in alcune applicazioni specifiche:

- **Routing iniziale nei protocolli ad-hoc** o di discovery.
- **Invio di messaggi di controllo** in protocolli come OSPF (per l'annuncio dei link).
- **Applicazioni broadcast o multicast** in reti semplici.

3.6.2 Distance Vector

Parametri fissi

Il **distance-vector routing** è un algoritmo di routing distribuito in cui ogni router mantiene un **tavola** con la stima del costo verso tutte le destinazioni conosciute e il "next hop" da utilizzare. Queste tabelle vengono **scambiate periodicamente** con i soli router direttamente collegati, mandando solo i parametri interessati (nodo e distanza minima), quindi un **vettore**, permettendo di aggiornare continuamente le rotte basandosi sulle informazioni ricevute.

L'algoritmo si basa sull'idea dell'equazione di programmazione dinamica di Bellman-Ford:

$$d_x(y) = \min_v \{c(x, v) + d_v(y)\}$$

dove:

- $d_x(y)$ è il costo del percorso a costo minimo dal nodo x al nodo y .
- \min_v riguarda tutti i nodi "vicini" di x , infatti dopo aver viaggiato da x a v si considera il percorso minimo da v a y (con la funzione $c(a, b)$ costo da a a b).

L'algoritmo parte allora così: ciascuno nodo inizia con $D_x(y)$ una stima del costo del percorso a costo minimo da sé stesso al nodo y , per tutti i nodi in N .

Sia $D_x = [D_x(y) : y \in N]$ il vettore delle distanze del nodo x , che è il vettore delle stime dei costi da x a tutti gli altri nodi, y , in N . Grazie all'algoritmo di Bellman-Ford, ciascuno nodo x mantiene i seguenti dati di

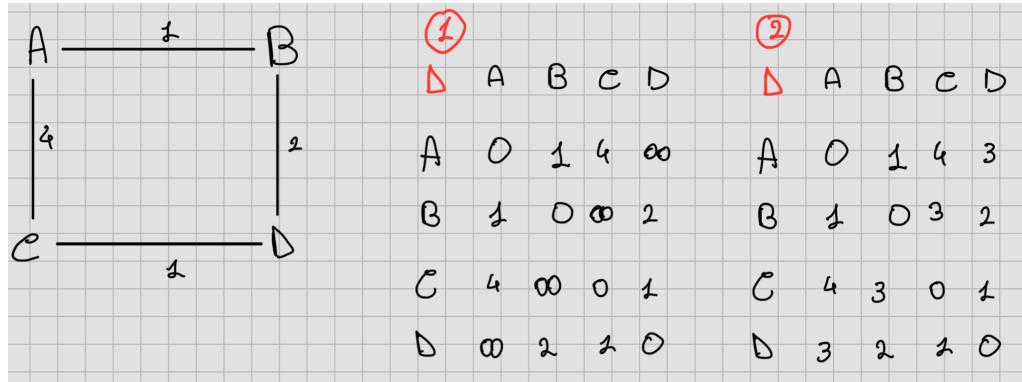
instradamento:

- Per ciascuno vicino v , il costo $c(x, v)$ da x a v .
- Il vettore delle distanze del nodo x , che è $D_x = [D_x(y) : y \in N]$, contenente la stima presso x del costo verso tutte le destinazioni, y , in N .
- I vettori delle distanze di ciascuno dei suoi vicini, ossia $D_v = [D_v(y) : y \in N]$, per ciascun vicino y di x .

Grazie a questo **algoritmo asincrono e distribuito**, si può ricostruire l'equazione di Bellman-Ford:

$$D_x(y) = \min_v \{c(x, v) + D_v(y)\}, \forall y \in N$$

Esempio

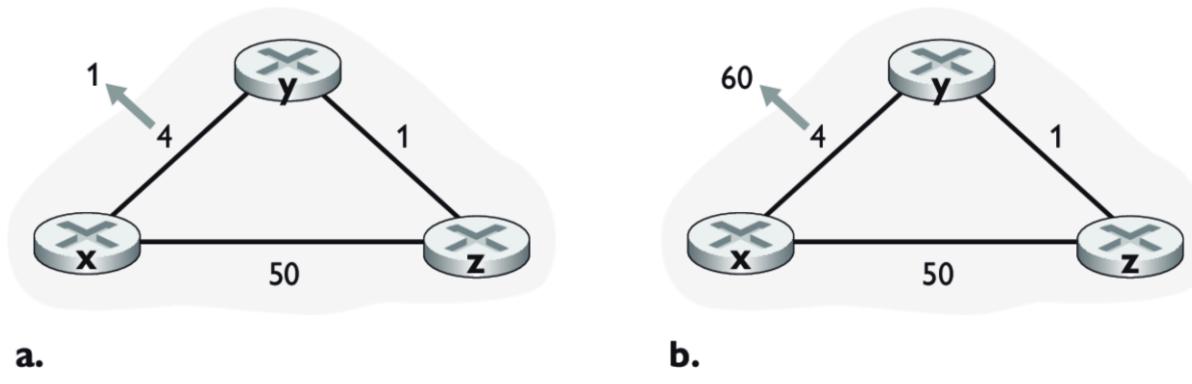


Si sa che alla prima iterazione dell'algoritmo, la tabella risulta con valori conosciuti per ogni nodo tranne per $A - D, B - C, C - B, D - A$. Già dopo la seconda iterazione, i valori si stabilizzano poiché:

- A deve scegliere, per arrivare a D se utilizzare il percorso $A \rightarrow B \rightarrow D$ (che ha valore totale 3) oppure $A \rightarrow C \rightarrow D$ (che ha valore totale 5). Poiché $3 < 5$ allora sceglie il primo e aggiorna la sua tabella.
- B deve scegliere, per arrivare a C , se utilizzare $B \rightarrow A \rightarrow C$ (5) oppure $B \rightarrow D \rightarrow C$ (3) e sceglie il secondo.
- C deve scegliere, per arrivare a B , se utilizzare $C \rightarrow A \rightarrow B$ (5) oppure $C \rightarrow D \rightarrow B$ (3) e sceglie il secondo.
- D deve scegliere, per arrivare ad A se utilizzare $D \rightarrow C \rightarrow A$ (5) oppure $D \rightarrow B \rightarrow A$ (3) e sceglie il secondo.

Parametri variabili

Quando un nodo che esegue l'algoritmo DV rileva un cambiamento nel costo dei collegamenti con un vicino aggiorna anche il proprio vettore delle distanze e se si verifica un cambiamento nel percorso a costo minimo, trasmette ai suoi vicini il proprio nuovo vettore delle distanze.



- All'istante t_0 y rileva un cambiamento nel costo del collegamento (che passa da 4 a 1) e aggiorna il proprio vettore delle distanze e informa i vicini del cambiamento.
- All'istante t_1 z riceve l'aggiornamento da y e aggiorna la propria tabella, calcola un nuovo costo minimo verso x (che passa da 5 a 2) e invia il nuovo vettore delle distanze ai vicini.
- All'istante t_2 y riceve l'aggiornamento di z e aggiorna la propria tabella delle distanze. I costi minimi di y non cambiano e y non manda alcun messaggio a z .

Quindi nell'immagine a, dopo due iterazioni, l'algoritmo raggiunge uno stato di quiete.

Prendiamo in considerazioni il caso b, quindi quando il collegamento tra x e y passa da 4 a 60:

1. Prima che il collegamento cambi abbiamo

$$a. D_y(x) = 4, D_y(z) = 1$$

$$b. D_z(y) = 1, D_z(x) = 5$$

All'istante t_0 y rileva che il costo del collegamento è passato da 4 a 60 e calcola il suo nuovo percorso:

$$\begin{aligned} D_y(x) &= \min\{c(y, x) + D_x(x), c(y, z) + D_z(x)\} \\ &= \min\{60 + 0, 1 + 5\} \\ &= 6 \end{aligned}$$

Ovviamente, con la nostra visione globale della rete, possiamo rilevare che questo nuovo costo attraverso z è errato. L'unica informazione che però il nodo y possiede è che il costo diretto verso x è 60 e che z ha ultimamente detto a y di essere in grado di raggiungere x con un costo di 5. Pertanto al fine di arrivare a x , y ora farebbe passare il percorso per z , aspettandosi che questo sia in grado di raggiungere x con un costo pari a 5.

All'istante t_1 , abbiamo un **instradamento ciclico**: al fine di giungere a x , y fa passare il percorso per z e z lo fa passare per y . Un ciclo in cui un instradamento assomiglia a un buco nero: un pacchetto destinato a x che arriva a y o a z all'istante T_1 rimbalzerà avanti e indietro.

2. Dato che il nodo y ha calcolato un nuovo costo minimo verso x , informa z del suo nuovo vettore delle distanze all'istante t_1 .
3. In un istante successivo a t_1 , z riceve il nuovo vettore delle distanze di y , che indica che il costo minimo di y verso x è 6, sa che può giungere a y a costo 1 e quindi calcola un nuovo costo minimo verso x pari a $D_z(x) = \min\{50 + 0, 1 + 6\} = 7$. Dato che il costo minimo di z verso x è aumentato, z informa y del suo nuovo vettore delle distanze al tempo t_2

4. Analogamente, dopo aver ricevuto il nuovo vettore delle distanze di y , z determina $D_y(x) = 8$ e invia a z il suo nuovo vettore delle distanze. z allora determina $D_z(x) = 9$ e invia il suo nuovo vettore delle distanze e così via.

Il ciclo prosegue per 44 iterazioni, fino a quando z considera il costo del proprio percorso attraverso y maggiore di 50.

Inversione avvelenata

Lo scenario dei cicli si può evitare con una tecnica nota come **inversione avvelenata**: se z instrada tramite y per giungere alla destinazione x , allora z avvertirà y che la sua distanza verso x è infinita, ossia comunicherà a y che $D_z(x) = +\infty$, anche se in realtà z sa che $D_z(x) = 5$ e continuerà a dire questa "bugia bianca" fintanto che instrada verso x passando per y .

Questa tecnica risolve il problema dei cicli in questo modo: quando il costo del collegamento (x, y) cambia da 4 a 60 all'istante t_0 , y aggiorna la propria tabella, ma continua a instradare direttamente verso x nonostante il costo più alto pari a 60 e informa z del suo nuovo costo verso x ossia $D_y(x) = 60$. Una volta ricevuto l'aggiornamento all'istante t_1 , z cambia immediatamente il proprio percorso verso x facendolo passare attraverso il collegamento diretto (x, y) al costo 50. Dato che questo è il nuovo percorso a costo minimo verso x e dato che non passa più attraverso y , z ora informa y che all'istante t_2 $D_z(x) = 60$. Dopo aver ricevuto l'aggiornamento da z , y adegua la propria tabella di distanza ponendo $D_y(x) = 51$. Ancora dato che z si trova ora sul percorso costo minimo di y verso x , y avvelena il percorso inverso da z a x informando z all'istante t_3 che $D_y(x) = +\infty$, anche se in realtà y sa che $D_y(x) = 51$ e così si risolve il problema.

3.6.3 Link State

In un instradamento link state **la topologia di rete e tutti i costi dei collegamenti sono noti**, ossia disponibili in input all'algoritmo. Ciò si ottiene **facendo inviare a ciascuno nodo pacchetti sullo stato dei suoi collegamenti a tutti gli altri nodi della rete**. Questi pacchetti contengono identità e costi dei collegamenti connessi al nodo che li invia.

L'algoritmo su cui si basa Link State è **Dijkstra**, è iterativo e ha delle proprietà: dopo la k -esima iterazione, i percorsi a costo minimo sono noti a k nodi di destinazione e tra i percorsi a costo minimo verso tutti i nodi di destinazione, questi k percorsi hanno i k costi più bassi. Si adotta la notazione:

- $D(v)$ costo minimo del percorso dal nodo origine alla destinazione v per quanto concerne l'iterazione corrente dell'algoritmo
- $p(v)$ immediato predecessore di v lungo il percorso a costo minimo dall'origine a v
- N' : sottoinsieme di nodi contenente tutti (e solo) i nodi v per cui il percorso a costo minimo dall'origine a v è definitivamente noto

L'algoritmo segue 5 fasi:

1. Identificazione dei vicini

Il router individua i router adiacenti tramite messaggi.

2. Determinazione del costo dei link

Ogni router misura il costo di ciascuno link verso i suoi vicini. Questo costo può essere fisso o calcolato dinamicamente (es. in base alla congestione).

3. Creazione del pacchetto link-state

Il router crea un pacchetto che elenca i propri vicini e il costo verso ciascuno

4. Diffusione dei pacchetti link-state

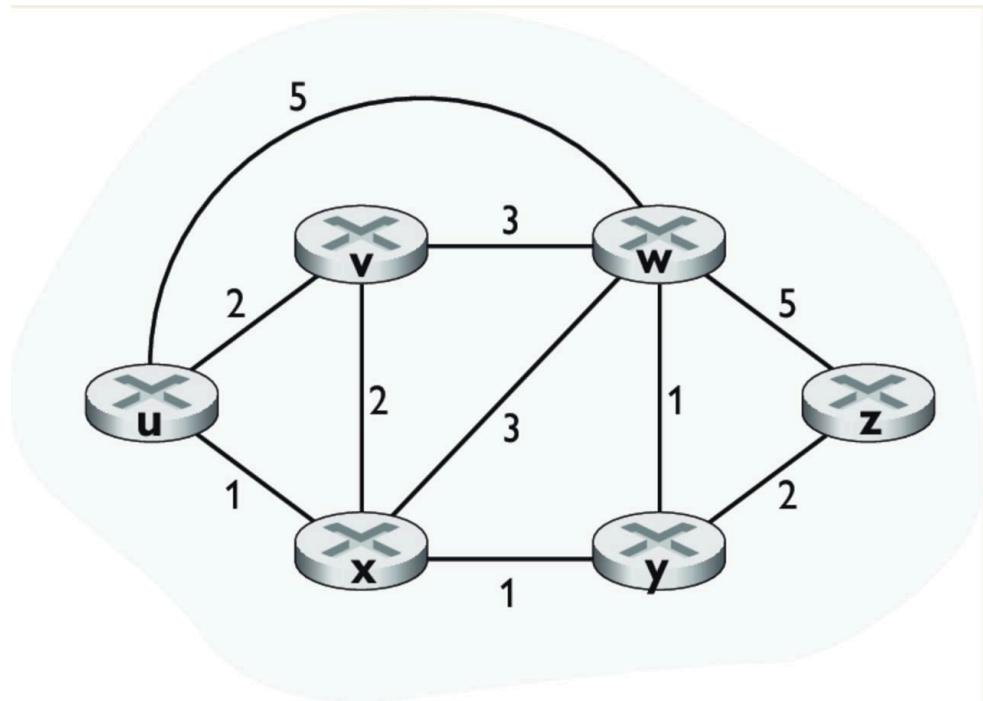
Gli LSP vengono inviati a

tutti i router della rete, tramite una procedura di **flooding affidabile**. Ogni router inoltra l'LSP ricevuto su tutte le interfacce tranne quella da cui è arrivato (così si evitano duplicati).

5. Costruzione della mappa e calcolo dei percorsi

Dopo aver raccolto tutti gli LSP, ogni router ha una rappresentazione completa della topologia. A questo punto **esegue localmente** l'algoritmo di Dijkstra per determinare il percorso minimo verso ogni nodo.

Esempio



Ipotizzando una rete di questo tipo, abbiamo che l'esecuzione dell'algoritmo crea questa tabella:

| Passo | N' | $D(v), p(v)$ | $D(w), p(w)$ | $D(x), p(x)$ | $D(y), p(y)$ | $D(z), p(z)$ |
|-------|--------|--------------|--------------|--------------|--------------|--------------|
| 0 | u | 2, u | 5, u | 1, u | ∞ | ∞ |
| 1 | ux | 2, u | 4, x | | 2, x | ∞ |
| 2 | uxy | 2, u | 3, y | | | 4, y |
| 3 | uxyv | | 3, y | | | 4, y |
| 4 | uxyvw | | | | | 4, y |
| 5 | uxyvwz | | | | | |

3.6.4 Confronto tra DV e LS

| Caratteristica | Distance Vector | Link State |
|-----------------------------|---------------------|--------------------|
| Tipo di informazione | Solo verso i vicini | Topologia completa |
| Metodo di update | Scambio vettori | Flooding di LSP |
| Algoritmo | Bellman-Ford | Dijkstra |
| Convergenza | Lenta | Rapida |
| Rischio di loop | Presente | Ridotto |
| Complessità | Bassa | Alta |

3.7 Sistemi autonomi

Nel routing su larga scala, come quello che avviene su Internet, l'intera rete globale non è gestita come un'unica entità. Per semplificare la gestione e garantire scalabilità, l'Internet è suddiviso in **sistemi autonomi (AS, Autonomous Systems)**.

Un sistema autonomo è un gruppo di router e reti **sotto un unico dominio amministrativo**, che condivide **una politica di instradamento comune**. Ad esempio, un'università, un provider Internet o un'azienda possono costituire un AS.

3.7.1 RIP – Routing Information Protocol

Il **Routing Information Protocol (RIP)** è un protocollo di **routing intra-AS**, ovvero usato **all'interno di un sistema autonomo**, e rappresenta uno dei più antichi e semplici protocolli distance-vector usati in Internet.

RIP implementa l'algoritmo **distance-vector**, seguendo il modello periodico di aggiornamento:

- Ogni **30 secondi**, ogni router invia la propria tabella di instradamento ai suoi vicini.
- Gli aggiornamenti includono il **numero di hop** (salti) per raggiungere ogni destinazione.
- Se un router riceve un aggiornamento che riduce il numero di hop per una destinazione, aggiorna la propria tabella.

Caratteristiche principali

- La **metrica di costo** è il **numero di hop**, con un massimo di **15**: un valore di **16** è considerato **infinito**, e indica una rete non raggiungibile.
- Utilizza UDP (porta 520) per il trasporto dei messaggi RIP.
- È un protocollo semplice, leggero, e adatto a **reti piccole o statiche**.

Limitazioni

- Non scala bene per reti ampie: la convergenza può essere lenta, e il conteggio infinito limita la propagazione.
- Vulnerabile ai problemi tipici del distance-vector: **loop**, **count-to-infinity**, e aggiornamenti lenti in caso di guasti.

Versioni

- **RIPv1**: il numero massimo di HOP consentiti è 15, se il percorso per un router non è aggiornato ogni 180 secondi viene impostato a infinito.
- **RIPv2**: utilizza poison reverse per DV, ha l'autenticazione dei messaggi e l'autenticazione a ogni nexthop.
- **RIPng**: è basato su RIPv2 ma non è un'estensione. E' progettato solo per IPv6, senza supporto per IPv4. Contiene tutte le feature di RIPv2 tranne l'autenticazione.

3.7.2 OSPF – Open Shortest Path First

Nel mondo reale, il protocollo RIP non è più sufficiente per gestire reti complesse e dinamiche. È per questo che si è affermato **OSPF (Open Shortest Path First)**, un protocollo di routing **intra-dominio** che implementa un approccio **link-state**, molto più scalabile e preciso rispetto ai vecchi protocolli di tipo distance-vector.

L'idea alla base di OSPF è che ogni router non si affida solo a ciò che i vicini gli dicono, ma cerca di ottenere una **visione completa della topologia della rete**. Per fare questo, ciascun router invia dei messaggi contenenti informazioni sui link a cui è connesso e sui relativi costi, e li propaga con un meccanismo di **flooding affidabile**. Tutti i router ricevono questi dati e costruiscono localmente una mappa della rete, che usano per calcolare i percorsi minimi tramite **l'algoritmo di Dijkstra**.

A differenza di RIP, che conta i salti (hop), OSPF permette di **personalizzare il costo dei link**, tenendo conto della capacità, della latenza o di altri criteri. Inoltre, OSPF reagisce molto più rapidamente ai cambiamenti, perché non invia aggiornamenti a intervalli fissi, ma solo quando c'è una variazione nella rete.

Un altro vantaggio importante di OSPF è la possibilità di **strutturare una rete in più aree**. Questo approccio gerarchico consente di contenere il traffico di routing e semplificare la gestione: l'area backbone (area 0) connette tutte le altre e funge da perno del sistema.

In ambienti aziendali o nei provider, OSPF è spesso la prima scelta: è **standard, veloce, sicuro** (supporta l'autenticazione dei messaggi), e consente perfino il **load balancing** su percorsi multipli con costo identico. È un protocollo maturo, estremamente affidabile e, nel suo funzionamento, rappresenta l'applicazione concreta dei concetti studiati nell'algoritmo link-state.

Intestazione

| Version | Type | Message Length |
|--------------------------|--------------------|----------------|
| Router ID | | |
| Area ID | | |
| Checksum | Autentication Type | |
| Autentication Data | | |
| Rest of the OSPF Message | | |

Type:

1. Hello
2. DB Description Router ID: router che ha generato il messaggio
3. LS Request
4. LS Update AreaID: Ip dell'area a cui si riferisce il messaggio
5. LS ACK

3.7.3 BGP – Border Gateway Protocol

Quando i pacchetti devono attraversare **diversi sistemi autonomi (AS)** su scala globale, come avviene ogni giorno su Internet, è necessario un protocollo in grado di gestire routing **inter-AS**. Questo ruolo è svolto dal **BGP (Border Gateway Protocol)**, il cuore del routing su Internet.

Diversamente da RIP o OSPF, che sono protocolli intra-AS, BGP gestisce le rotte tra AS distinti, ciascuno con le proprie politiche e preferenze. BGP non cerca il percorso più breve, ma quello **più conforme alle politiche di peering, sicurezza e gestione del traffico** tra organizzazioni. Questo lo rende un protocollo **basato su politiche** più che su metriche classiche.