Appunti del corso di
Sistemi di elaborazione: Reti I

PROF. G. BONGIOVANNI

5) Il livello tre (Network)

Il livello network è incaricato di muovere i pacchetti dalla sorgente fino alla destinazione finale, attraversando tanti sistemi intermedi (***router***) della subnet di comunicazione quanti è necessario.

Ciò è molto diverso dal compito del livello data link, che è di muovere informazioni solo da un capo all'altro di un singolo canale di comunicazione wire-like.

Le incombenze principali di questo livello sono:
- conoscere la topologia della rete;
- scegliere di volta in volta il cammino migliore (***routing***);
- gestire il flusso dei dati e le congestioni (***flow control*** e ***congestion control***);
- gestire le problematiche derivanti dalla presenza di più reti diverse (***internetworking***).

Nel progetto e nella realizzazione del livello network di una architettura di rete si devono prendere decisioni importanti in merito a:
- ***servizi offerti*** al livello transport;
- ***organizzazione interna*** della subnet di comunicazione.

5.1) Servizi offerti

In merito ai servizi offerti al livello superiore, ci sono (come abbiamo già accennato) due tipologie fondamentali di servizi:
- servizi connection-oriented;
- servizi connectionless.

In proposito, ci sono due scuole di pensiero:
- fautori dei servizi connection-oriented (compagnie telefoniche);
- fautori dei servizi connectionless (Internet Community).

La prima scuola di pensiero afferma che il livello network deve fornire un servizio sostanzialmente affidabile e orientato alla connessione. In questa visione, succede che:
- le peer entitiy stabiliscono una connessione, negoziandone i parametri (di qualità, di costo, ecc.), alla quale viene associato un identificatore;
- tale identificatore viene inserito in ogni pacchetto che verrà inviato;
- la comunicazione è bidirezionale e i pacchetti viaggiano, in sequenza, lungo il cammino assegnato alla connessione;

- il controllo di flusso è fornito automaticamente (attraverso alcuni dei parametri negoziati, come ad esempio il dimensionamento di una o più finestre scorrevoli).

La seconda scuola di pensiero ritiene invece che la sottorete debba solo muovere dati e nient'altro:
- la sottorete è giudicata inerentemente inaffidabile, per cui gli host devono provvedere per conto proprio alla correzione degli errori e al controllo di flusso;
- una ovvia conseguenza è che il servizio offerto dal livello network dev'essere datagram, visto che è inutile inserire le funzioni di controllo degli errori e del flusso in due diversi livelli;
- i pacchetti viaggiano indipendentemente, e dunque devono tutti contenere un identificatore (ossia l'indirizzo) della destinazione.

Di fatto, il problema è dove mettere la complessità della realizzazione:
- la prima scuola la mette nei nodi della subnet, che si devono occupare del ***setup delle connessioni*** e di fornire la necessaria affidabilità;
- la seconda scuola la mette negli host, i cui livelli transport forniscono l'affidabilità e l'orientamento alla connessione.

In realtà le decisioni sono due, separate:
- offrire o no un servizio affidabile;
- offrire o no un servizio orientato alla connessione.

Le scelte più comuni sono di offrire ***servizi connection oriented affidabili*** oppure ***servizi connectionless non affidabili***, mentre le altre due combinazioni, anche se tecnicamente possibili, non sono diffuse.

5.2) Organizzazione interna della subnet

Questo è un problema separato ed indipendente da quello dei servizi offerti, anche se spesso c'è una relazione fra i due.

Una subnet può essere organizzata con un funzionamento interno:
- basato su connessioni (che in questo contesto si chiamano ***circuiti virtuali***):
    - la subnet stabilisce un circuito virtuale (sul quale verrà tipicamente veicolato il traffico di un servizio connection oriented), cioé crea un cammino fra la sorgente e la destinazione;
    - tutti i router lungo tale cammino ricordano, in una apposita struttura dati, la parte di loro competenza di tale cammino (e cioé quale linea in entrata e quale in uscita sono assegnate al cammino);

    - quando arrivano pacchetti che contengono l' ID di tale circuito virtuale, essi vengono instradati di conseguenza (tutti nello stesso modo).
- connectionless:
    - i router si limitano a instradare ogni pacchetto che arriva sulla base del suo indirizzo di destinazione, decidendo di volta in volta come farlo proseguire;
    - i router hanno delle ***tabelle di instradamento*** (***routing table***)che indicano, per ogni possibile destinazione, quale linea in uscita utilizzare; si noti che queste tabelle esistono anche nelle subnet del tipo precedente, dove però servono solamente nella fase di setup della connessione (per decidere come instradare i pacchetti di setup);
    - quando offre un servizio connection-oriented, questo livello fa credere al livello superiore che esista una connessione, ma poi i pacchetti viaggiano indipendentemente (e quindi hanno tutti l'indirizzo del destinatario) e vengono rimessi in ordine dal livello network solo a destinazione, prima di essere consegnati al livello superiore.

Ognuna delle due organizzazioni della subnet sopra viste ha i suoi supporter e i suoi detrattori, anche sulla base delle seguenti considerazioni:

	Subnet basata su connessioni	**Subnet connectionless**
Banda trasmissiva	***Minore*** (piccole ID in ogni pacchetto)	***Maggiore*** (intero indirizzo di dest. in ogni pacchetto)
Spazio sui router	***Maggiore*** (strutture dati per i circuiti virtuali)	***Minore***
Ritardo per il setup	***Presente***	***Assente***
Ritardo per il routing	***Assente***	***Presente***
Congestione	***Minore*** (risorse allocate in anticipo)	***Maggiore*** (possibile in ogni momento)
Vulnerabilità	***Alta***	***Bassa***

Dev'essere chiaro che i servizi offerti sono indipendenti dalla realizzazione interna della subnet. E' possibile avere tutte le quattro combinazioni di servizio offerto e implementazione della subnet:

- servizi connection oriented su circuiti virtuali;
- servizi connectionless su subnet datagram;
- servizi connection oriented su subnet datagram (si cerca di fornire comunque un servizio robusto);
- servizi connectionless su circuito virtuale (esempio: IP su subnet ATM).

5.3) Algoritmi di routing

La funzione principale del livello network è di instradare i pacchetti sulla subnet, tipicamente facendo fare loro molti ***hop*** (letteralmente, salti) da un router ad un altro.

Un ***algoritmo di routing*** è quella parte del software di livello network che decide su quale linea di uscita instradare un pacchetto che è arrivato:
- in una subnet datagram l'algoritmo viene applicato ex novo ad ogni pacchetto;
- in una subnet basata su circuiti virtuali l'algoritmo viene applicato solo nella fase di setup del circuito; in tale contesto si usa spesso il termine ***session routing***.

Da un algoritmo di routing desideriamo:
- correttezza (deve muovere il pacchetto nella giusta direzione);
- semplicità (l'implementazione non deve essere troppo complicata);
- robustezza (deve funzionare anche in caso di cadute di linee e/o router e di riconfigurazioni della topologia);
- stabilità (deve convergere, e possibilmente in fretta);
- equità (non deve favorire nessuno);
- ottimalità (deve scegliere la soluzione globalmente migliore.

Purtroppo, gli ultimi due requisiti sono spesso in conflitto fra loro; inoltre, a proposito dell'ottimalità, non sempre è chiaro cosa si voglia ottimizzare. Infatti, supponiamo che si vogliano:
- minimizzare il ritardo medio pacchetti;
- massimizzare il throughput totale dei pacchetti.

Si scopre facilmente che questi due obiettivi sono in conflitto fra loro, perché di solito aumentare il throughput allunga le code sui router e quindi aumenta il ritardo: questo è vero per qualunque sistema basato su code gestito in prossimità della sua capacità massima.

Gli algoritmi di routing si dividono in due classi principali:
- ***algoritmi non adattivi*** (***static routing***):
    - le decisioni di routing sono prese in anticipo, all'avvio della rete, e sono comunicate ai router che poi si attengono sempre a quelle;
- ***algoritmi adattivi*** (***dynamic routing***):
    - le decisioni di routing sono riformulate (sulla base del traffico, della topologia della rete, ecc.) molto spesso.

Gli algoritmi adattivi differiscono fra loro per:
- come ricevono le informazioni:

    - localmente;
    - dai router adiacenti;
    - da tutti i router;
- quanto spesso rivedono le decisioni:
    - a intervalli di tempo prefissati;
    - quando il carico cambia;
    - quando la topologia cambia;
- quale metrica di valutazione adottano:
    - distanza;
    - numero di hop;
    - tempo di transito stimato.

5.3.1) Il principio di ottimalità

E' possibile fare una considerazione generale sull'ottimalità dei cammini, indipendentemente dallo specifico algoritmo adottato per selezionarli.

Il ***principio di ottimalità*** dice che se il router j è nel cammino ottimo fra i e k, allora anche il cammino ottimo fra j e k è sulla stessa strada:

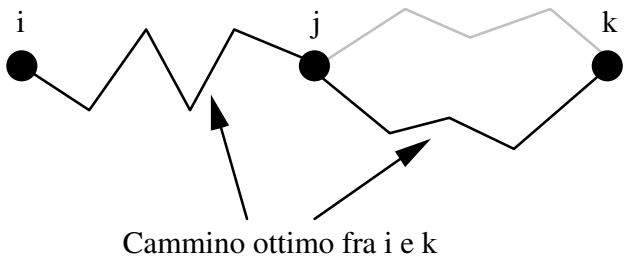

Figura 5-1: Principio di ottimalità

Se così non fosse, ci sarebbe un altro cammino (ad es. quello tratteggiato in figura) fra j e k migliore di quello che è parte del cammino ottimo fra i e k, ma allora ci sarebbe anche un cammino fra i e k migliore di quello ottimo.

Una diretta conseguenza è che l'insieme dei cammini ottimi da tutti i router a uno specifico router di destinazione costituiscono un ***albero***, detto ***sink tree*** per quel router.

In sostanza, gli algoritmi di routing cercano e trovano i sink tree relativi a tutti i possibili router di destinazione, e quindi instradano i pacchetti esclusivamente lungo tali sink tree.

5.3.2) Algoritmi statici

Questi algoritmi, come abbiamo già accennato, sono eseguiti solamente all'avvio della rete, e le decisioni di routing a cui essi pervengono sono poi applicate senza più essere modificate.

Esamineremo i seguenti algoritmi statici:
- ***shortest path routing***;
- ***flooding***;
- ***flow-based routing***.

Shortest path routing
L'idea è semplice: un host di gestione della rete mantiene un grafo che rappresenta la subnet:
- i nodi rappresentano i router;
- gli archi rappresentano le linee punto-punto.

All'avvio della rete (o quando ci sono variazioni permanenti della topologia) l'algoritmo:
- applica al grafo un algoritmo per il calcolo del ***cammino minimo*** fra ogni coppia di nodi; ad esempio, il noto algoritmo di Dijkstra ('59) può essere usato;
- invia tali informazioni a tutti i router.

Quale sia il cammino minimo dipende da qual'è la grandezza che si vuole minimizzare. Tipicamente si usano:
- numero di hop, cioè di archi, da attraversare;
- lunghezza dei collegamenti;
- tempo medio di accodamento e trasmissione;
- una combinazione di lunghezza, banda trasmissiva, traffico medio, ecc.

Flooding
La tecnica del ***flooding*** consiste nell'inviare ogni pacchetto su tutte le linee eccetto quella da cui è arrivato.

In linea di principio il flooding può essere usato come algoritmo di routing (ogni pacchetto inviato arriva a tutti i router) ma presenta l'inconveniente di generare un numero enorme (teoricamente infinito!) di pacchetti.

Ci sono delle tecniche per limitare il traffico generato:
- inserire in ogni pacchetto un ***contatore*** che viene decrementato ad ogni hop. Quando il contatore arriva a zero, il pacchetto viene scartato. Un appropriato valore iniziale può essere il diametro della subnet;

- inserire la coppia (***source router ID, sequence number***) in ogni pacchetto. Ogni router esamina tali informazioni e ne tiene traccia, e quando le vede per la seconda volta scarta il pacchetto;
- ***selective flooding***: i pacchetti vengono duplicati solo sulle linee che vanno all'incirca nella giusta direzione (per questo si devono mantenere apposite tabelle a bordo).

Il flooding non è utilizzabile in generale come algoritmo di routing, però:
- è utile in campo militare (offre la massima affidabilità e robustezza);
- è utile per l'aggiornamento contemporaneo di informazioni distribuite;
- è utile come strumento di paragone per altri algoritmi, visto che trova sempre, fra gli altri, il cammino minimo.

Flow-based routing
Questo algoritmo è basato sull'idea di:
- calcolare in anticipo il traffico atteso su ogni linea;
- da questi calcoli derivare una stima del ritardo medio atteso per ciascuna linea;
- basare su tali informazioni le decisioni di routing.

Le informazioni necessarie per poter applicare l'algoritmo sono:
- la topologia della rete;
- la matrice delle quantità di traffico T(i,j) stimate fra ogni coppia (i,j) di router;
- le capacità (in bps ad esempio) delle linee point to point.

Vengono fatte le seguenti assunzioni:
- il traffico è stabile nel tempo e noto in anticipo;
- il ritardo su ciascuna linea aumenta all'aumentare del traffico sulla linea e diminuisce all'aumentare della velocità della linea secondo le leggi della teoria delle code.

Dai ritardi calcolati per le singole linee si può calcolare il ritardo medio dell'intera rete, espresso come somma pesata dei ritardi delle singole linee. Il peso di ogni linea è dato dal traffico su quella linea diviso il traffico totale sulla rete.

Il metodo nel suo complesso funziona così:
- si sceglie un algoritmo di routing;
- sulla base di tale algoritmo si determinano i percorsi che verranno seguiti per il collegamento fra ogni coppia di router;
- si calcola il traffico che incide su ogni linea (che è uguale alla somma di tutti i T(i,j) instradati su quella linea);
- si calcola il ritardo di ogni linea;
- si calcola il ritardo medio della rete;
- si ripete il procedimento con vari algoritmi di routing, scegliendo alla fine quello che minimizza il ritardo medio dell'intera rete.

5.3.3) Algoritmi dinamici

Nelle moderne reti si usano algoritmi dinamici, che si adattano automaticamente ai cambiamenti della rete. Questi algoritmi non sono eseguiti solo all'avvio della rete, ma rimangono in esecuzione sui router durante il normale funzionamento della rete.

Distance vector routing
Ogni router mantiene una tabella (***vector***) contenente un elemento per ogni altro router. Ogni elemento della tabella contiene:

- la distanza (numero di hop, ritardo, ecc.) che lo separa dal router in oggetto;
- la linea in uscita da usare per arrivarci;

Per i suoi vicini immediati il router stima direttamente la distanza dei collegamenti corrispondenti, mandando speciali ***pacchetti ECHO*** e misurando quanto tempo ci mette la risposta a tornare.

A intervalli regolari ogni router manda la sua tabella completa (cioè il suo vector) a tutti i vicini, e riceve quelle dei vicini.

Quando un router ha ricevuto i vector dei vicini, calcola i nuovi valori da inserire nel proprio vector. Analizzando a turno i vector di tutti i propri vicini e per ogni possibile router di destinazione X, calcola la concatenazione migliore (cioè di minor costo complessivo) di

percorso fra se stesso e vicino immediato , percorso da quel vicino immediato a X

Ovviamente, la migliore è la concatenazione che produce la minore somma di:

- distanza fra il router stesso ed un suo vicino immediato (viene dalla misurazione diretta);
- distanza fra quel vicino immediato ed il router remoto di destinazione (viene dalla tabella ricevuta da quel vicino immediato).

L'algoritmo distance vector routing funziona piuttosto bene, ma è molto lento nel reagire alle cattive notizie, cioè quando un collegamento va giù. Ciò è legato al fatto che i router non conoscono la topologia della rete.

Infatti, consideriamo questo esempio:

```
A     B     C     D     E   <- Router
*-----*-----*-----*-----*   <- Collegamenti (topologia lineare)
      1     2     3     4   <- Distanze da A
```

Se ora cade la linea fra A e B, dopo uno scambio succede questo:

```
A     B     C     D     E   <- Router
*     *-----*-----*-----*   <- Collegamenti
      3     2     3     4   <- Distanze da A (dopo uno scambio)
```

Ciò perché B, non ricevendo risposta da A, crede di poterci arrivare via C, che ha distanza due da A. Col proseguire degli scambi, si ha la seguente evoluzione:

```
A     B     C     D     E   <- Router
*     *-----*-----*-----*   <- Collegamenti
      3     4     3     4   <- Distanze da A (dopo due scambi)
      5     4     5     4   <- Distanze da A (dopo tre scambi)
      5     6     5     6   <- Distanze da A (dopo quattro scambi)
ecc.
```

A lungo andare, tutti i router vedono lentamente aumentare sempre più la distanza per arrivare ad A. Questo è il problema del ***count-to-infinity***.

Se la distanza rappresenta il numero di hop si può porre come limite il diametro della rete, ma se essa rappresenta il ritardo l'upper bound dev'essere molto alto, altrimenti cammini con un ritardo occasionalmente alto (magari a causa di congestione) verrebbero considerati interrotti.

Sono state proposte molte soluzioni al problema count-to-infinity, ma nessuna veramente efficace.

Nonostante ciò, il distance vector routing era l'algoritmo di routing di ARPANET ed è usato anche in Internet col nome di ***RIP*** (***Routing Internet Protocol***), e nelle prime versioni di DECnet e IPX.

Link state routing
Sopratutto a causa della lentezza di convergenza del distance vector routing, si è cercato un approccio diverso, che ha dato origine al ***link state routing***.

L'idea è questa:
- ogni router tiene sott'occhio lo stato dei collegamenti fra se e i suoi vicini immediati (misurando il ritardo di ogni linea) e distribuisce tali informazioni a tutti gli altri;
- sulla base di tali informazioni, ogni router ricostruisce localmente la topologia completa dell'intera rete e calcola il cammino minimo fra se e tutti gli altri.

I passi da seguire sono:
1. scoprire i vicini e identificarli;
2. misurare il costo (ritardo o altro) delle relative linee;
3. costruire un pacchetto con tali informazioni;
4. mandare il pacchetto a tutti gli altri router;
5. previa ricezione degli analoghi pacchetti che arrivano dagli altri router, costruire la topologia dell'intera rete;
6. calcolare il cammino più breve a tutti gli altri router.

Quando il router si avvia, invia un ***pacchetto HELLO*** su tutte le linee in uscita. In risposta riceve dai vicini i loro indirizzi (univoci in tutta la rete).

Inviando vari pacchetti ECHO, misurando il tempo di arrivo della risposta (diviso 2) e mediando su vari pacchetti si deriva il ritardo della linea.

Si costruisce un pacchetto con:
- identità del mittente;
- numero di sequenza del pacchetto;
- età del pacchetto;
- lista dei vicini con i relativi ritardi.

La costruzione e l'invio di tali pacchetti si verifica tipicamente:
- a intervalli regolari;
- quando accade un evento significativo (es.: una linea va giù o torna su).

La distribuzione dei pacchetti è la parte più delicata, perché errori in questa fase possono portare qualche router ad avere idee sbagliate sulla topologia, con conseguenti malfunzionamenti.

Di base si usa il flooding, inserendo nei pacchetti le coppie (source router ID, sequence number) per eliminare i duplicati. Tutti i pacchetti sono confermati. Inoltre, per evitare che pacchetti vaganti (per qualche errore) girino per sempre, l'età del pacchetto viene decrementata ogni secondo, e quando arriva a zero il pacchetto viene scartato.

Combinando tutte le informazioni arrivate, ogni router costruisce il grafo della subnet e calcola il cammino minimo a tutti gli altri router.

Il link state routing è molto usato attualmente:
- in Internet ***OSPF*** (***Open Shortest Path First***) è basato su tale principio e si avvia ad essere l'algoritmo più utilizzato;
- un altro importante esempio è ***IS-IS*** (***Intermediate System-Intermediate System***), progettato per DECnet e poi adottato da OSI. La sua principale caratteristica è di poter gestire indirizzi di diverse architetture (OSI, IP, IPX) per cui può essere usato in reti miste o multiprotocollo.

5.3.4) Routing gerarchico

Quando la rete cresce fino contenere decine di migliaia di nodi, diventa troppo gravoso mantenere in ogni router la completa topologia. Il routing va quindi impostato in modo gerarchico, come succede nei sistemi telefonici.

La rete viene divisa in ***zone*** (spesso dette ***regioni***):
- all'interno di una regione vale quanto visto finora, cioè i router (detti ***router interni***) sanno come arrivare a tutti gli altri router della regione;
- viceversa, quando un router interno deve spedire qualcosa a un router di un'altra regione sa soltanto che deve farlo pervenire a un particolare router della propria regione, detto ***router di confine***.
- il router di confine sa a quale altro router di confine deve inviare i dati perché arrivino alla regione di destinazione.

Di conseguenza, ci solo due livelli di routing:
- un primo livello di routing all'interno di ogni regione;
- un secondo livello di routing fra tutti i router di confine.

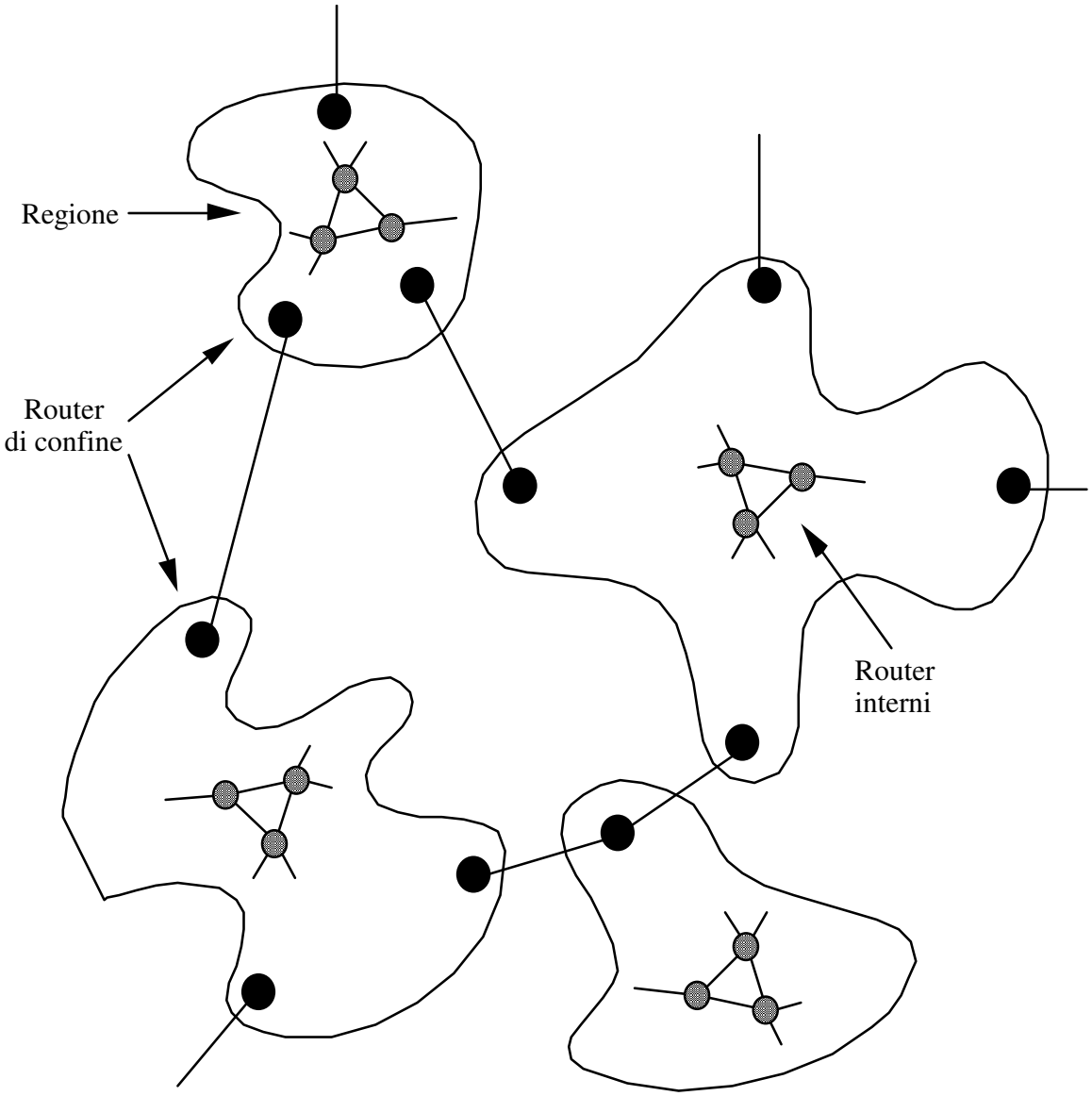

Figura 5-2: Routing gerarchico

I router interni mantengono nelle loro tabelle di routing:

- una entrata per ogni altro router interno, con la relativa linea da usare per arrivarci;
- una entrata per ogni altra regione, con l'indicazione del relativo router di confine e della linea da usare per arrivarci.

I router di confine, invece, mantengono:

- una entrata per ogni altra regione, con l'indicazione del prossimo router di confine da contattare e della linea da usare per arrivarci.

Non è detto che due livelli siano sufficienti. In tal caso il discorso si ripete su più livelli.

5.4) Controllo della congestione

Quando troppi pacchetti sono presenti in una parte della subnet, si verifica una ***congestione*** che degrada le prestazioni. Ciò dipende dal fatto che, quando un router non riesce a gestire tutti i pacchetti che gli arrivano, comincia a perderli, e ciò causa delle ritrasmissioni che aggravano ancor più la congestione.

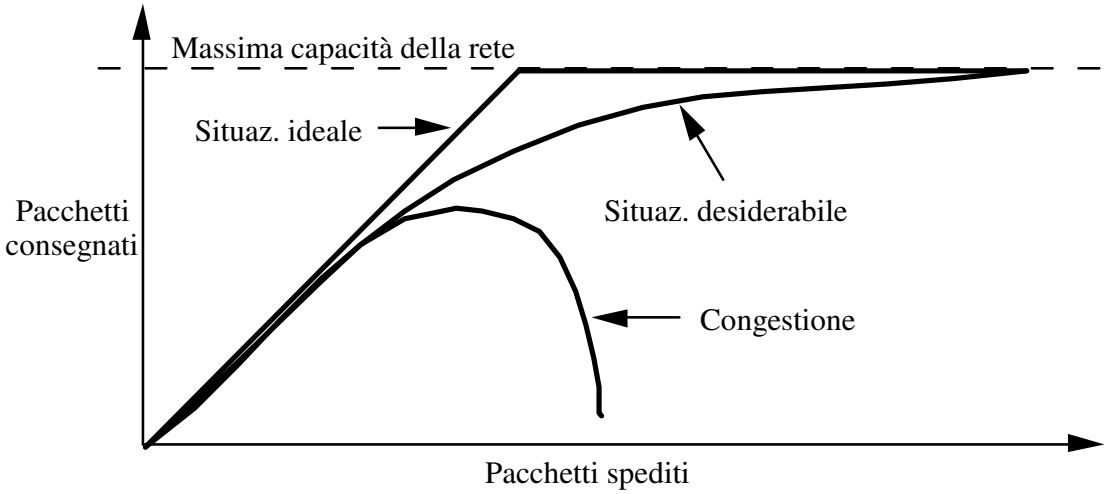

Figura 5-3: Effetti della congestione

La congestione in un router può derivare da diversi fattori:
- troppi pochi buffer nel router;
- processore troppo lento nel router;
- linea di trasmissione troppo lenta (si allunga la coda nel router di partenza).

Inoltre, la congestione in un router tende a propagarsi ai suoi vicini che gli inviano dati. Infatti, quando tale router è costretto a scartare i pacchetti che riceve non li conferma più, e quindi i router che li hanno spediti devono mantenerli nei propri buffer, aggravando così anche la propria situazione.

Il ***controllo della congestione*** è un problema globale di tutta la rete, ed è ben diverso dal problema del controllo di flusso nei livelli data link, network (nel caso dei servizi connection oriented) e trasporto, che invece riguarda una singola connessione sorgente-destinazione.

Ci sono due approcci al problema della congestione:
- *open loop* (senza controreazione);
- *closed loop* (con controreazione).

Il primo cerca di impostare le cose in modo che la congestione non si verifichi, ma poi non effettua azioni correttive.

Il secondo tiene sott'occhio la situazione della rete, intraprendendo le azioni opportune quando necessario.

5.4.1) Traffic shaping

In questo approccio, di tipo open loop, l'idea è di forzare la trasmissione dei pacchetti a un ritmo piuttosto regolare, onde limitare la possibilità di congestioni.

Verdemo tre tecniche per implementare il traffic shaping:
- *leaky bucket*;
- *token bucket*;
- *flow specification*.

Algoritmo Leaky bucket (secchio che perde)
L'idea è semplice, e trova un analogia reale in un secchio che viene riempito da un rubinetto (che può essere continuamente manovrato in modo da risultare più o meno aperto) e riversa l'acqua che contiene attraverso un forellino sul fondo, a ritmo costante. Se viene immessa troppa acqua, essa fuoriesce dal bordo superiore del secchio e si perde.

Sull'host si realizza (nell'interfaccia di rete o in software) un leaky bucket, che è autorizzato a riversare sulla rete pacchetti con un fissato data rate (diciamo b bps) e che mantiene, nei suoi buffer, quelli accodati per la trasmissione.

Se l'host genera più pacchetti di quelli che possono essere contenuti nei buffer, essi si perdono.

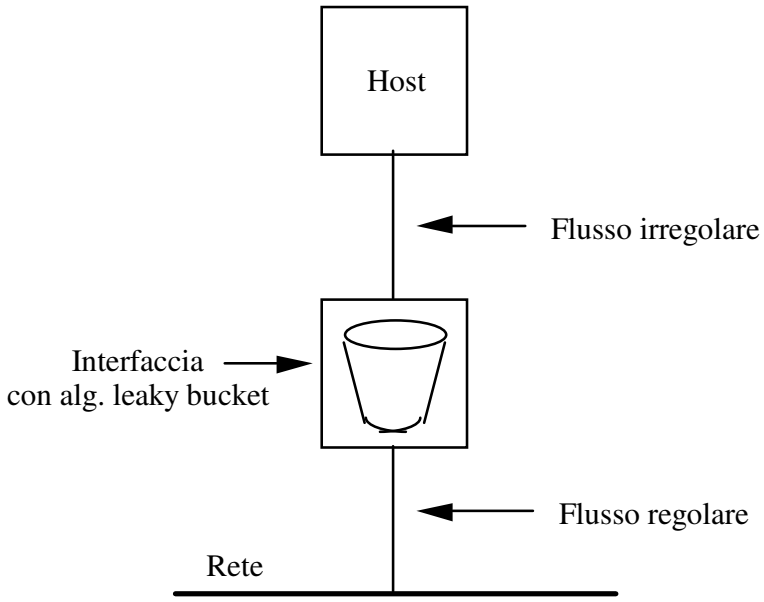

Figura 5-4: Leaky bucket

In questo modo, l'host può anche produrre un traffico bursty senza creare problemi sulla rete; finché il data rate medio non supera i b bps tutto funziona regolarmente, oltre si cominciano a perdere pacchetti.

Algoritmo token bucket (secchio di gettoni)
E' una tecnica per consentire un grado di irregolarità controllato anche nel flusso che esce sulla rete.

Essenzialmente, si accumula un credito trasmissivo con un certo data rate (fino ad un massimo consentito) quando non si trasmette nulla.

Quando poi c'è da trasmettere, lo si fa sfruttando tutto il credito disponibile per trasmettere, fino all'esaurimento di tale credito, alla massima velocità consentita dalla linea.

Il secchio contiene dei ***token***, che si creano con una cadenza prefissata (ad esempio uno ogni millisecondo) fino a che il loro numero raggiunge un valore M prefissato, che corrisponde all'aver riempito il secchio di token.

Per poter tramettere un pacchetto (o una certa quantità di byte), deve essere disponibile un token.

Se ci sono k token nel secchiello e h > k pacchetti da trasmettere, i primi k sono trasmessi subito (al data rate consentito dalla linea) e gli altri devono aspettare dei nuovi token.

Dunque, potenzialmente dei burst di M pacchetti possono essere trasmessi in un colpo solo, fermo restando che mediamente non si riesce a trasmettere ad una velocità più alta di quella di generazione dei token.

Un'altra differenza col leaky bucket è che i pacchetti non vengono mai scartati (il secchio contiene token, non pacchetti). Se necessario, si avverte il livello superiore, produttore dei dati, di fermarsi per un pò.

Questi due algoritmi possono essere usati per regolare il traffico host-router e router-router; in quest'ultimo caso però, se il router sorgente è costretto a fermarsi invece di inviare dati e non ha spazio di buffer a sufficienza, questi possono perdersi.

Flow specification
Il traffic shaping è molto efficace se tutti (sorgente, subnet e destinazione) si accordano in merito.

Un modo di ottenere tale accordo consiste nello specificare:
- le caratteristiche del traffico che si vuole inviare (data rate, grado di burstiness, ecc.);
- la qualità del servizio (ritardo massimo, frazione di pacchetti che si può perdere, ecc.).

Tale accordo si chiama ***flow specification*** e consiste in una struttura dati che descrive le grandezze in questione.

Sorgente, subnet e destinatario si accordano di conseguenza per la trasmissione.
Questo accordo viene preso prima di trasmettere, e può essere fatto sia in subnet connesse (e allora si riferisce al circuito virtuale) che in subnet non connesse (e allora si riferisce alla sequenza di pacchetti che sarà trasmessa).

Controllo della congestione nelle reti basate su circuiti virtuali
In generale nelle reti connesse è più facile il controllo della congestione, perché le risorse per una connessione sono allocate in anticipo.

Quindi, per evitare la congestione è possibile negare l'attivazione di nuovi circuiti virtuali ove non vi siano sufficienti risorse per gestirli. Questa tecnica va sotto il nome di ***admission control***.

5.4.2) Choke packet

In questo approccio, di tipo closed loop, è previsto che un router tenga d'occhio il grado di utilizzo delle sue linee di uscita. Il router misura, per ciascuna linea, l'utilizzo istantaneo U e accumula, entro una media esponenziale M, la storia passata:

$$M_{nuovo} = a\, M_{vecchio} + (1 - a)U$$

dove
- il parametro a (compreso fra 0 ed 1) è il peso dato alla storia passata;
- (1 - a) è il peso dato all'informazione più recente.

Quando, per una delle linee in uscita, M si avvicina a una soglia di pericolo prefissata, il router esamina i pacchetti in ingresso per vedere se sono destinati alla linea d'uscita che è in allarme.

In caso affermativo, invia all'host di origine del pacchetto un ***choke packet*** (to choke significa soffocare) per avvertirlo di diminuire il flusso.

Quando l'host sorgente riceve il choke packet diminuisce il flusso (tipicamente lo dimezza) e ignora i successivi choke packet per un tempo prefissato, perché tipicamente ne arriveranno molti in sequenza.

Trascorso tale tempo prefissato, l'host si rimette in attesa di altri choke packet. Se ne arrivano altri, riduce ancora il flusso. Altrimenti, aumenta di nuovo il flusso.

Hop-by-hop choke packet
L'unico problema della tecnica precedente è la lentezza di reazione, perché l'host che produce i pacchetti ci mette un certo tempo a ricevere i choke packet ed a diminuire di conseguenza il ritmo della trasmissione. Per migliorare le cose si può costringere ogni router sul percorso, appena riceve tali pacchetti, a rallentare subito il ritmo. In tal caso si parla di ***hop-by-hop choke packet***.

Questa tecnica rende molto più veloce il sollievo del router che ha per primo i problemi di congestione, ma richiede più spazio di buffer nei router sul percorso dall'host originario a quel router.

5.5) Internetworking

Come sappiamo, esistono diverse architetture di rete, ciascuna caratterizzata dalle scelte effettuate in molti settori fra i quali ricordiamo:

- i servizi offerti dai vari livelli (connection oriented o no, reliable o no, ecc.);
- le modalità di indirizzamento;
- la dimensione massima dei pacchetti.

Per connettere fra loro reti eterogenee si devono superare problemi non banali, tra i quali:

- difformità nei servizi offerti (ad esempio, un servizio connected viene offerto solo su una delle reti);
- difformità nei formati dei pacchetti e degli indirizzi;
- difformità, nelle subnet, dei meccanismi di controllo dell'errore e della congestione;
- difformità nella dimensione massima dei pacchetti.
-

A causa di tali problemi, in generale (a meno che le architetture di rete non siano molto simili) non si usa questo approccio generale, ma altre tecniche più specifiche.

Tali tecniche sono basate sull'uso di particolari attrezzature, che operano a livelli diversi:

- i bridge, che abbiamo già visto e che operano a livello data link;
- i ***router multiprotocollo***: sono dei router in grado di gestire contemporaneamente più pile di protocolli.

Reti di router multiprotocollo

Mediante ricorso a tali apparecchiature diventa possibile, per esempio, impostare una internetwork costituita di reti eterogenee. Ogni rete può comunicare con le altre reti a lei conformi attraverso una porzione di rete costituita di router multiprotocollo.

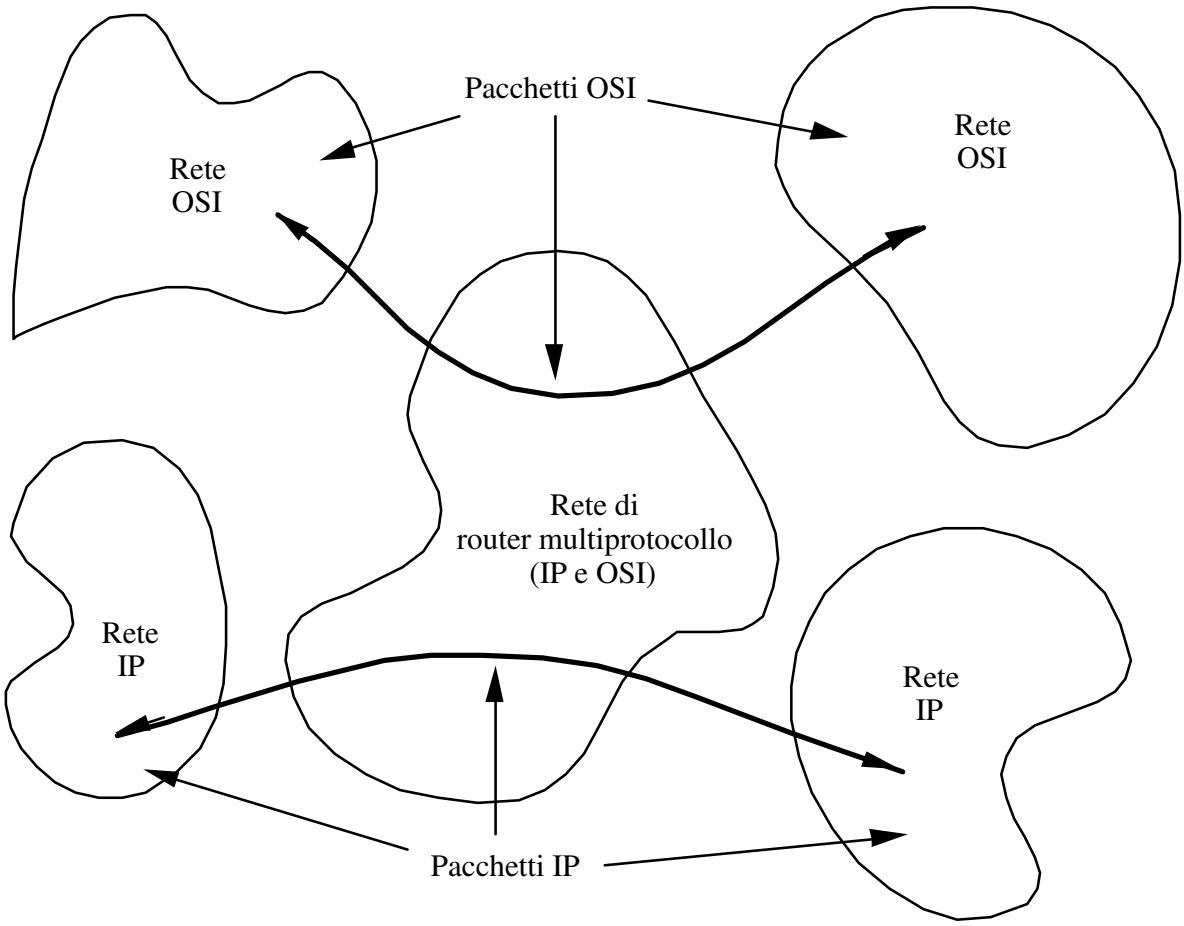

Figura 5-5: Internetwork basata su router multiprotocollo

Le reti OSI possono parlare fra loro, e così quelle IP. Nella subnet costituita dai router multiprotocollo circolano pacchetti di entrambe le architetture, che vengono instradati secondo le regole di competenza dell'architettura di sui fanno parte.

Tunneling

Un' altra tecnica che risolve un problema analogo è il ***tunneling***, che si utilizza per mettere in comunicazione due reti uguali per mezzo di una rete diversa.

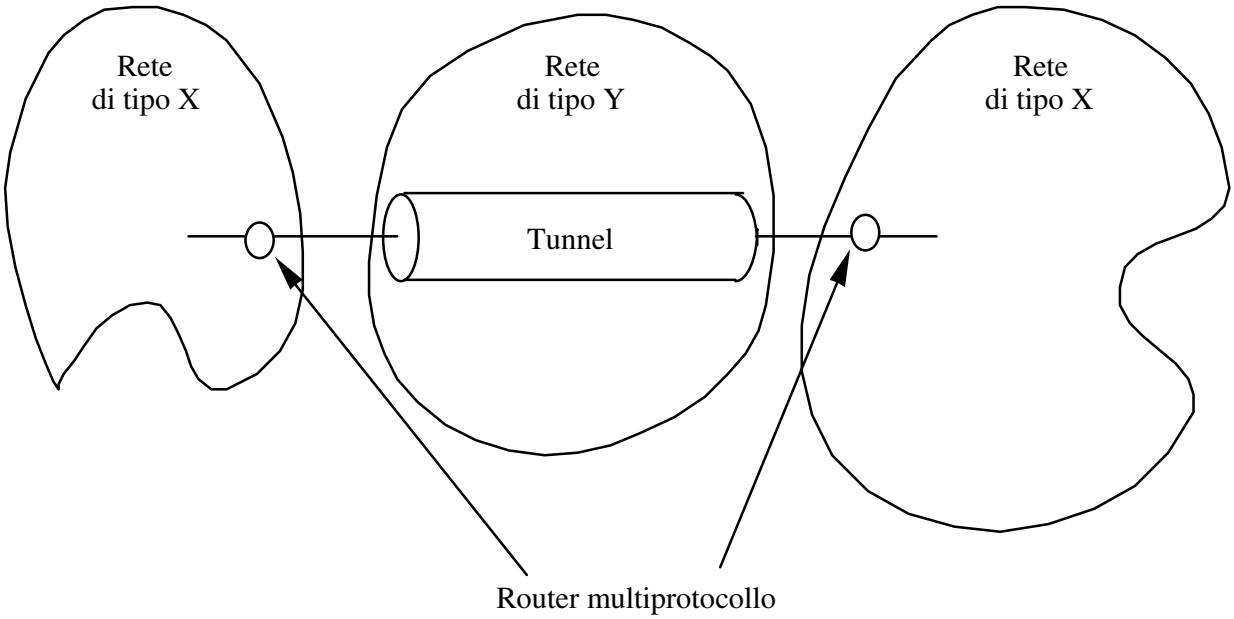

Figura 5-6: Tunneling

In questo caso la rete di tipo Y non è dotata di router multiprotocollo. Invece, un router designato in ciascuna delle due reti di tipo X è multiprotocollo e incapsula i pacchetti delle reti di tipo X dentro pacchetti di tipo Y, consegnandoli poi alla rete di tipo Y. Si noti che in questi pacchetti ci sono due buste di livello network:

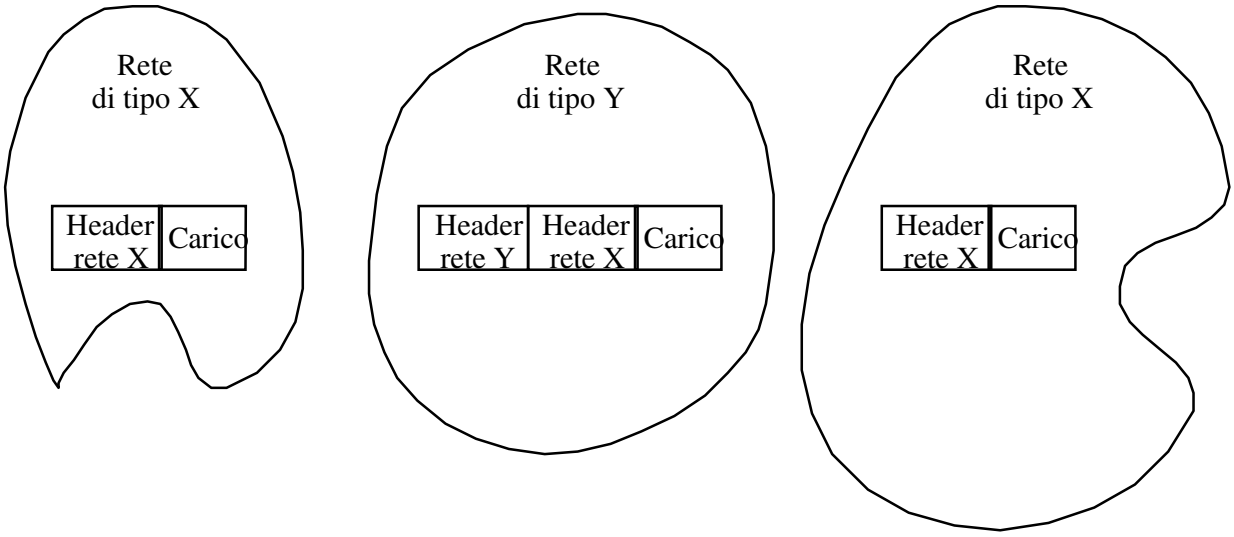

Figura 5-7: Doppio livello network nel tunneling

Frammentazione
Un diverso problema che talvolta si presenta è legato al fatto che in generale è possibile che i pacchetti in arrivo da una rete siano troppo grandi per transitare su un altra.

In questo caso è necessario che il livello network della rete di origine (e di quella di destinazione) prevedano meccanismi di ***spezzettamento*** del pacchetto in ***frammenti*** prima di consegnarli alla rete di transito, e di ***ricomposizione*** dei frammenti appena essi giungono dalla rete di transito in quella di destinazione (come vedremo, il protocollo IP è fornito di questa funzionalità).

Circuiti virtuali concatenati
Se tutte le reti interessate offrono servizi connessi nel livello network, è possibile costruire un circuito virtuale che si estende attraverso più reti eterogenee come ***concatenazione di circuiti virtuali*** che attraversano ciascuno una delle reti. Ai confini fra ogni rete ci sono dei router multiprotocollo che:

- creano la porzione di circuito virtuale che attraversa la rete di competenza, arrivando fino ad un router multiprotocollo situato all'altra estremità della rete;
- instradano successivamente i pacchetti lungo tale circuito virtuale.

Transport gateway
Un meccanismo simile è piuttosto diffuso a livello transport, dato che esso offre servizi connessi in quasi tutte le architetture. In tale ambito le apparecchiature interessate prendono il nome di ***transport gateway***.

Poiché i transport gateway operano a livello transport, essi ricevono i dati dal livello application di partenza e li consegnano al livello application di arrivo.

Application gateways
Un ultimo tipo di attrezzatura è costituito dagli ***application gateway***, che effettuano una conversione di dati a livello application.

Ad esempio, per mandare da un host Internet un messaggio di posta elettronica a un utente OSI:

- si compone il messaggio secondo il formato Internet; l'indirizzo del destinatario è un indirizzo OSI, ma viene espresso secondo regole valide per Internet;
- si invia il messaggio, che viene consegnato a un ***mail gateway***, cioè a un server di posta elettronica dotato della capacità di conversione dei formati;

il mail gateway estrae il testo dal messaggio, lo inserisce in un nuovo messaggio costruito secondo il formato OSI (che, per la posta elettronica, è definito dallo standard X.400) e lo invia sulla rete OSI (consegnandolo al pertinente livello trasporto).

5.5.1) Internetwork routing

In generale, in una internetwork le singole reti componenti sono entità autonome e vengono chiamate ***AS*** (***Autonomous System***).

In questo caso, il routing complessivo è a due livelli:

- un primo livello è costituito dall'***Interior Gateway Protocol*** (***IGP***). Questo termine identifica l' algoritmo di routing usato da un AS al proprio interno. Naturalmente, diversi AS possono utilizzare diversi IGP. Come abbiamo già visto, un IGP può anche essere gerarchico, in particolare quando le dimensioni dell'AS sono considerevoli;
- un secondo livello è dato dall'***Exterior Gateway Protocol*** (***EGP***), che è l'algoritmo che si usa per gestire il routing fra diversi AS. Tipicamente, in questo scenario EGP è l'algoritmo di routing che riguarda i router multiprotocollo. L'aspetto più interessante relativo a EGP è che esso deve spesso tener conto di specifiche leggi nazionali (ad esempio, divieto di far transitare dati sul suolo di una nazione ostile), per cui le decisioni di routing devono adattarsi a tali direttive.

5.6) Il livello network in Internet

Internet è una collezione di AS connessi gli uni con gli altri. Non esiste una struttura rigida, ma comunque si possono distinguere alcune componenti:

- backbone principali (linee ad alta velocità);
- reti regionali (USA);
- reti nazionali (Europa e resto del mondo);
- reti locali.

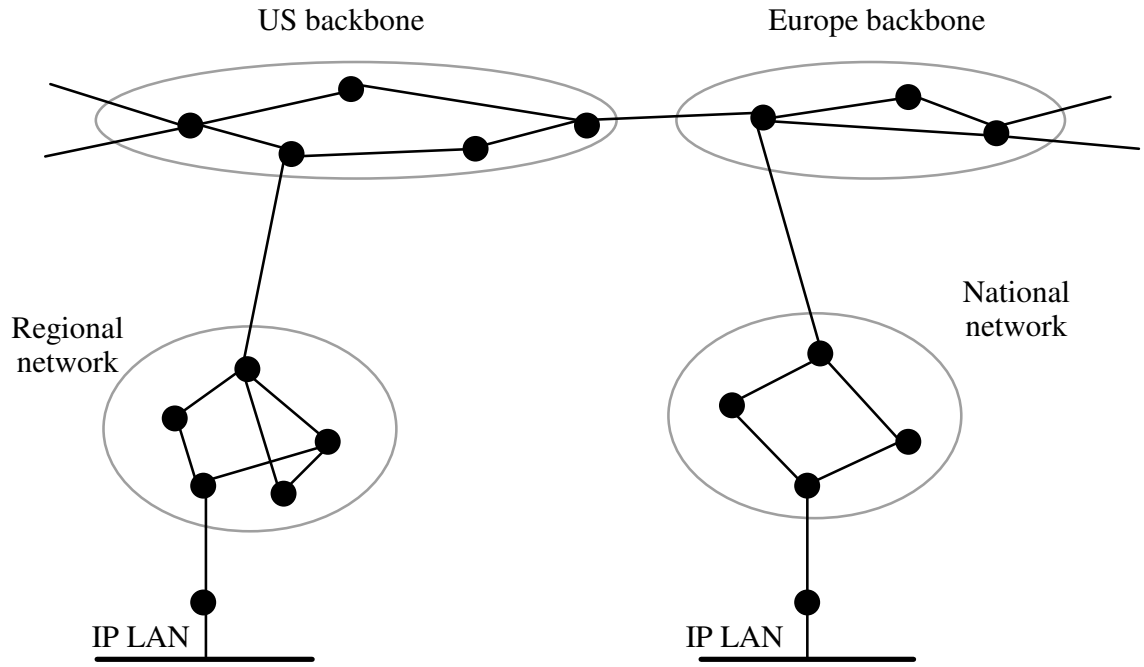

Figura 5-8: Organizzazione della rete Internet

Ciò che tiene tutto insieme è il protocollo di livello network dell'architettura TCP/IP, e cioè ***IP*** (***Internet Protocol***, RFC 791).

IP è un protocollo datagram, quindi non connesso e non affidabile, che opera come segue:

- riceve i dati dal livello transport e li incapsula in pacchetti di dimensione massima pari a 64 Kbyte (normalmente circa 1.500 byte);
- instrada i pacchetti sulla subnet, eventualmente frammentandoli lungo il viaggio;
- a destinazione:
    - riassembla (se necessario) i frammenti in pacchetti;
    - estrae da questi i dati del livello transport;
    - consegna al livello transport i dati nell'ordine in cui sono arrivati (che non è necessariamente quello in cui sono partiti).

5.6.1) Lo header IP (versione 4)

Un pacchetto IP è costituito da un ***header*** e da una parte dati.

L'header ha una parte fissa di 20 byte e una parte, opzionale, di lunghezza variabile.

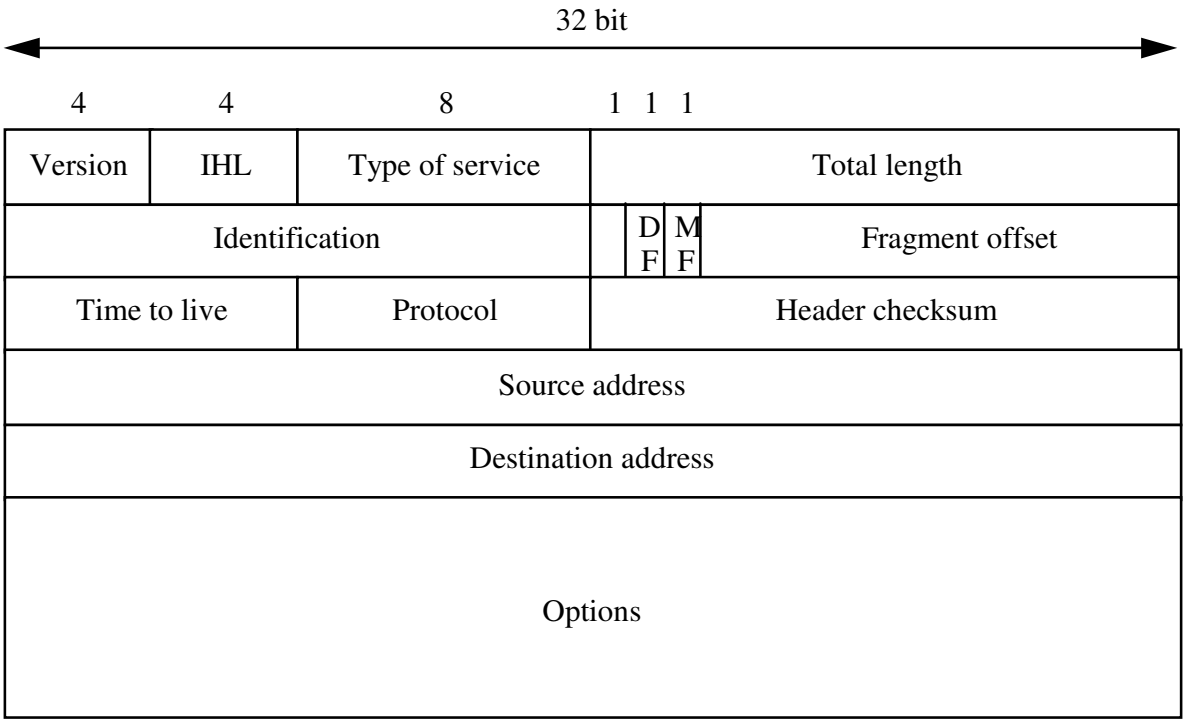

Figura 5-9: Formato dell'header IP

I campi dell'header hanno le seguenti funzioni:

Version	il numero di versione del protocollo (oggi è 4).
IHL	lunghezza dell'header in parole di 32 bit (minimo 5, massimo 15).
Type of service	caratterizza affidabilità e velocità richieste. E' di fatto ignorato dai router.
Total length	lunghezza del pacchetto (inclusi dati), massimo 65.535 byte.
Identification	tutti i frammenti di uno stesso pacchetto hanno lo stesso valore.
DF	***don't fragment*** (se uguale a 1, non si deve frammentare il pacchetto a costo di scegliere una strada meno veloce).
MF	***more fragments*** (se uguale a 1, il pacchetto non è ancora finito).
Fragment offset	indice del frammento nel pacchetto.
Time to live	contatore (inizializzato a 255) che viene decrementato di uno a ogni hop (o ad ogni secondo). Quando arriva a zero, il pacchetto viene scartato.
Protocol	codice del protocollo di livello transport a cui consegnare i dati (i codici sono definiti in RFC 1700).
Header checksum	checksum di controllo del solo header: • si sommano (in complemento ad uno) le parole a 16 bit dello header, considerando il checksum a zero; • si complementa ad uno il risultato; • viene ricalcolato ad ogni hop (time to live cambia).
Source e destination address	indirizzi di mittente e destinatario.
Options	opzioni, solo cinque sono definite oggi: • ***security***: quanto è segreto il pacchetto; • ***strict source routing***: cammino da seguire; • ***loose source routing***: lista di router da non mancare; • ***record route***: ogni router appende il suo indirizzo; • ***timestamp***: ogni router appende il suo indirizzo più un timestamp.

5.6.2) Indirizzi IP

Un indirizzo IP è formato da 32 bit e codifica due cose:

- ***network number***, cioé il numero assegnato alla rete IP (detta ***network***) su cui si trova l'elaboratore; in questo contesto una network è caratterizzata dal fatto di essere costituita da un unico canale di comunicazione cui sono connessi tutti gli host della network stessa (e quindi, ad esempio, una LAN oppure una linea punto punto fra due router);
- ***host number***, cioé il numero assegnato all'elaboratore.

La combinazione è unica: non possono esistere nell'intera rete Internet due indirizzi IP uguali.

Si noti che solitamente si ritiene che ogni host sulla rete abbia un singolo indirizzo IP. In realtà gli indirizzi sono assegnati alle interfacce di rete, quindi:
- se un host ha un'unica interfaccia di rete (come è il caso di un PC in LAN) allora ha un unico indirizzo IP;
- se un host ha X interfacce di rete (come è il caso di un router connesso ad una LAN e ad X-1 linee punto-punto) ha X indirizzi.

Gli indirizzi IP sono assegnati da autorità nazionali (***NIC***, ***Network Information Center***) coordinate a livello mondiale.

I formati possibili degli indirizzi sono i seguenti:

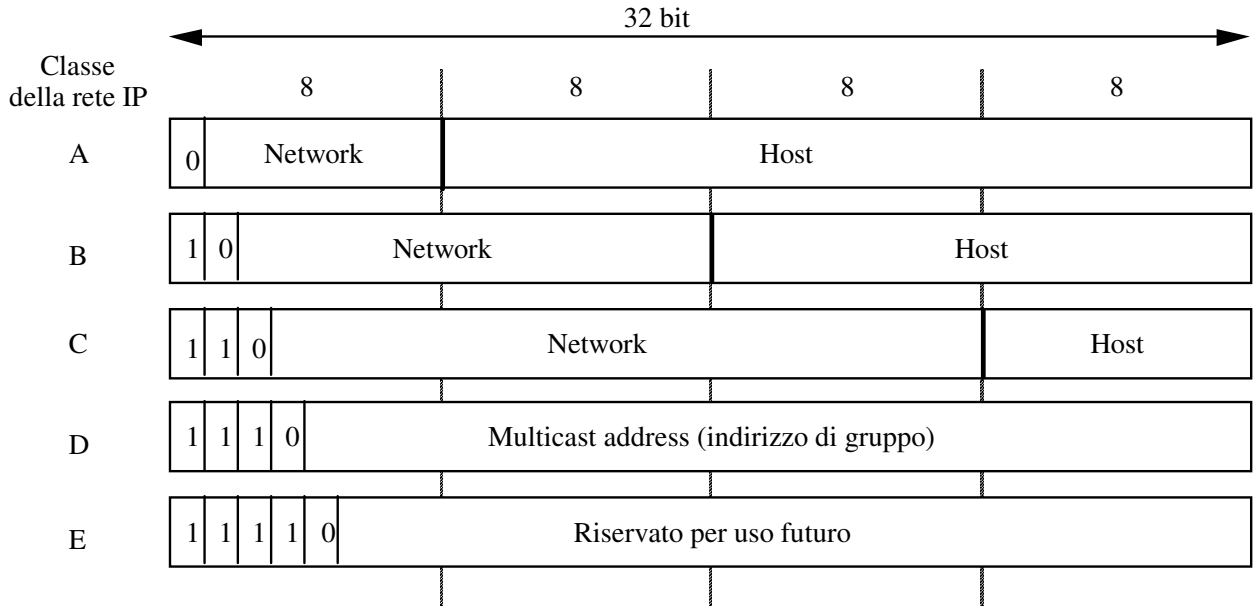

Figura 5-10: Formati degli indirizzi IP

Inoltre, esistono alcuni indirizzi con un significato speciale:

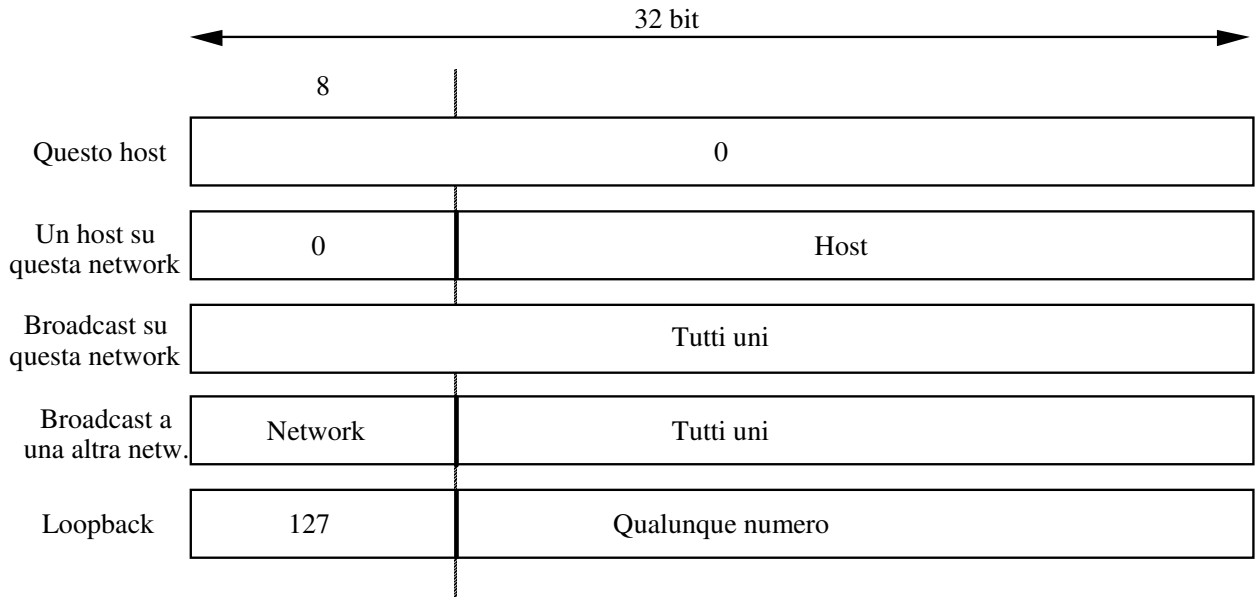

Figura 5-11: Indirizzi IP speciali

Quando si utilizza il ***loopback***, il pacchetto non viene inviato sulla rete ma viene elaborato come se fosse in arrivo: questo è molto utile, ad esempio, per effettuare localmente dei test su un software di rete in fase di sviluppo.

Ricapitolando, poiché alcune configurazioni binarie per gli indirizzi sono impegnate per gli indirizzi speciali, possono esistere:

- 126 network di classe A, le quali possono contenere 16 milioni di host ciascuna;
- 16382 network di classe B, con circa 64.000 host ciascuna;
- 2 milioni di network di classe C, con 254 host ciascuna.

Gli indirizzi sono usualmente espressi nella ***dotted decimal notation***, cioè i valori dei singoli byte sono espressi in decimale e sono separati da un punto, come nell'indirizzo:

```
141.192.140.37
```

In tale notazione, è possibile rappresentare separatamente il network number e l'host number. Per distinguerli, il primo è seguito da un punto. Ad esempio, nel caso dell'indirizzo IP precedente (che è relativo ad una network di tipo B), si ha:

- il network number è `141.192.` (notare il punto finale);
- l'host number è `140.37` (non c'è il punto finale).

5.6.3) Routing IP

Il collegamento fra due router non avviene direttamente, ma attraverso una network (in realtà di solito una ***subnet***, come vedremo poi) che li collega, e che di fatto è costituita dalle due interfacce di rete e dalla linea di comunicazione che le collega:

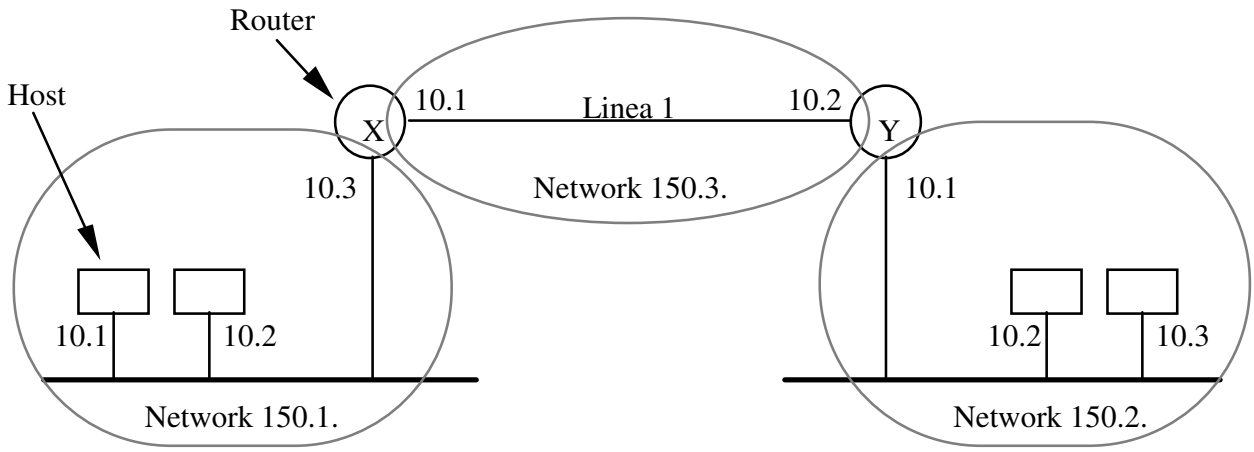

Figura 5-12: Collegamento di router IP

Ogni router possiede una tabella (costruita e mantenuta dinamicamente dall'algoritmo di routing in esercizio) che contiene elementi del tipo:

1. (this network number, host number) per ciascun host della network a cui il router è direttamente collegato;
2. (network number, 0) per ciascuna network lontana di cui il router conosce l'esistenza.

Associate a tali elementi ci sono le informazioni sull'interfaccia di rete da usare per instradare il pacchetto e, nel caso delle linee punto punto, l'indirizzo del router che si trova dall'altra parte.

Inoltre, viene mantenuto l'indirizzo di un ***default router*** (e la corrispondente linea seriale da usare per raggiungerlo) a cui inviare tutti i pacchetti destinati a network sconosciute.

Nell'esempio della figura precedente, il router X si comporta come segue:

- se arriva da fuori un pacchetto destinato a `150.1.10.1` il router, attraverso un elemento di tipo 1, scopre che deve inviarlo sulla LAN;
- se arriva dalla LAN un pacchetto per `150.2.10.3` e il router ha un elemento di tipo 2 quale (150.2., 0), allora invia il pacchetto (al router Y) sulla linea 1.

5.6.4) Subnet

Al fine di economizzare nel numero di network da usare (utilizzando al meglio quelle che possono contenere migliaia o milioni di host) una network può essere divisa in varie ***subnet***, ciascuna contenente i suoi host.

Questo è un fatto privato della network che viene suddivisa, e non ha bisogno di essere comunicato all'esterno.

Il meccanismo usato è di considerare, nell'indirizzo IP originario, l'host number come una coppia di valori: un ***subnet number*** e l'host number. Ciò avviene sulla base di una maschera di bit, detta ***subnet mask***, che deve essere unica per tutta la network e che delimita la parte di host number che viene usata come subnet number.

Ad esempio, per una rete di classe A si potrà avere:

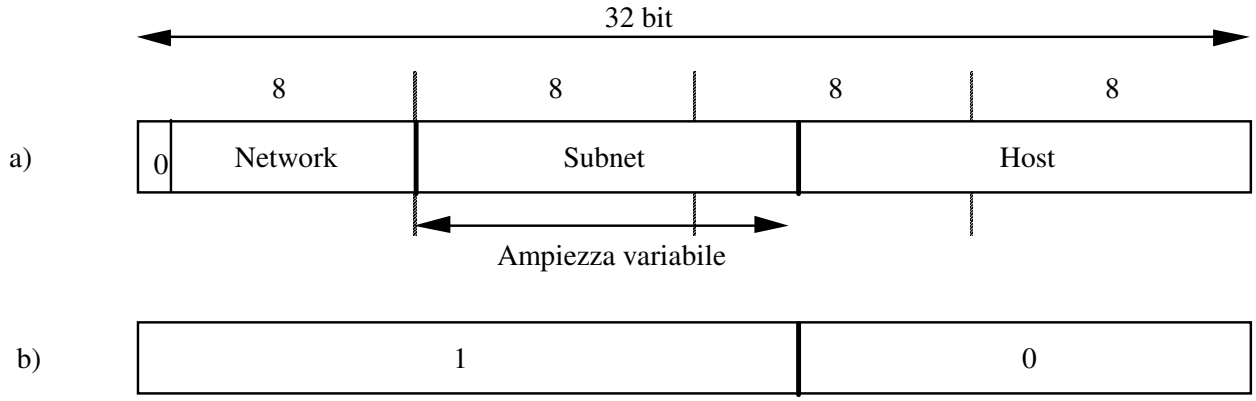

Figura 5-13: Indirizzo IP con subnet (a) e relativa subnet mask (b)

A seconda dell'ampiezza del campo dedicato alla subnet, si possono ottenere molte subnet contenenti ciascuna pochi host oppure poche subnet che però potranno contenere molti host.

Nei router si aggiungono elementi del tipo:
- (this network number, this subnet number, host number) per gli host delle subnet a cui il router è collegato direttamente;
- (this network number, subnet number, 0) per le altre subnet a cui il router non è collegato direttamente;

con le relative informazioni sulle interfacce di rete da usare.

La rete di figura 5-12, che nella sua formulazione originaria impegnava ben tre network di classe B, può essere realizzata con un sola network di classe B suddivisa in tre subnet. A tal fine, usiamo un network number uguale a 150.1. ed una subnet mask (espressa in dotted decimal notation) uguale a 255.255.255.0, cioé:

- i primi due byte dell'indirizzo identificano la network;
- il terzo byte identifica la subnet;
- il quarto byte identifica l'host.

Il risultato è il seguente:

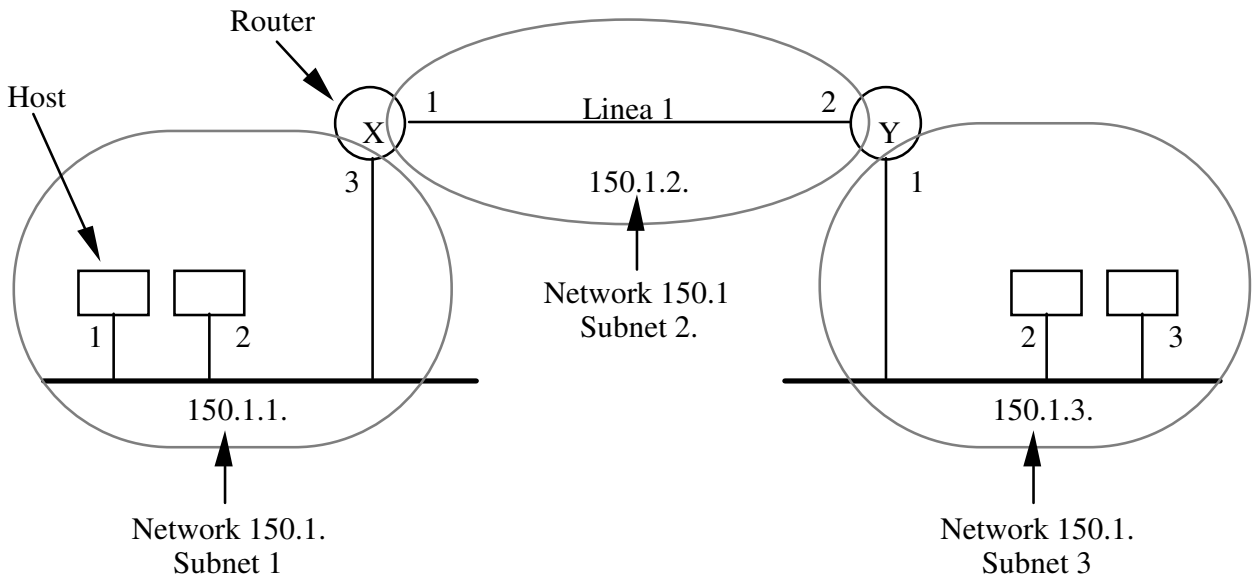

Figura 5-14: Collegamento di router IP mediante uso di subnet

5.6.5) Protocolli di controllo

Assieme a IP esistono diversi protocolli per il controllo del funzionamento della subnet.

ICMP (Internet Control Message Protocol, RFC 792)
L'operatività della subnet è controllata continuamente dai router, che si scambiano informazioni mediante messaggi conformi al protocollo *ICMP* (tali messaggi viaggiano dentro pacchetti IP).

Esistono molti tipi di messaggi, fra i quali:
- ***destination unreachable***: non si trova la destinazione del pacchetto. Viene inviato al mittente del pacchetto;
- ***time exceeded***: il contatore di un pacchetto è arrivato a zero. Viene inviato al mittente del pacchetto;
- ***redirect***: il router ha ragione di pensare che il pacchetto gli è arrivato per errore, ad esempio perché un host mobile si è spostato. Viene inviato al mittente del pacchetto;
- ***echo request, reply***: si vuole sapere se una destinazione è viva e raggiungibile. Si invia request, si aspetta reply;
- ***timestamp request, reply***: come il precedente, con in più la registrazione dell'istante di arrivo e partenza, per misurare le prestazioni della rete.

ARP (Address Resolution Protocol, RFC 826)
Il protocollo ***ARP*** serve per derivare, dall'indirizzo IP dell'host di destinazione, l'indirizzo di livello data link necessario per inviare il frame che incapsulerà il pacchetto destinato all'host di cui all'indirizzo IP.

Esso opera appoggiandosi direttamente sul livello data link e non su IP:
- viene inviata a tutte le stazioni della LAN, in data link broadcast, una richiesta del tipo: "chi ha l'indirizzo IP uguale a 151.100.17.102 ?"
- solo l'host che ha quell'indirizzo IP risponde, inserendo nella risposta il proprio indirizzo data link;
- quando riceve la risposta, l'host la mantiene in memoria per circa 15 minuti.

Se l'indirizzo IP è relativo ad un'altra network:
- la soluzione più semplice è mandare il pacchetto ARP come prima, configurando però il router in modo che risponda alle richieste ARP relative ad altre reti fornendo il proprio indirizzo ethernet(***proxy ARP***); il router farà poi da tramite nella conversazione IP fra il mittente e il destinatario, inviando di volta in volta all'uno i pacchetti IP che gli giungono dall'altro;
- alternativamente, si configura l'host impostando al suo interno l'indirizzo ethernet di default (quello del router) a cui mandare i pacchetti IP per le altre reti; anche in questo caso il router deve fare da tramite nella conversazione IP fra il mittente e il destinatario.

RARP (Reverse Address Resolution Protocol, RFC 903)
Il protocollo ***RARP*** risolve il problema inverso, cioè consente di trovare quale indirizzo IP corrisponda a un determinato indirizzo data link.

Esso è utile nel caso di stazioni senza disco, che al momento dell'avvio caricano l'immagine del codice binario del sistema operativo da un server.

Il vantaggio che si ottiene è che una sola immagine binaria va bene per tutte le stazioni: ogni stazione personalizza poi l'immagine binaria con la determinazione del proprio indirizzo IP mediante una richiesta RARP, nella quale invia il proprio indirizzo data link (che è cablato nell'interfaccia di rete).

5.6.6) Protocolli di routing

Come già detto, Internet è una collezione di AS connessi da backbone ad alta velocità.

Ciò che caratterizza un AS è il fatto di essere gestito da una singola autorità.

Esempi tipici di AS sono:
- la rete degli enti di ricerca di una nazione (rete GARR in Italia);
- la rete di una singola azienda.

Il routing complessivo è organizzato in modo gerarchico:
- all'interno di un singolo AS si usa un solo Interior Gateway Protocol (IGP);
- viceversa, per il routing fra gli AS si usa un Exterior Gateway Protocol (EGP).

Interior Gateway Protocol per Internet
Il protocollo originario era il ***RIP*** (***Routing Information Protocol***, RFC 1058) di tipo distance vector, ormai sostituito da ***OSPF*** (***Open Shortest Path First***, RFC 1247), che è di tipo link state.

OSPF consente fra l'altro un routing gerarchico all'interno dell'AS, che viene suddivisa in diverse aree:
- ***backbone area*** (che è connessa a tutte le altre)
- altre aree

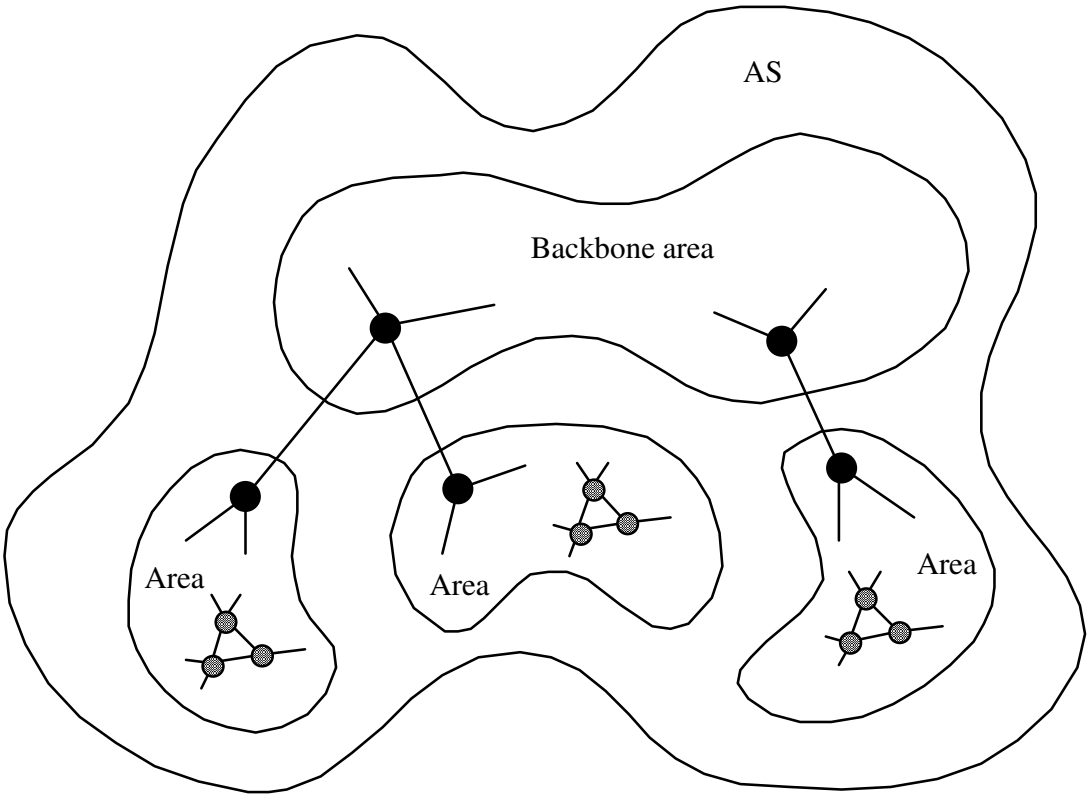

Figura 5-15: Suddivisione di un AS in aree

I router di un AS possono essere:

- ***interni a un area***: si occupano del routing all'interno dell'area;
- ***sul confine di un'area***: si occupano del routing fra le diverse aree via backbone;
- ***nell'area backbone***: si occupano del routing sul backbone;
- ***sul confine dell'AS***: si occupano del routing fra gli AS (applicando EGP); possono trovarsi anche sul confine di un'area.

Exterior Gateway Protocol per Internet

Il protocollo EGP, usato dai router sul confine dell'AS e da quelli sui backbone ad alta velocità che connettono gli AS, si chiama ***BGP*** (***Border Gateway Protocol***, RFC 1654).

E' fondamentalmente di tipo distance vector, con due novità:

- possiede la capacità di gestire politiche di instradamento (derivanti, ad esempio, da leggi nazionali) che vengono configurate manualmente nei router;
- mantiene (e scambia con gli altri router) non solo il costo per raggiungere le altre destinazioni, ma anche il cammino completo. Ciò consente di risolvere il problema del

count to infinity, perché se una linea va giù il router può subito scartare, senza quindi poi distribuirli, tutti i cammini che ci passano.

5.6.7) IP versione 6

Dopo un lungo lavoro, IETF ha approvato il successore di IP versione 4, cioè la versione 6 (***IPv6***, RFC 1883 - 1887).

I requisiti principali di progetto erano:
- aumentare il numero di indirizzi, ormai quasi esauriti;
- ottenere una maggiore efficienza nei router (tavole più piccole, routing più veloce);
- supportare meglio il traffico real time;
- offrire maggiore sicurezza ai dati riservati.

Le principali differenze rispetto alla versione 4 sono:
- indirizzi di 16 byte, il che significa disporre di 2^{128} indirizzi IP, e cioé $7*10^{23}$ indirizzi IP per metro quadro su tutto il nostro pianeta;
- header semplificato: 7 campi contro 13;
- funzioni di autentificazione e privacy, basate su crittografia;
- gestione della qualità di servizio attraverso un campo ***flow label***, che consente di istituire delle pseudoconnessioni con caratteristiche negoziate in anticipo.