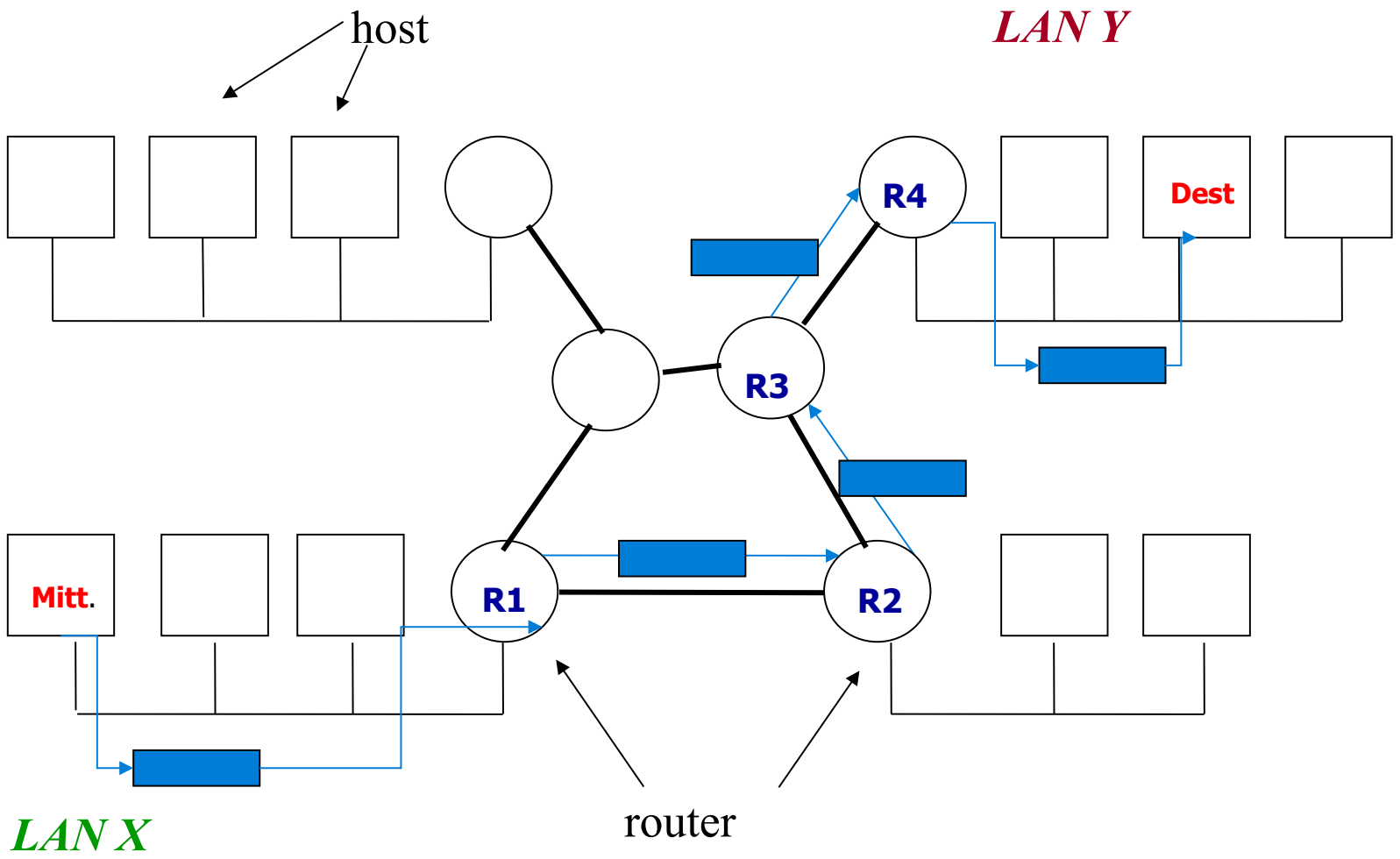
Livello Network: si occupa del trasferimento dei pacchetti dal mittente della rete ***X*** al destinatario della rete ***Y***, ottenuto con salti (***hop***) da un ***router*** all'altro attraverso le ***reti*** intermedie.



Principali funzioni del ***Livello Network***

Il livello network deve

* ***routing***: *scegliere di volta in volta il cammino migliore*;
* ***flow control*** e ***congestion control***: *gestire il flusso dei dati e le congestioni*;

__internetworking__: *prevedere la co-presenza di più reti diverse*.

Tali incombenze richiedono la conoscenza della topologia della rete.

In fase di progettazione e realizzazione del livello network occorre stabilire:

* i ***servizi offerti*** al livello transport;

* l'***organizzazione interna*** della subnet di comunicazione.

Due scuole di pensiero:

* fautori dei ***servizi connection-oriented*** (compagnie telefoniche);

* fautori dei ***servizi connectionless*** (Internet Community).

Connection-oriented (circuiti virtuali): *viene stabilito un circuito virtuale fra sorgente e destinazione*

L'ID di tale circuito viene trasmesso con il pacchetto, e tutti i router, lungo il cammino, inoltrano il pacchetto in base ad esso.

Connectionless: *i router di volta in volta stabiliscono (in base all'indirizzo del nodo destinatario) su quale linea far proseguire la trasmissione*.

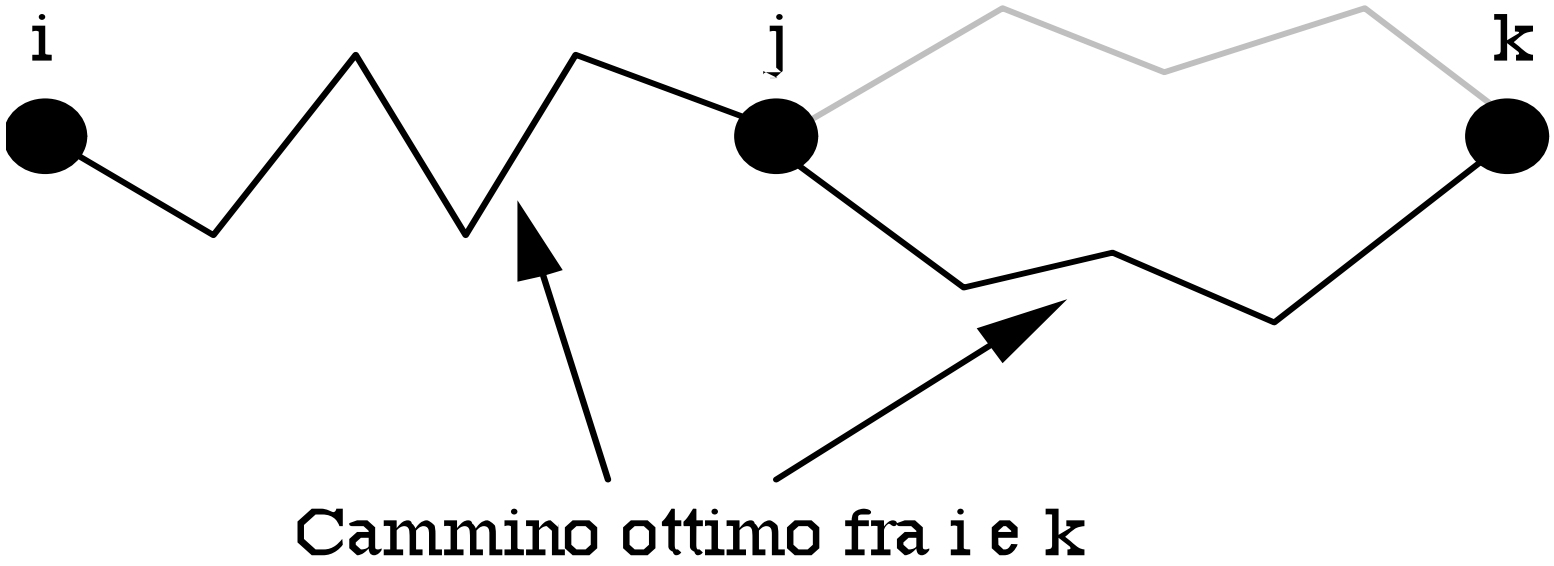
La scelta si basa su apposite *tabelle di instradamento* (*routing table*) in cui, per ogni possibile destinazione, viene indicata la linea d'uscita.

Mentre nelle subnet ***connection-oriented*** l'***algoritmo di routing*** viene applicato solo nella fase di setup (***session routing***), in quelle ***connectionless*** viene applicato ex novo per ogni pacchetto.

Un algoritmo di routing deve essere:

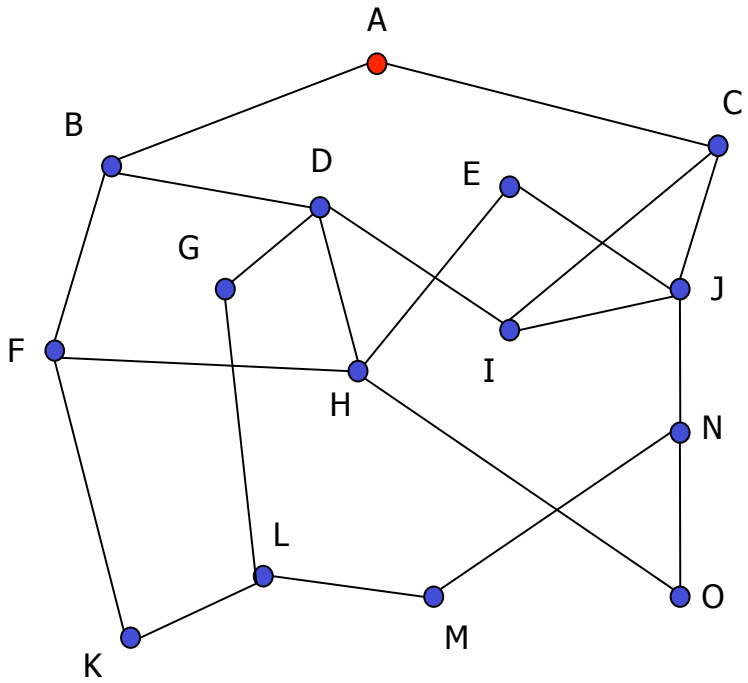
* di ***facile implementazione***;
* ***robusto*** (deve essere in grado di operare anche in caso di cadute di linee e/o router);
* ***convergente*** (e possibilmente in fretta);
* ***ottimale*** (scegliere il percorso ottimo).

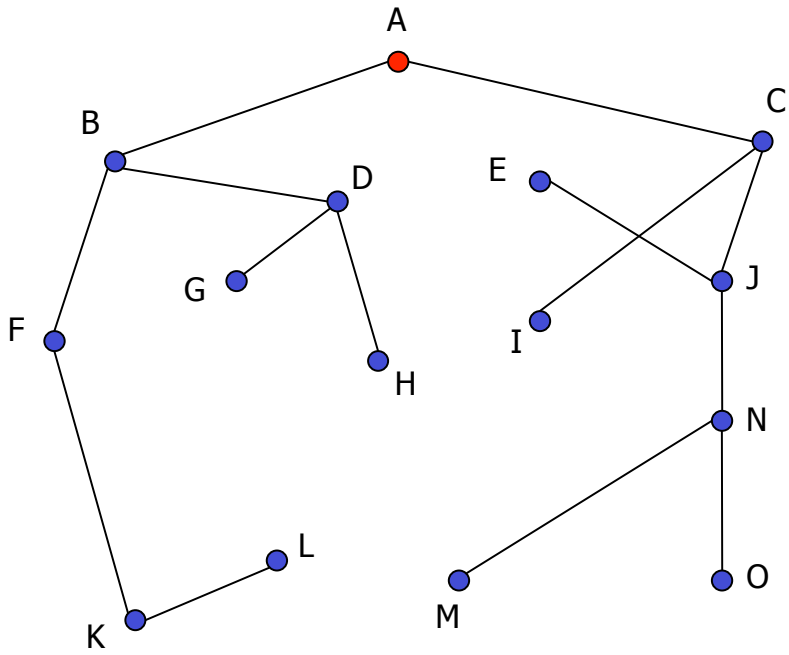
Principio di ottimalità: *se il router j è nel cammino ottimo fra i e k, allora anche il cammino ottimo fra j e k è sulla stessa strada*.



Se così non fosse, ci sarebbe un cammino fra j e k migliore di quello che comprende il cammino ottimo fra i e k, ma allora ci sarebbe anche un cammino fra i e k migliore di quello ottimo.

Il ***Sink tree*** per un router ***A*** è l'***albero*** costituito dall'insieme dei cammini ottimi da tutti i router ad ***A***.





Gli algoritmi di routing individuando i sink tree relativi a tutti i possibili router destinazione, ed instradando esclusivamente attraverso i cammini previsti dal sink-tree, instradano i pacchetti sempre per il cammino ottimo.

Algoritmi di routing

Static routing (*algoritmi non adattivi*):

i router instradano i pacchetti secondo il circuiti virtuali ad essi inizialmente associati;

Dynamic routing (*algoritmi adattivi*):

ogni router sceglie la porta di uscita in base all'indirizzo del destinatario, al traffico, alla topologia della rete, ecc.

Shortest path routing (*routing statico*): calcola il minimo cammino tra una coppia di router

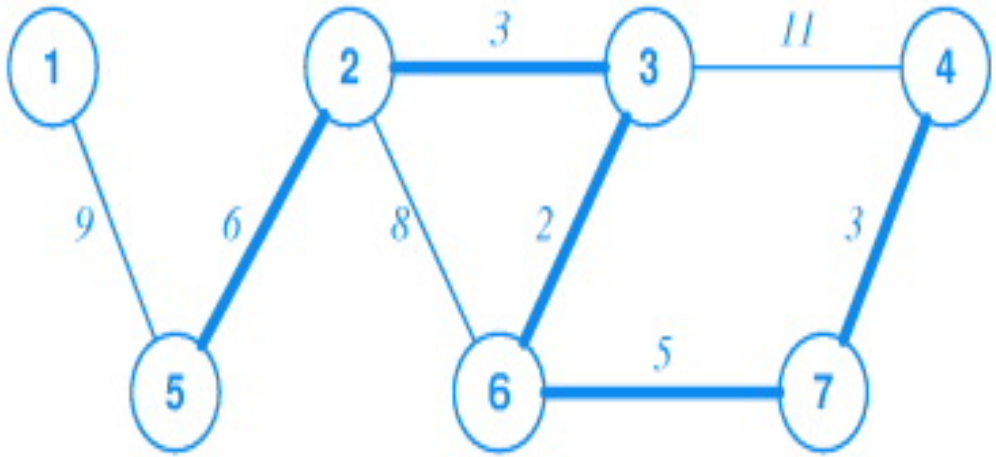
All'avvio della rete, o quando ci sono variazioni alla topologia, l'host impiegato per la gestione della rete:

** costruisce (o ricostruisce) un grafo della subnet (individuando i router e linee punto-punto tra essi)*

** applica un algoritmo (ad es. Dijkstra) che calcola il cammino minimo fra ogni coppia di nodi.*
** invia tali informazioni a tutti i router.*

Il cammino minimo cambia in funzione delle grandezze che si desidera minimizzare (numero di hop, lunghezza dei collegamenti, …).

Algoritmo di Dijkstra (1959): lavora su grafi orientati, che hanno pesi non negativi sui collegamenti. Ogni nodo del grafo rappresenta un router ed ogni arco una linea di comunicazione (canale). Tra tutti i possibili cammini tra il router mittente ed il router destinatario l'algoritmo sceglie quello con peso minimo.



Possibili metriche:

* distanza geografica
* costi
* capacità

Flooding: *Usato come algoritmo di routing (statico), genera un numero enorme (teoricamente infinito) di pacchetti.*

Alcune tecniche per limitare il traffico generato:

* *in ogni pacchetto si inserisce 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*;

* *in ogni pacchetto si inserisce la coppia (****source router ID, sequence number****). Un router prende nota ed accetta il pacchetto solo la prima volta, scartando arrivi succesvi*

* ***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).*

Flow-based routing (routing statico a ritardo minimo): *sceglie il cammino che minimizza il ritardo medio*.

Calcolando in anticipo il traffico atteso su ogni linea, stima il ritardo medio atteso per ciascuna linea.

Per applicare l'algoritmo è necessario conoscere:

* *la topologia della rete*;

* *la matrice del traffico **T(i,j)** (l'elemento **t(i,j)** indica il traffico stimato fra il router **i** ed il router **j**)*;

* *la matrice delle capacità **B(i,j)**, espresse in bps (l'elemento **b(i,j)** indica la capacità della linea point to point che collega il router **i** al router **j**)*.

Il ***Flow-based routing*** calcola il ***ritardo medio dell'intera rete*** effettuando la 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).

Fissato un percorso tra mittente e destinatario:

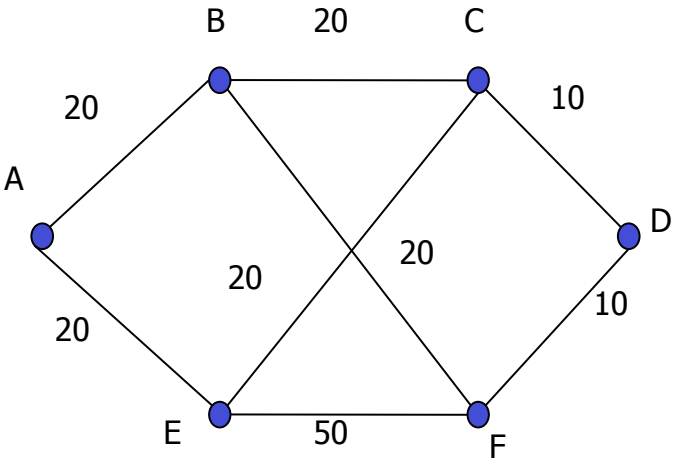
** si calcola il traffico che incide su ogni linea (dato alla somma di tutti i **t(i,j)** instradati su quella linea);*

** si calcola il ritardo di ogni linea;*

** si calcola il ritardo medio dell'intero percorso;*

** si ripete il procedimento per tutti i possibili percorsi, scegliendo alla fine quello che ha il minimo ritardo medio*

Routing basato su flusso



Devono essere noti:

* Topologia della rete
* La matrice di traffico
* La matrice delle capacità dei canali

	A	B	C	D	E	F
A		9 AB	4 ABC	1 ABFD	7 AE	4 AEF
B	9 BA		8 BC	3 BFD	2 BFE	4 BF
C	4 CBA	8 CB		3 CD	3 CE	2 CEF
D	1 DFBA	3 DFB	3 DC		3 DCE	4 DF
E	7 EA	2 EFB	3 EC	3 ECD		5 EF
F	4 FEA	4 FB	2 FEC	4 FD	5 FE	

La Formula di Little ci dice che :

$$\text{Ritardo medio} = \frac{1}{\text{Pacchetti/ s - Traffico}}$$

Routing basato su flusso

i	Linea	Traffico (p/s)	Capacità(kbps)	Pacchetti/s	Ritardo(ms)	Peso
1	AB	14	20	25	91	0.17
2	BC	12	20	25	77	0.14
3	CD	6	10	12.5	154	0.07
4	AE	11	20	25	71	0.13
5	EF	13	50	62.5	20	0.15
6	FD	8	10	12.5	222	0.09
7	BF	10	20	25	67	0.12
8	EC	8	20	25	59	0.09

Dimensione media del pacchetto 800 bit e Traffico totale 82 pacchetti/s

Routing basato su flusso

Con riferimento alla tabella precedente, se consideriamo una dimensione media del pacchetto di 800 bit ed un traffico totale di 82 pacchetti/s distribuiti come in tabella, per la linea AB, avendo ipotizzato una capacità di 20 Kbps si ha:

Pacchetti/s=20 Kbps /800b=20000/800=25

Ritardo(ms)= 1/(25 – 14)p/s = 0.091 s = 91 ms

Peso =14/82 = 0.17

Distance vector routing (*routing dinamico basato sulla distanza*)

*Il generico router registra nella propria tabella (**vector**) di routing la distanza (numero di hop, ritardo, ecc.) che lo separa da ogni altro router e, per ogni uno di essi, la linea in uscita da usare per arrivarci.*

*La stima della distanza viene fatta misurando il tempo di risposta a speciali **pacchetti ECHO**, che il generico router invia ai router ad esso fisicamente connessi.*

Nel ***Distance Vector Routing*** *ogni router invia* ***periodicamente*** *la propria tabella a tutti i vicini, e riceve quelle dei vicini.*

In base a tali tabelle, per ogni destinazione,
vengono ricalcolati e registrati nella propria tabella (*vector*) i migliori percorsi

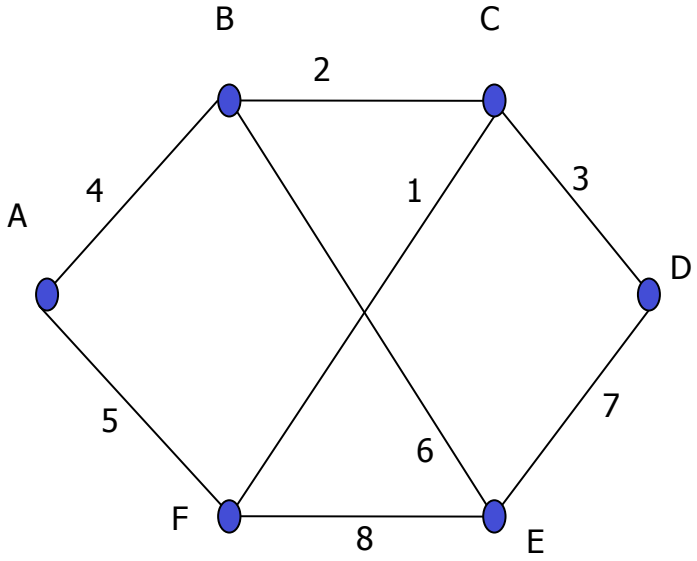
se stesso --> router intermedi --> router destinazione

Per valutare il singolo percorso il generico router rileva dal proprio *vector* le distanze con i router confinanti, e dalle tabelle ricevute quelle tra i vicini ed i router remoti.

Distance Vector Routing

Questo tipo routing richiede la gestione di una tabella contenente la più piccola distanza conosciuta per ogni destinazione, e quale canale utilizzare per raggiungerla.

Queste tabelle sono aggiornate scambiando informazioni con i propri vicini.



Distanza A-B=4; A-F=5;

B

A	4
B	0
C	2
D	5
E	6
F	3

F

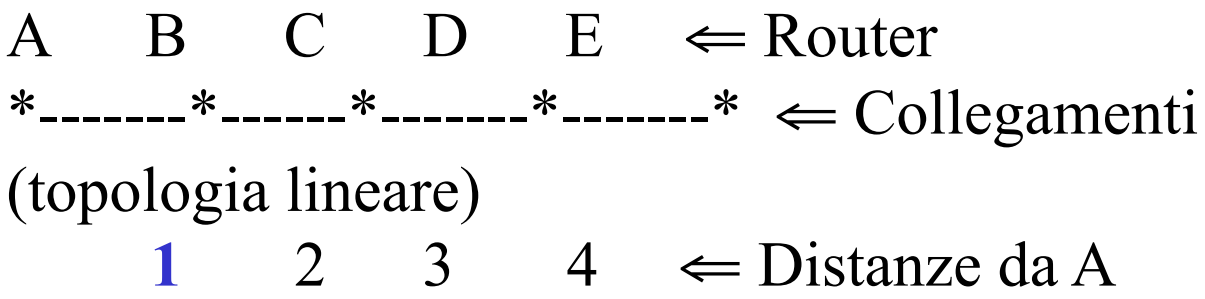
A	5
B	3
C	1
D	4
E	8
F	0

→

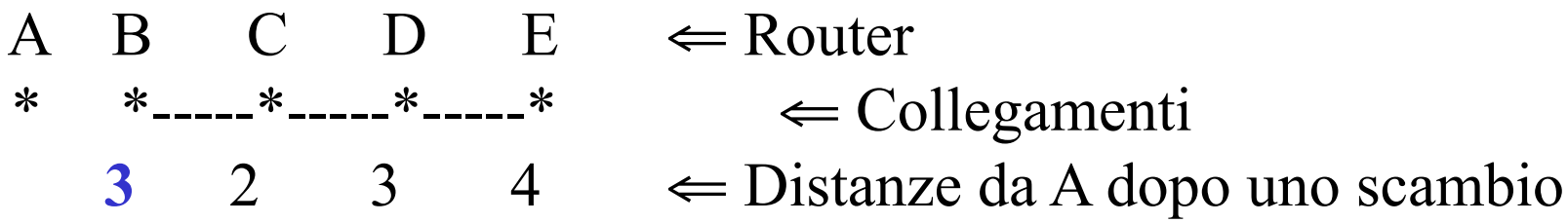
A

A	0	-
B	4	B
C	6	B
D	9	F
E	10	B
F	5	F

L'algoritmo distance vector routing è lento a reagire quando un collegamento va giù.

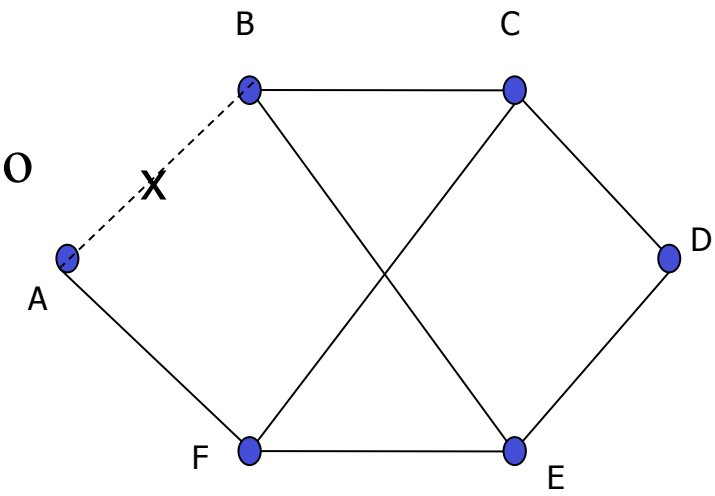


Se cade la linea fra A e B, dopo uno scambio:



Non ricevendo risposta da **A**, **B** crede di poterci arrivare via **C**, che ha distanza due da **A**. **B** passa quindi la propria distanza da A a **3** (**1** da **B** a **C** + **2** da **C** a **A**).

A	B	C	D	E	⇐ Router
1	2	3	4		Distanze da A dopo
3	2	3	4		⇐ uno scambio
3	4	3	4		⇐ due scambi
5	4	5	4		⇐ tre scambi
5	6	5	6		⇐ quattro scambi



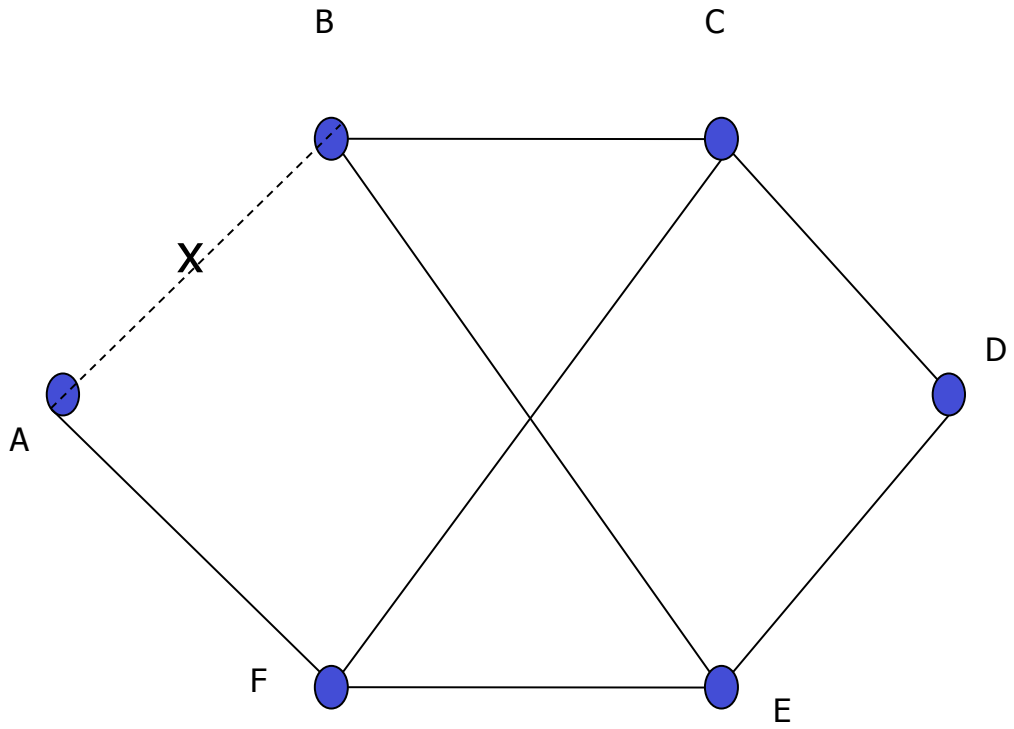
Count-to-infinity: a lungo andare, tutti i router vedono lentamente aumentare la distanza da A.

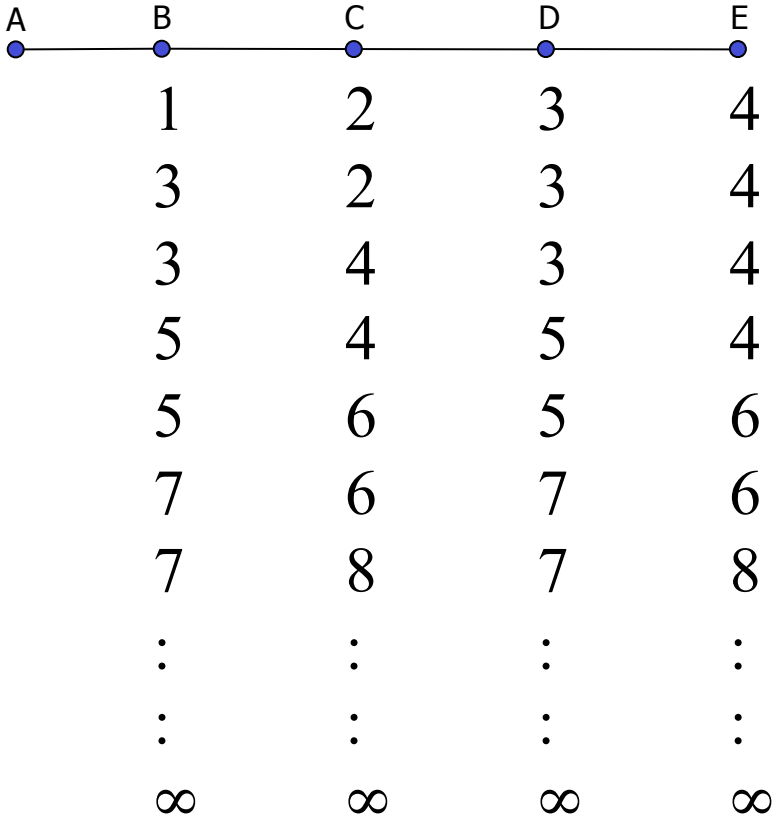
Se la distanza rappresenta il numero di hop si può porre come limite il diametro della rete; nel caso invece rappresentasse il tempo, i cammini che a causa della congestione dovessero presentare un forte ritardo, verrebbero erroneamente considerati interrotti.

RIP (***Routing Internet Protocol***) è il ***distance vector routing***, impiegato fino al 1979 da ARPANET

Problemi del ***distance vector routing***

- Il problema del conteggio all'infinito:





- Non tiene conto delle capacità delle linee

- Converge con tempi molto lunghi

Link state routing (*routing dinamico basato sullo stato dei collegamenti*)

Ogni router controlla lo stato dei collegamenti (misurando il ritardo di ogni linea) con i suoi vicini immediati, e distribuisce tali informazioni a tutti gli altri.

Ciascun router, sulla base delle informazioni ricevute, costruisce localmente la topologia completa dell'intera rete e calcola il cammino minimo fra se e tutti gli altri router.

In avvio il router:

* **invia** un ***pacchetto HELLO*** su tutte le linee in uscita;
* **riceve** in risposta gli indirizzi dei propri vicini.

Successivamente ***invia vari pacchetti di ECHO***, ***misura il tempo di arrivo della risposta*** e, mediando su pacchetti di varia lunghezza, ***calcola il ritardo della linea.***

Quindi 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:

* ***ad intervalli regolari***;
* ***quando accade un evento significativo*** (es.: una linea va giù o torna su).

Errori nella distribuzione dei pacchetti possono indurre qualche router in errore sulla effettiva topologia, con conseguenti malfunzionamenti.

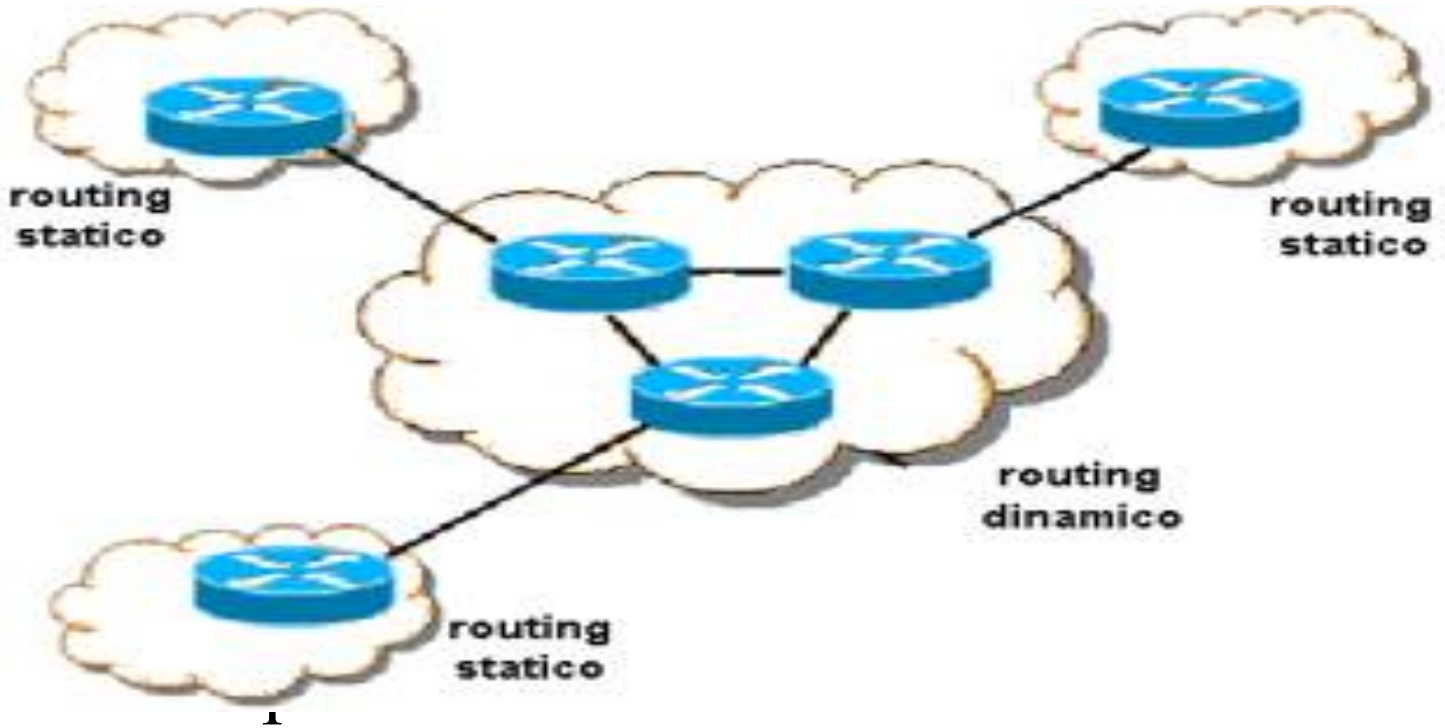
Per la distribuzione si usa il ***flooding***. Le coppie (source router ID, sequence number) inserite nei pacchetti consentono in ricezione di eliminare i duplicati. In ogni pacchetto viene inserito anche un età, che viene decrementata via via nel tempo. I router non propagano i pacchetti con il valore nullo nel campo età.

Il link state routing venne attivato nel 1979 su ARPANET.

1. Scopre i propri vicini e i loro indirizzi di rete spedendo uno speciale pacchetto di HELLO su ogni linea
2. Misura il ritardo o il costo per ognuno dei suoi vicini con speciali pacchetti ECHO
3. Costruisce un pacchetto contenente tutto quello che ha appena scoperto, più l'identità del router ed un numero di sequenza a 32 bit
4. Via flooding spedisce questo pacchetto a tutti i router
5. Calcola il cammino minimo per ogni altro router utilizzando l'algoritmo di Dijkstra

OSPF (***Open Shortest Path First***): basato sul link state routing è uno degli algoritmi più utilizzati in Internet.

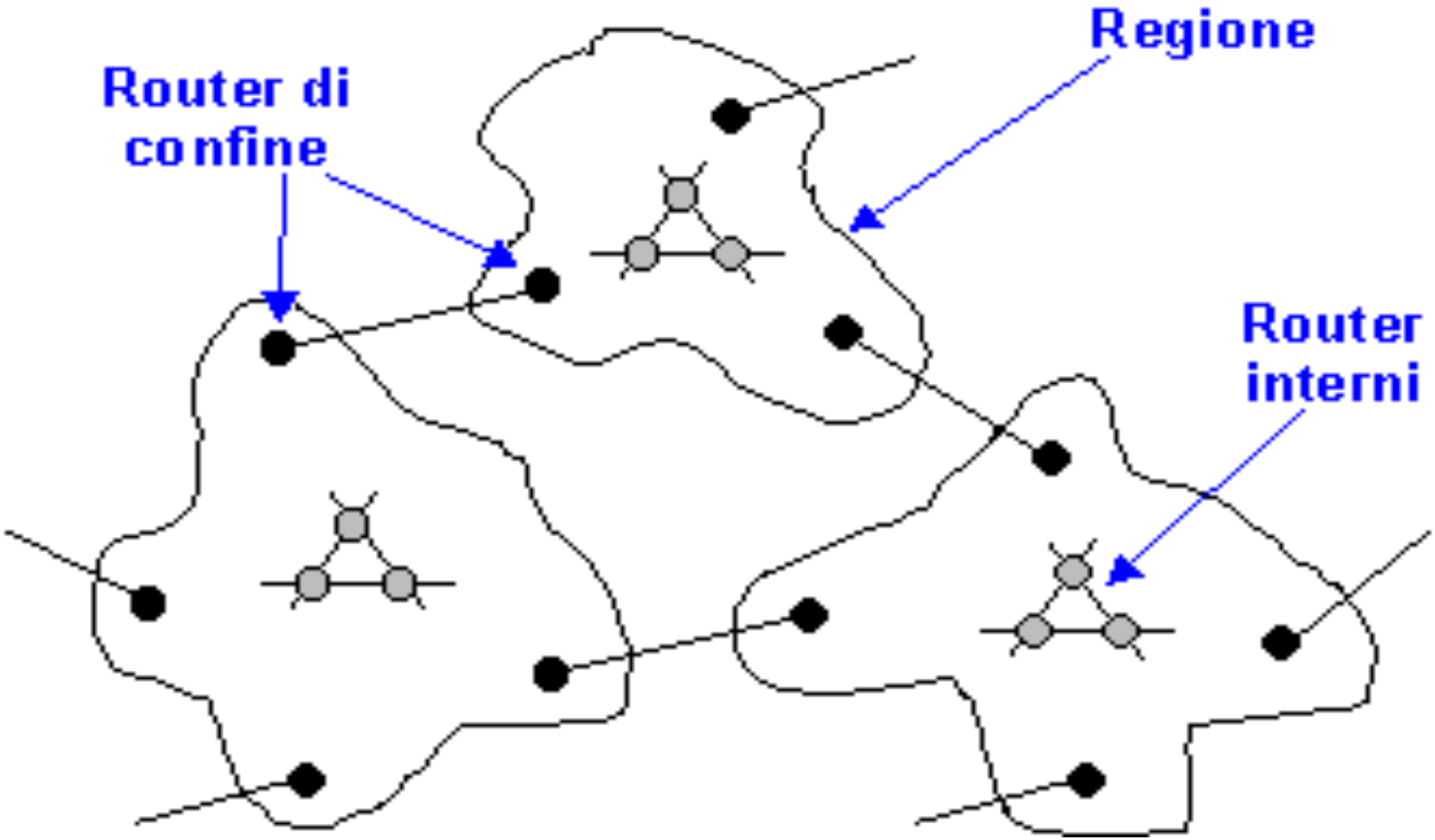
Routing statico: preferibile in reti di dimensioni ridotta.



Routing di

possibili frequenti cambi di topologia (dovuti a guasti di sistemi e/o percorsi), variazioni di prestazioni in dipendenza del carico di lavoro di router o linee, etc.

Reti divise in zone (regioni): per reti con decine di migliaia di nodi, diventa troppo gravoso mantenere in ogni router la completa topologia.



Reti divise in regioni

connessioni interne alla stessa regione: i router (***router interni***) adottano gli algoritmi visti;

connessioni tra router di regioni diverse: la comunicazione avviene attraverso i ***router di confine***.

È compito del ***router di confine*** stabilire a quale altro ***router di confine*** (di diversa regione) inviare i dati.

Se due livelli non sono sufficienti, l'organizzazione si ripete su più livelli.

Tabelle per il routing gerarchico

Router interni:

* una riga per ogni altro router interno, con la relativa linea da usare per arrivarci;

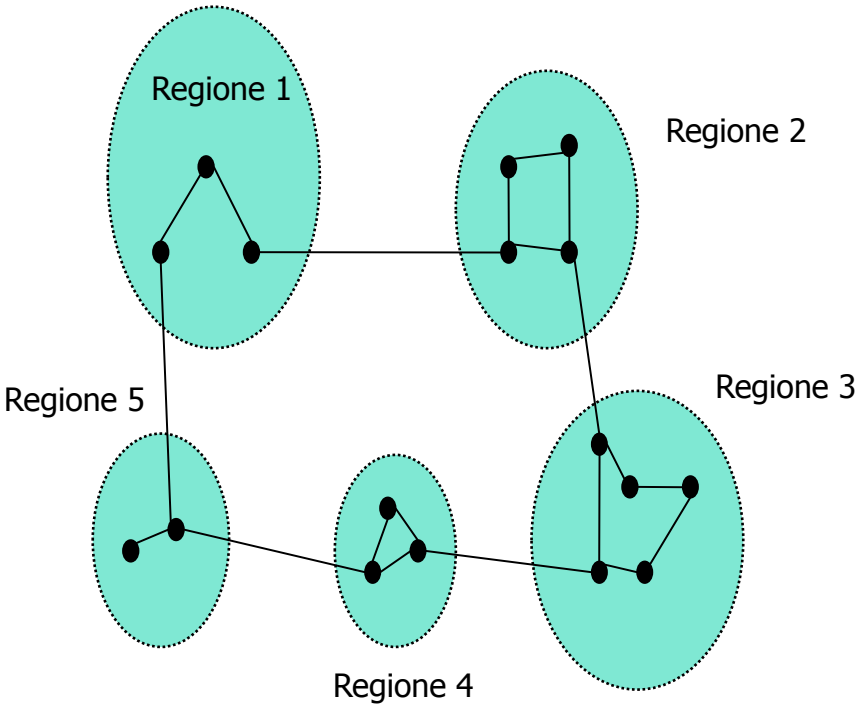
* una riga per ogni regione, con l'indicazione del relativo router di confine e della linea da usare per arrivarci.

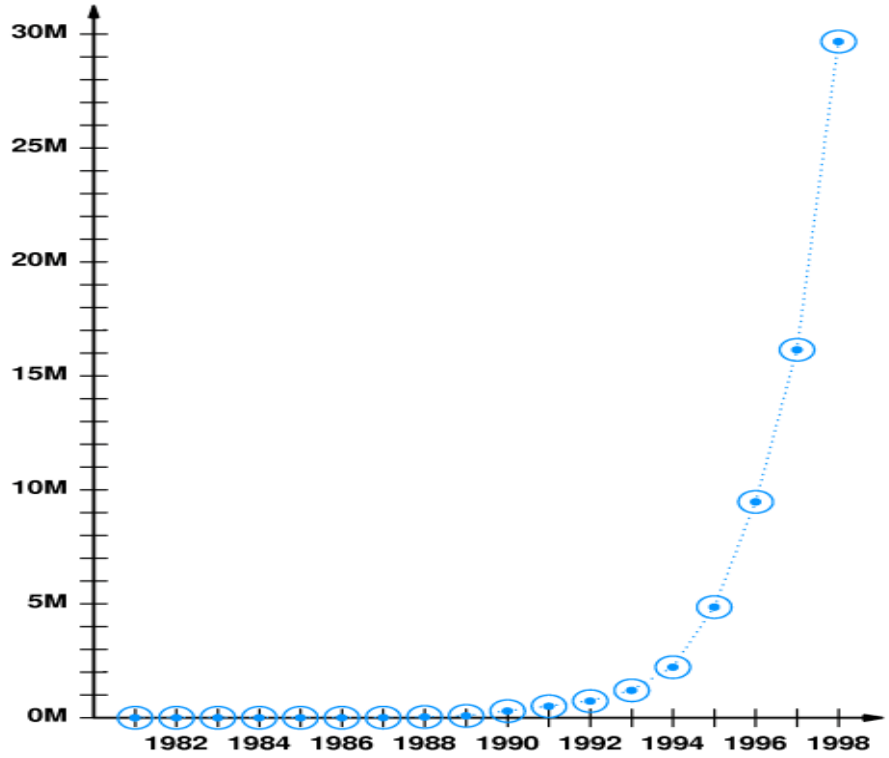
Router di confine:

* una riga per ogni regione, con l'indicazione del router di confine da contattare e della linea da usare per arrivarci.

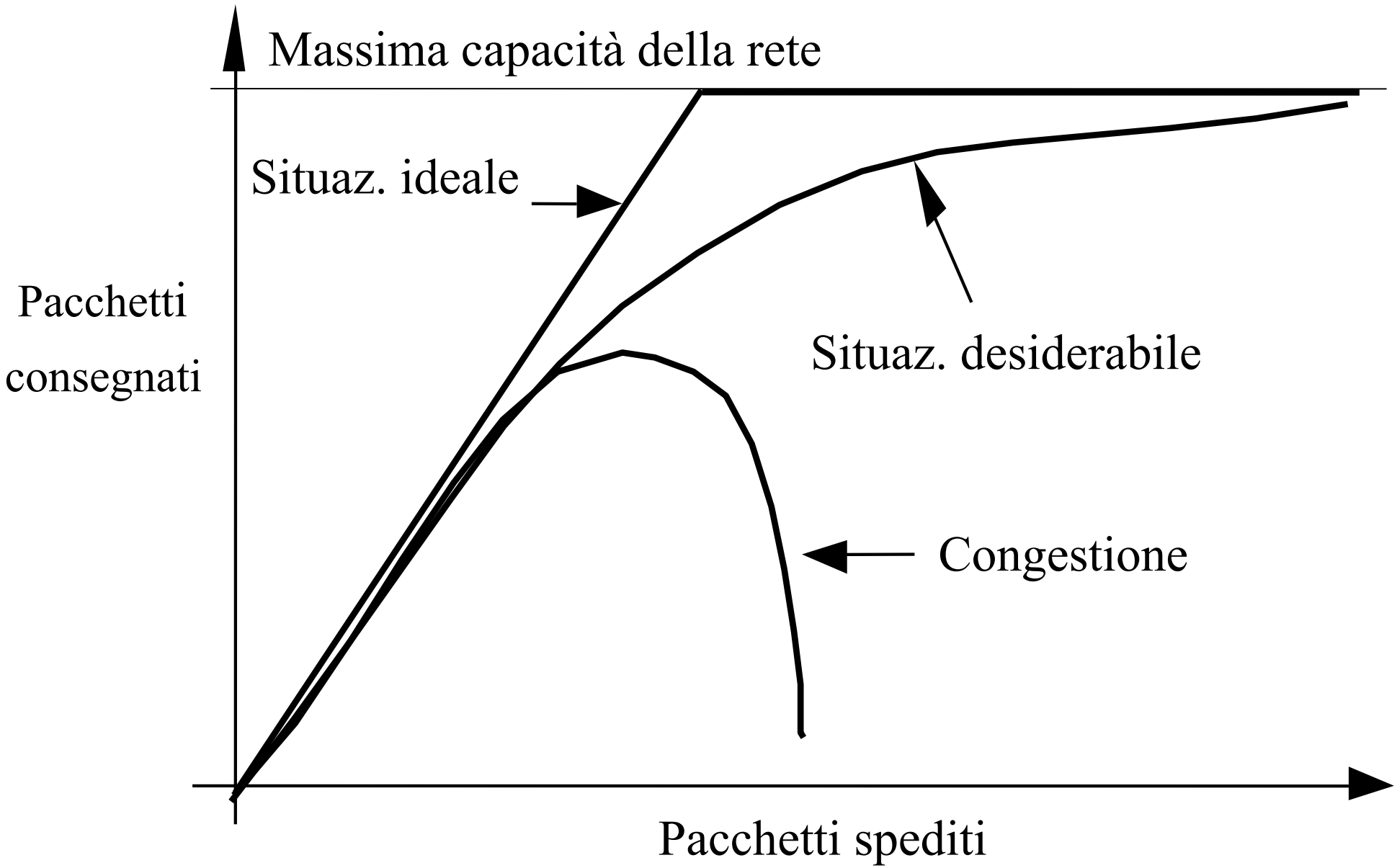
Routing gerarchico

La crescita esponenziale di Internet, avrebbe richiesto tabelle di routing sempre più grandi. Per ovviare a ciò si dividono i router in più gruppi (regioni); ogni router conosce solo i dettagli della propria regione e come comunicare con le altre, ma non conosce la loro struttura interna.





Congestione: degrado delle prestazioni dovuto alla presenza di troppi pacchetti in una parte della subnet.



Fattori che possono determinare la congestione in un router:

* troppi ***pochi buffer*** nel router;

* ***processore del router troppo lento***;

* ***linea di trasmissione troppo lenta*** (si allunga la coda nel router di partenza).

La congestione in un router tende a propagarsi ai suoi vicini. Poiché un router non conferma i pacchetti che scarta, i router mittenti non potendo rimuoverli dai propri buffer, iniziano a diventare loro stessi congestionati.

Il ***controllo della congestione*** è un problema globale di tutta la rete

Approccio ***open loop*** (*senza controreazione*):
si fissano parametri per la rete (*velocità di trasmissione, ampiezza dei buffer,* ..) in modo che la congestione non si verifichi, ma poi non effettua azioni correttive

Approccio ***closed loop*** (*con controreazione*): controlla la situazione della rete, intraprendendo le azioni opportune quando necessario.

Traffic shaping (***open loop***): per limitare la possibilità di congestione, la trasmissione dei pacchetti è forzata ad un ritmo regolare.

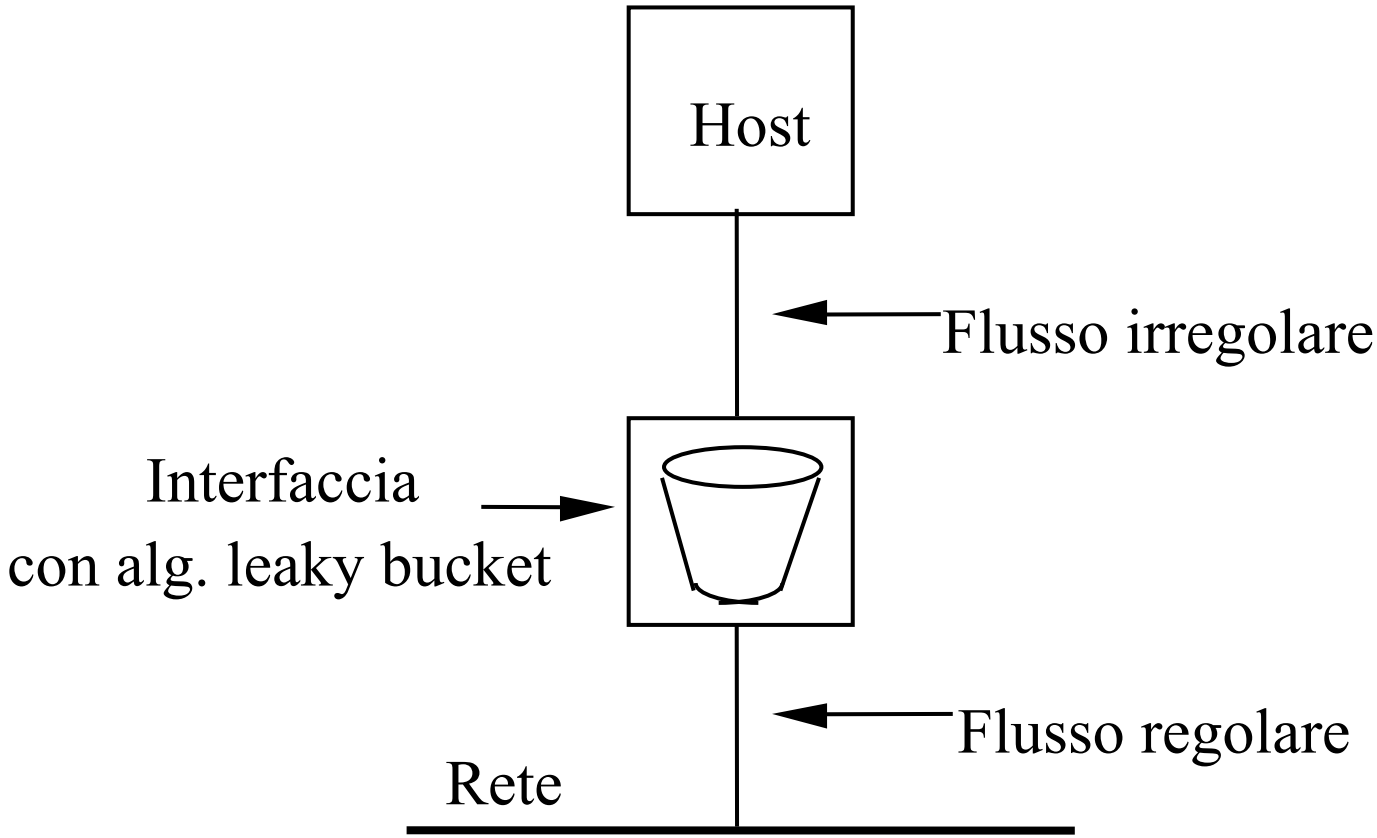
Tre tecniche per implementare il traffic shaping:

* ***leaky bucket***;

* ***token bucket***;

* ***flow specification***.

***Algoritmo Leaky bucket* (*secchio che perde*)**: un secchio riempito da un rubinetto (di cui si può regolare l'apertura) 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.



Leaky bucket: il router riversa sulla rete pacchetti con un data rate fissato (diciamo ***b*** bps) accodando, nei propri buffer, quelli ricevuti ma non ancora trasmessi.

L'host perde tutti i pacchetti che non possono essere contenuti nei buffer.

Un traffico bursty, quindi, non crea problemi sulla rete finché il data rate medio non supera i ***b*** bps, oltre si cominciano a perdere pacchetti.

Algoritmo token bucket (*secchio di gettoni*): consente un grado di irregolarità controllato anche nel flusso che esce sulla rete.

In base a tale algoritmo un host:

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

* quando invece ha dati da trasmettere, sfruttando se necessario tutto il credito disponibile, trasmette alla massima velocità consentita dalla linea.

Flow specification

Sorgente, ***subnet*** e ***destinatario*** si accordano riguardo:

*le ***caratteristiche del traffico*** che si vuole inviare (data rate, grado di burstness, ecc.);

* la ***qualità del servizio*** (ritardo massimo, frazione di pacchetti che si può perdere, ecc.).

Questo accordo, preso prima di trasmettere, può essere fatto sia in subnet *connection-oriented* (circuito virtuale) che in subnet *connectionless*.

Flow specification

Mentre nelle subnet connesse l'accordo si riferisce ai circuiti virtuali, in quelle non connesse si riferisce alla sequenza di pacchetti che sarà trasmessa.

Admission control: nelle reti basate su circuiti virtuali, per evitare la congestione si nega (***admission control***) l'attivazione di nuovi circuiti virtuali ove non vi fossero sufficienti risorse per gestirli.

Choke packet (***closed loop***): *quando il grado di utilizzo di una linea in uscita si avvicina ad una soglia prefissata, il router con un choke packet (to choke: soffocare) invita l'Host d'origine a diminuire il flusso*.

Quando l'host sorgente riceve un ***choke packet*** diminuisce il flusso (lo dimezza), ma continua a trasmettere ignorando, per un tempo prefissato, i successivi choke packet (ne arrivano molti in sequenza).

Trascorso tale tempo, l'host si rimette in attesa di choke packet e, se ne arrivano, riduce ulteriormente il flusso di trasmissione.

Occorre un certo tempo perché l'host riceva i choke packet ed inizi a diminuire il ritmo della trasmissione.

Hop-by-hop choke packet:

quando i router sul percorso ricevono un ***choke packet***, rallentano immediatamente il ritmo di trasmissione, mantenendo parte dei pacchetti nei propri buffer (si occupa quindi più spazio di buffer nei router sul percorso dall'host originario a quel router congestionato).

Internetworking

Per connettere reti eterogenee si devono risolvere problemi non banali:

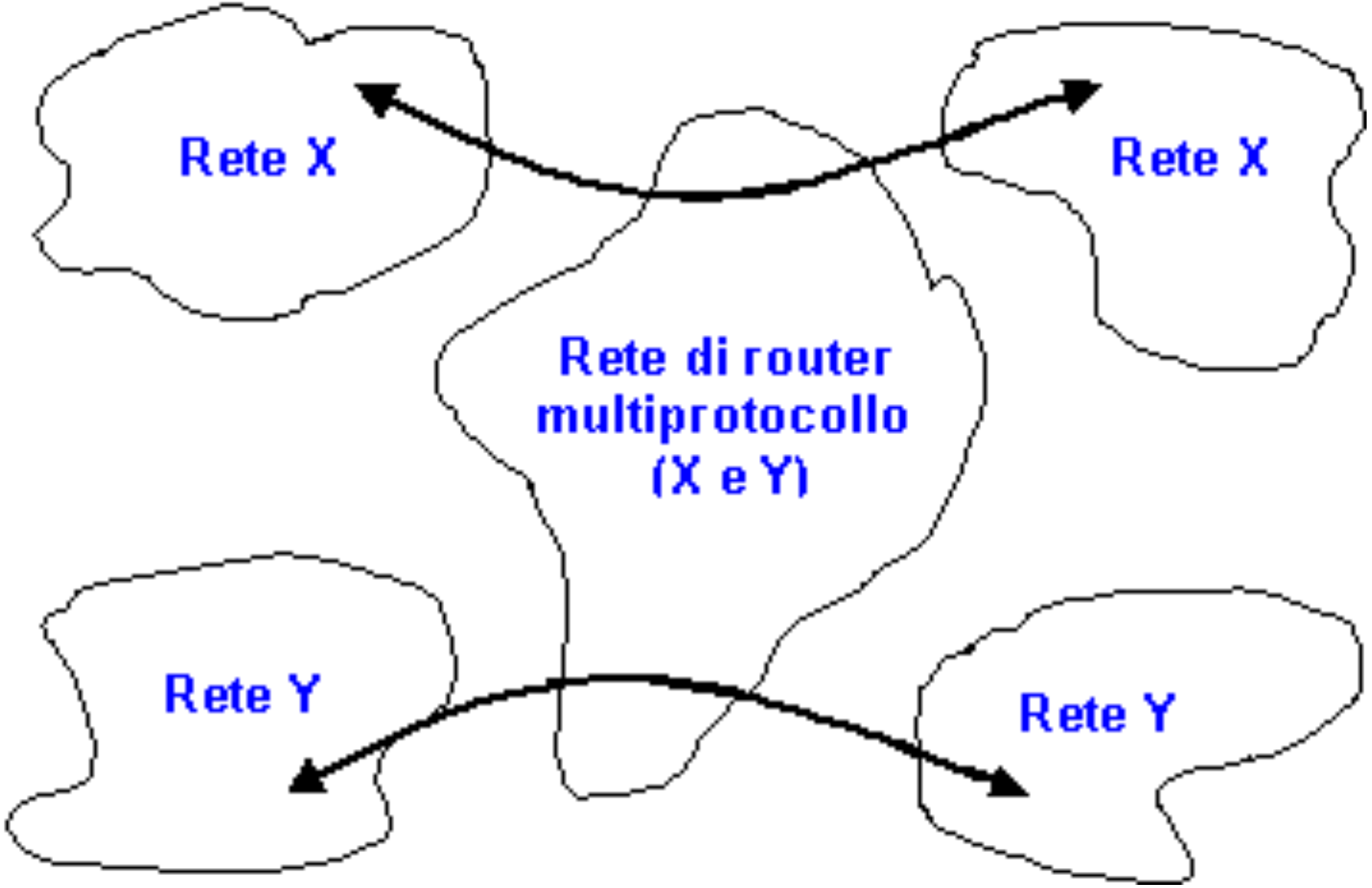
* diversi servizi offerti (servizio connected viene offerto solo su una delle reti);

* diversi formati dei pacchetti e degli indirizzi;

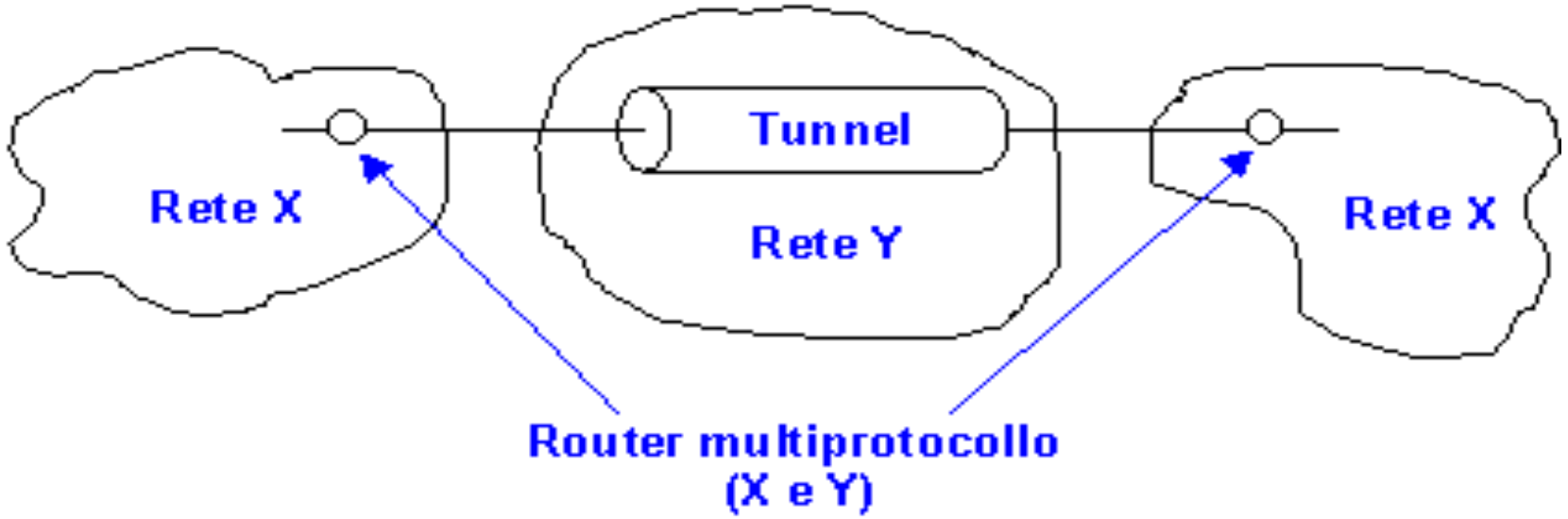
* diversi meccanismi di controllo dell'errore e della congestione;

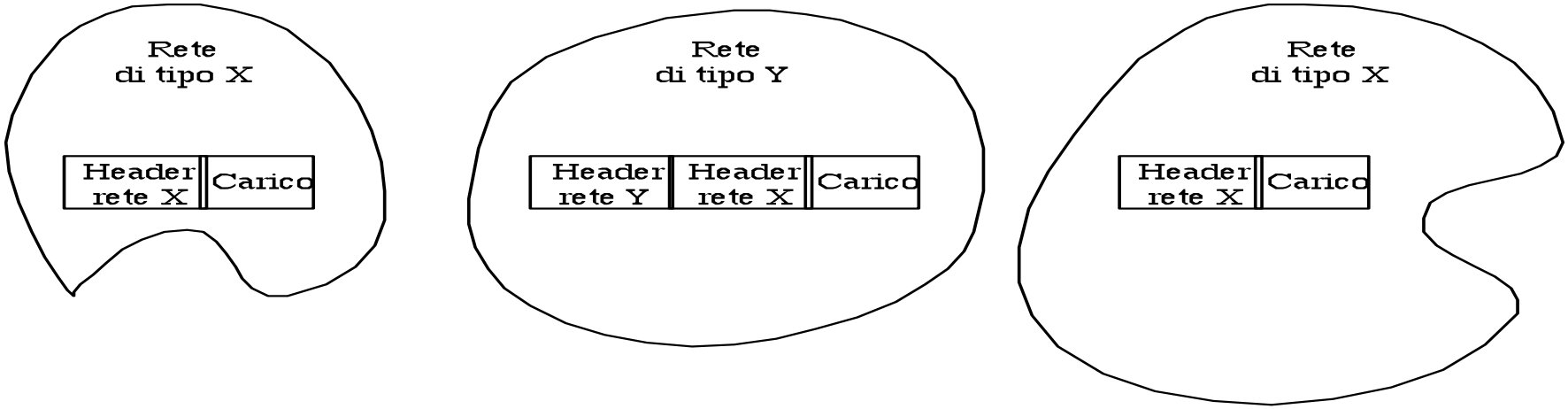
*diverse dimensioni massime dei pacchetti.

Reti di router multiprotocollo



Tunneling: collega due reti uguali attraverso una rete diversa.





Rete Y: pacchetti con due buste di livello network

Frammentazione: il livello network della rete di origine e di quella di destinazione devono prevedere meccanismi di

spezzettamento del pacchetto in ***frammenti*** prima di consegnarli alla rete di transito;

ricomposizione dei frammenti appena giungono dalla rete di transito a quella di destinazione

Il protocollo IP prevede questa funzionalità.

Circuiti virtuali concatenati: circuiti virtuali che si estendono attraverso più reti eterogenee come ***concatenazione di circuiti virtuali*** che attraversano ciascuno una delle reti.

I router multiprotocollo di confine:

* 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.

Un internetwok è in genere composta da singole reti (**AS**: **Autonomous System**) connesse tra loro da dorsali (backbone) ad alta velocità.

Routing a due livelli:

Interior Gateway Protocol (***IGP***): routing interno ad un **AS**. (vi possono essere diversi IGP per diversi AS, inoltre per AS di dimensioni considerevoli l'IGP può essere anche gerarchico).

Exterior Gateway Protocol (**EGP**): routing fra diversi AS. (Tipicamente, EGP è l'algoritmo di routing implementato dai router multiprotocollo).