

NETWORK LAYER

Il Network Layer, livello 3, svolge la seguente funzione principale:

trasportare dati contenuti in pacchetti, detti "datagrammi", da un **HOST** mittente ad un **HOST** destinatario.

Le entità del livello di rete sono presenti in TUTTI i nodi della rete, sia negli end-host che nei routers.

- Più nel dettaglio, QUALI FUNZIONI SVOLGE?

1) Path Determination: utilizza Algoritmi di Routing per calcolare, per ogni pacchetto, il percorso che deve seguire nella rete per giungere a destinazione.

2) Forwarding: muovere un datagramma da una porta di input di un router all'appropriata porta di output, mediante Tabelle di indirizzamento (FORWARDING TABLE)

3) Cell Setup (VC networks): in reti a circuito virtuale (ATM) fa il setup iniziale dello stato dei routers (percorsi) prima di trasmettere pacchetti.

CONNECTION SETUP:

Nelle reti a circuito virtuale, prima di cominciare col flusso di datagrammi, i 2 end-hosts + i ROUTER che intervengono stabiliscono una CONNESSIONE VIRTUALE → Sono COINVOLTI i ROUTER!

* Connessione a Livello di Trasporto (4):

(TCP) è ORIENTATO alla connessione! ⇒ È una connessione LOGICA tra 2 PROCESSI!

NETWORK SERVICE MODEL:

Cose offre la Rete Internet (IP)? Dà garanzie sul singolo datagramma o sul flusso?

→ La RETE INTERNET NON OFFRE NIENTE!

E' solo un servizio **BEST EFFORT**: fa del proprio meglio, ma non sa come garantisce!

Non offre garanzie sulla Bandwidth, né affidabilità sulla Perdita, né suff'arrivo in ordine di pacchetti, né sui ritardi, né dà feedback sulla congestione!

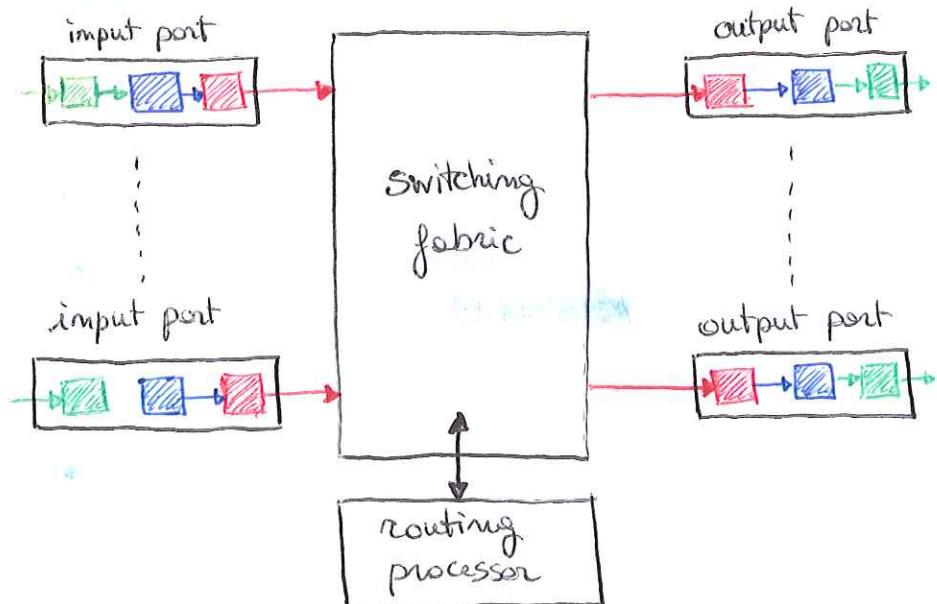
- Ma allora come mai le reti Internet ha avuto così tanto successo, prevalendo sulle reti delle ATM che offrivano molti servizi?

Uno dei principali motivi è l'elevato costo delle reti ATM; ma analizziamo meglio la situazione mettendole a confronto:

INTERNET (datagram network)	ATM (VC network)
<ul style="list-style-type: none"> • Rete a datogramme: Connectionless • Scambio di dati tra computers: <ul style="list-style-type: none"> - "ELASTIC" SERVICE, non ci sono restrizioni legate al tempo • "SMART" end-systems (computers): <ul style="list-style-type: none"> - possono adattarsi, fornire controllo d'errore e recuperare dall'errore - rete interna semplice: → COMPLEXITY AT "EDGE" • molte tipologie di canali di comunicazione: <ul style="list-style-type: none"> - caratteristiche differenti - un servizio uniforme è difficile 	<ul style="list-style-type: none"> • VC network: connection service • Human conversations: <ul style="list-style-type: none"> - richieste restrizioni di tempo e affidabilità - necessarie per i servizi garantiti • end-systems "STUPIDI" (telefoni): Le reti ATM evolvono sulle linee telefoniche. Gli operatori hanno scarsa capacità di calcolo ↳ TUTTA LA COMPLESSITÀ È NELLA RETE INTERNA.

* Per aggiungere o modificare funzionalità alla rete Internet, molto probabilmente dovrà agire sugli END-SYSTEMS, non c'è bisogno di andare a toccare tutti gli operatori di rete. Ciò la rende un'architettura RIUSABILE e molto più AGILE ed ORIENTATA AL EVOLVERE RAPIDAMENTE!

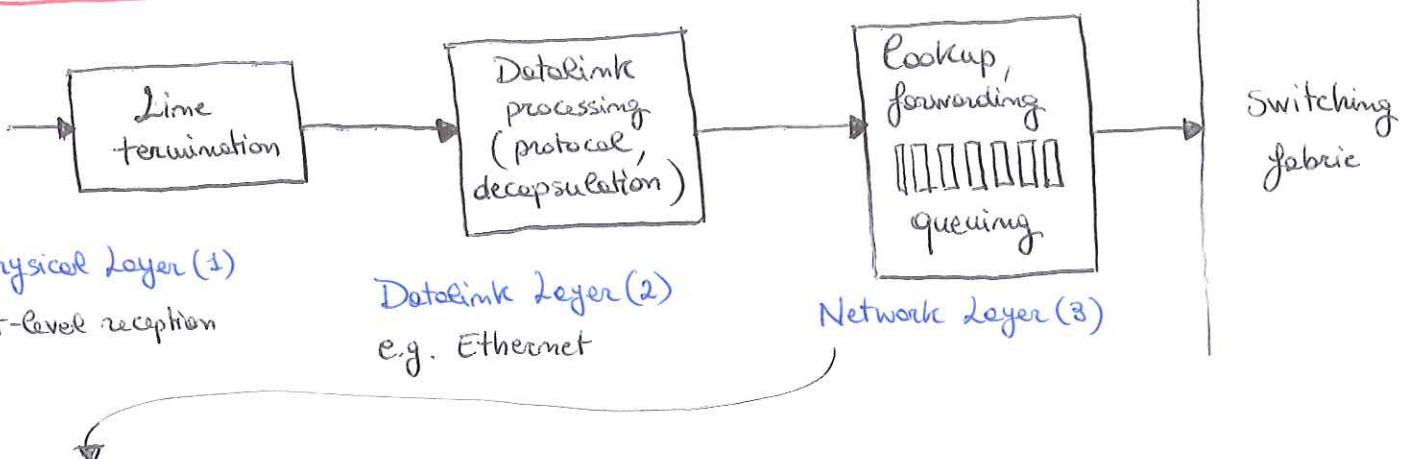
• Overview dell'ARCHITETTURA DI UN ROUTER:



2 FUNZIONI CHIAVE DEI ROUTER

- Eseguire PROTOCOLLI e/o ALGORITMI DI ROUTING
- INSTRADARE i datagrammi del link d'ingresso al link d'uscita (FORWARDING)

• INPUT PORT FUNCTIONS :



SWITCHING DECENTRALIZZATO :

- date la destinazione di un datagramma, consultare la FORWARDING TABLE delle input port (che ha in memoria) per decidere l'output port;
- obiettivo: completare il processamento delle input port a "velocità di linea";
- queuing: se i datagrammi arrivano più velocemente del rate di ~~trasmissione~~ commutazione delle switching fabric

• SWITCHING FABRIC :

Ci sono principalmente 3 tipologie di metriadi commutazione:

1) SWITCHING VIA MEMORY: è un computer tradizionale che fa da router, con un processo dedicato.

Molto INEFFICIENTE perché:

- pacchetti copiati nella memoria di sistema;
- velocità limitata delle lunghezze di bande delle memorie → 2 attraversamenti del bus per (condivisa) datagramma

2) SWITCHING VIA Bus: datagrammi delle input port alle output port passando direttamente per il bus di sistema (condiviso).

- Non pesa delle memorie
- Pesce 1 sola volta per il bus

Tuttavia:

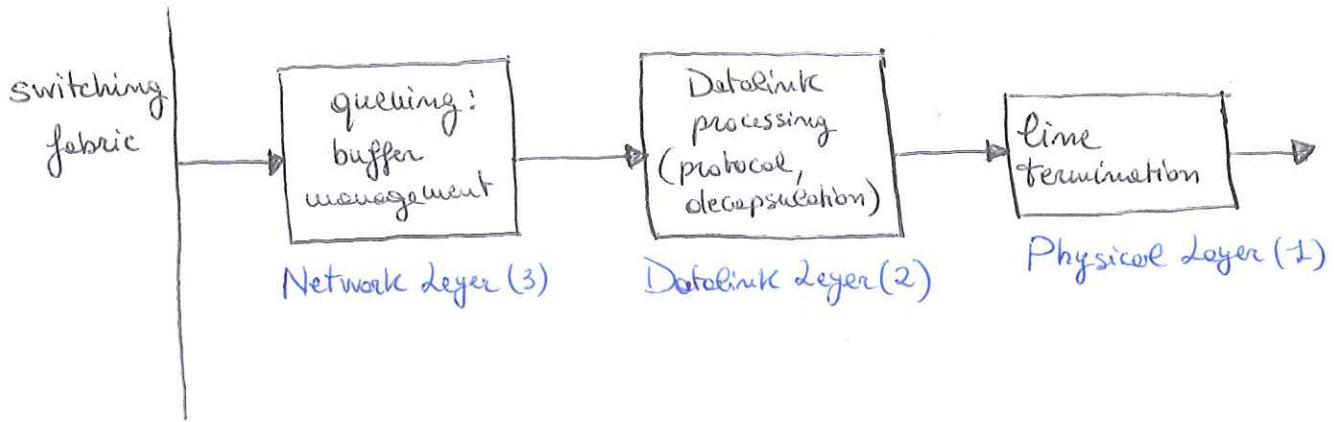
- limitato dalle bande del bus
- bisogna gestire le concorrenze al bus

3) VIA AN INTERCONNECTION NETWORK: ad esempio CROSSBAR, tecnologie già sviluppate per far comunicare processori in sistemi multiprocessore

- supera la limitazione delle bande del bus

- DESIGN AVANZATO: frammentare i datagrammi in celle di dimensione fissa e ricomporli in uscite → grandi velocità.

• OUTPUT PORT FUNCTIONS:



- BUFFERING: richiesto quando i datagrammi arrivano dalla matrice di commutazione più velocemente del rate di trasmissione

- SCHEDULING DISCIPLINE: politiche di scelte del datagramma da trasmettere tra quelli bufferizzati; può essere FIFO, Priority Queue, etc....

* QUEUING DELAY e LOSS dovuti a buffer overflow dell'output port.

• QUANTO BUFFERIZZARE?

Stabilito l'RTT "tipico" (e.g. 250 ms), se C è la capacità del link, viene consigliata una dimensione del buffer pari:

$$\text{buffer} = \text{RTT} \cdot C$$

Con N flussi si consiglia:

$$\text{buffer} = \frac{\text{RTT} \cdot C}{VN}$$

- Più grande è il buffer e meglio è (concretamente). Tuttavia, buffer grandi sono più costosi. Inoltre, sorge il seguente problema:

* Buffer grandi riducono le PERDITA di pacchetti, quindi invogliano TCP ad aumentare il rate di trasmissione, aumentando così le congestione.

→ I buffer e regime tenderanno ad essere pieni, introducendo un forte RALLENTAMENTO dovuto al QUEUING DELAY!

Riassumendo:

$$\text{BUFFER GRANDI} = \begin{cases} - \text{PERDITA (LOSS)} \\ + \text{RITARDO (DELAY)} \end{cases}$$

• INPUT Port Queuing :

- BUFFERING alle input port è NECESSARIO: la switching fabric puo' essere più lenta delle input ports combinate;

• HEAD-OF-THE-LINE (HOL) BLOCKING PROBLEM:

datalogramma bufferizzato in testa alle code, bloccato e causa delle CONTESA delle output port, che a sua volta blocca l'avanzamento degli altri datagrammi nelle code, che potrebbero essere trasmessi.

↳ Si puo' risolvere direttamente in HARDWARE, guardando non solo il 1° datagramma nelle code, ma anche i successivi.

* QUEUEING DELAY e LOSS dovuti anche all'overflow dei buffer delle output ports.

• PERCHÉ SI CHIAMA "INTERNET"?

Quando è stato proposto, esistevano già molte reti, con caratteristiche molto diverse tra loro del livello 3 in su.

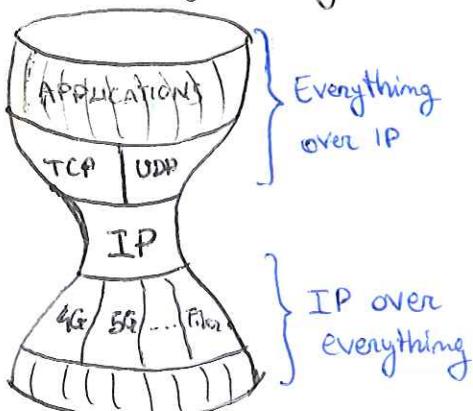
C'era il seguente problema: COME FARLE COMUNICARE?

Occorreva un nuovo meccanismo / protocollo che giresse SOPRA i livelli di rete esistenti, senza sostituirli, per farli comunicare in una "lingua comune", cioè IP!

IP! → INTER-NET, tra le diverse reti!

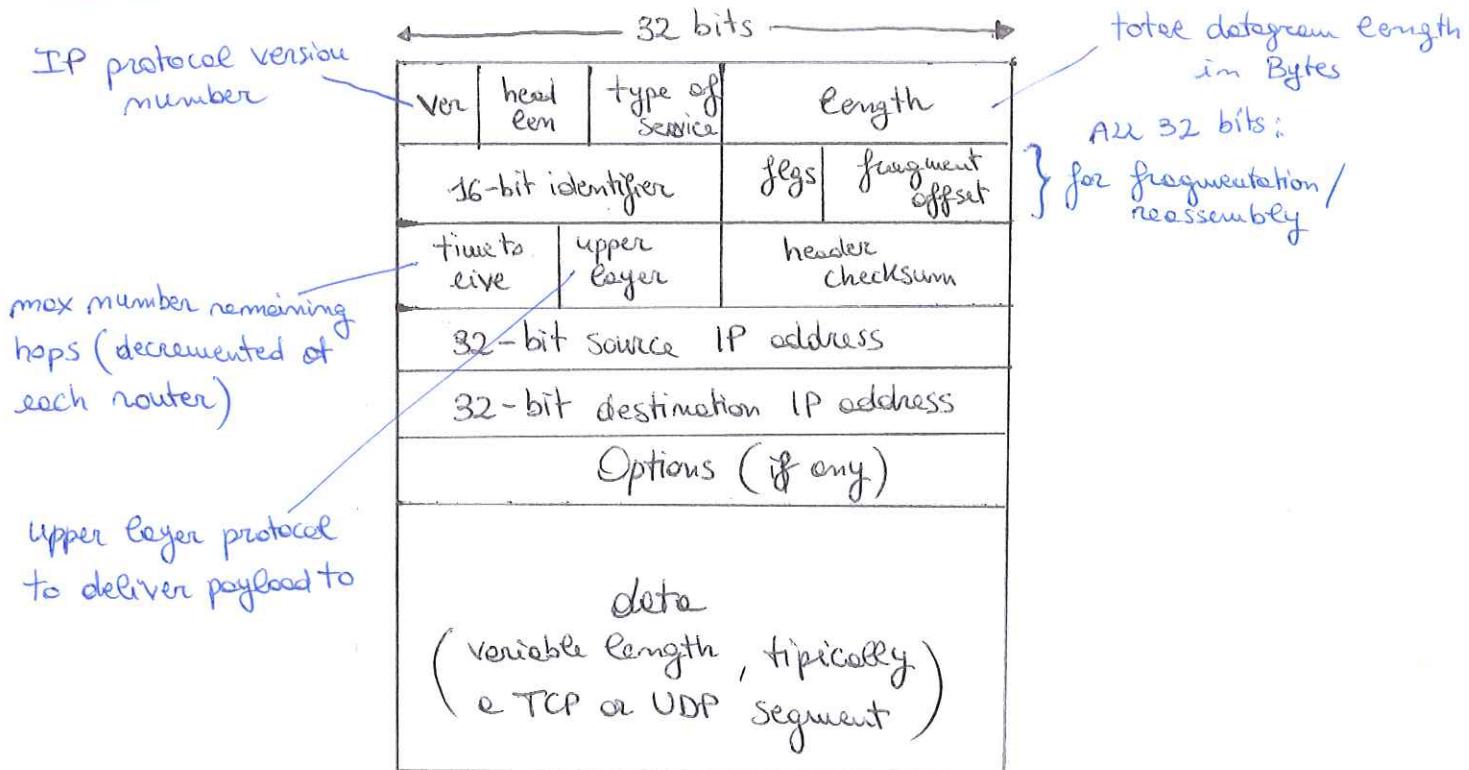
- I modi interposti tra una rete e l'altra per farle comunicare, che "parlavano IP", erano detti GATEWAY (portali); oggi sono i ROUTER.
- Oggi, esistono diverse reti: fisse in rete, fisse in fibra, 4G, 5G, Wi-Fi, ...; ma sono ai livelli 1 e 2 → Comunicano tutte con lo stesso livello di rete (3), cioè IP.

Cio' de' luogo alle famose CLESSIDRA (HOURGLASS):



"Everything over IP, IP over everything"

• IP DATAGRAM FORMAT:



• HEADER = (20 Bytes) obbligatori + optionali \Rightarrow Header TCP + Header IP = 40 Byte

* Indirizzi IPv4 a 32 bit : $2^{32} \approx 4 \text{ MILIARDI di indirizzi}$ \rightarrow è poco, sono finiti!

\hookrightarrow IPv6 usa indirizzi a 128 bit : 2^{128} indirizzi sarebbero largamente sufficienti, ma è difficile gestire la transizione da IPv4 a IPv6.

• TOS (type of service) : di solito inutilizzato, ma potrebbe essere usato ad esempio per gestire le priorità; oggi vengono usati per SERVIZI DIFFERENZIATI.

• header checksum : fa il checksum SOLO DELL'HEADER! Delle restanti parte è già fatto da TCP e UDP. È importante controllare l'header per non sbagliare destinazione o sorgente!

• upper layer (8 bit) : protocollo di livello superiore a cui consegnare il pacchetto (TCP, UDP o altro)

• time-to-live (8 bit) : tempo di vita massimo del pacchetto, quanto dovrebbe durare prima di sparire dalla rete. In realtà è un'approssimazione; **NUMERO DI HOP RIMANENTE CHE PUO' ATTRAVERSARE**; ogni router, alla ricezione del datagramma, decremente il TTL di 1.

* NOTA : il time-to-live è a 8 bit \Rightarrow max 255 HOP attraversabili!

* Il time-to-live serve per SCARTARE PACCHETTI, poiché si potrebbero creare dei **CICLI** di Hop nel route del datagramma, e cause di errori nelle tabelline di indirizzamento.

COSA FA IL ROUTER QUANDO RICEVE UN DATAGRAMMA?

1) Calcola il checksum dell'header e lo confronta:

- se coincide, PROCESSA il datagramma
- Se non coincide, scarta il datagramma

2) Decrementa di 1 il campo time-to-live

3) Guarda la forwarding table e invia il pacchetto verso l'output link corretto.

* NOTA: Al punto (2), modificare il campo time-to-live implica anche RICALCOLARE il nuovo CHECKSUM dell'header e sostituirlo.

↳ la fase (2) introduce ritardo e spesso viene DISABILITATA!

FRAMMENTAZIONE

• MTU := Maximum Transfer Unit di un frame a livello 2

• Essendoci diversi tipi di links, ci sono diversi MTUs.

• In IPv6, se un datagramma IP è troppo grande per il link di livello 2 viene SCARTATO.

• In IPv4, se il datagramma è troppo grande, i router delle reti provvedono alla FRAMMENTAZIONE del datagramma:

→ un datagramma diventa molti datagrammi

- vengono RIASSEMBLATI solo alla destinazione finale!

- si usa le 2° parole (di 32 bit) dell'header IP per identificare e ordinare i frammenti e gestire così la frammentazione e il riassemblaggio.

• Consideriamo adesso un esempio di frammentazione con i seguenti parametri:

- 4000 Byte datagram (20 header + 3980 payload)

- MTU = 1500 Byte \Rightarrow max 1480 Byte payload

	Length = 4000	ID = X	frag flag = 0	offset = 0	
--	------------------	-----------	------------------	---------------	--

1480 Byte in data field

FRAMMENTAZIONE

	Length = 1500	ID = X	frag flag = 1	offset = 0	
--	------------------	-----------	------------------	---------------	--

$$\text{offset} = \frac{1480}{8} = 185$$

	Length = 1500	ID = X	frag flag = 1	offset = 185	
--	------------------	-----------	------------------	-----------------	--

	Length = 1040	ID = X	frag flag = 0	offset = 370	
--	------------------	-----------	------------------	-----------------	--

Tutti i frammenti di uno stesso datagramma hanno lo STESMO ID (16 bit)

frag flag = 0 indica che è l'ULTIMO frammento; si differisce da un datagramma non frammentato poiché ha OFFSET ≠ 0

- Notare che l'OVERHEAD è TRIPPLICATO: da 20 Byte → a 60 Byte

* Si cerca di EVITARE LA FRAMMENTAZIONE il più possibile, proprio perché introduce overhead.

→ Tuttavia, ad oggi l'MTU limitante è quello di Ethernet, pari a 1500 Byte, e ormai tutte le applicazioni tendono a generare pacchetti di dimensioni non più grandi di tale MTU!

IP ADDRESSING

IP address: indirizzo identificativo a 32 bit (IPv4) per host/router INTERFACE

INTERFACE: connessione tra host/router e il link fisico
 - tipicamente i router hanno interface multiple
 - gli host tipicamente 1 unica interfaccia

- Usano la DOT NOTATION: e.g. 192.68.0.4

- Gli indirizzi possono essere divisi in 2 parti:

NetID	HostID
-------	--------

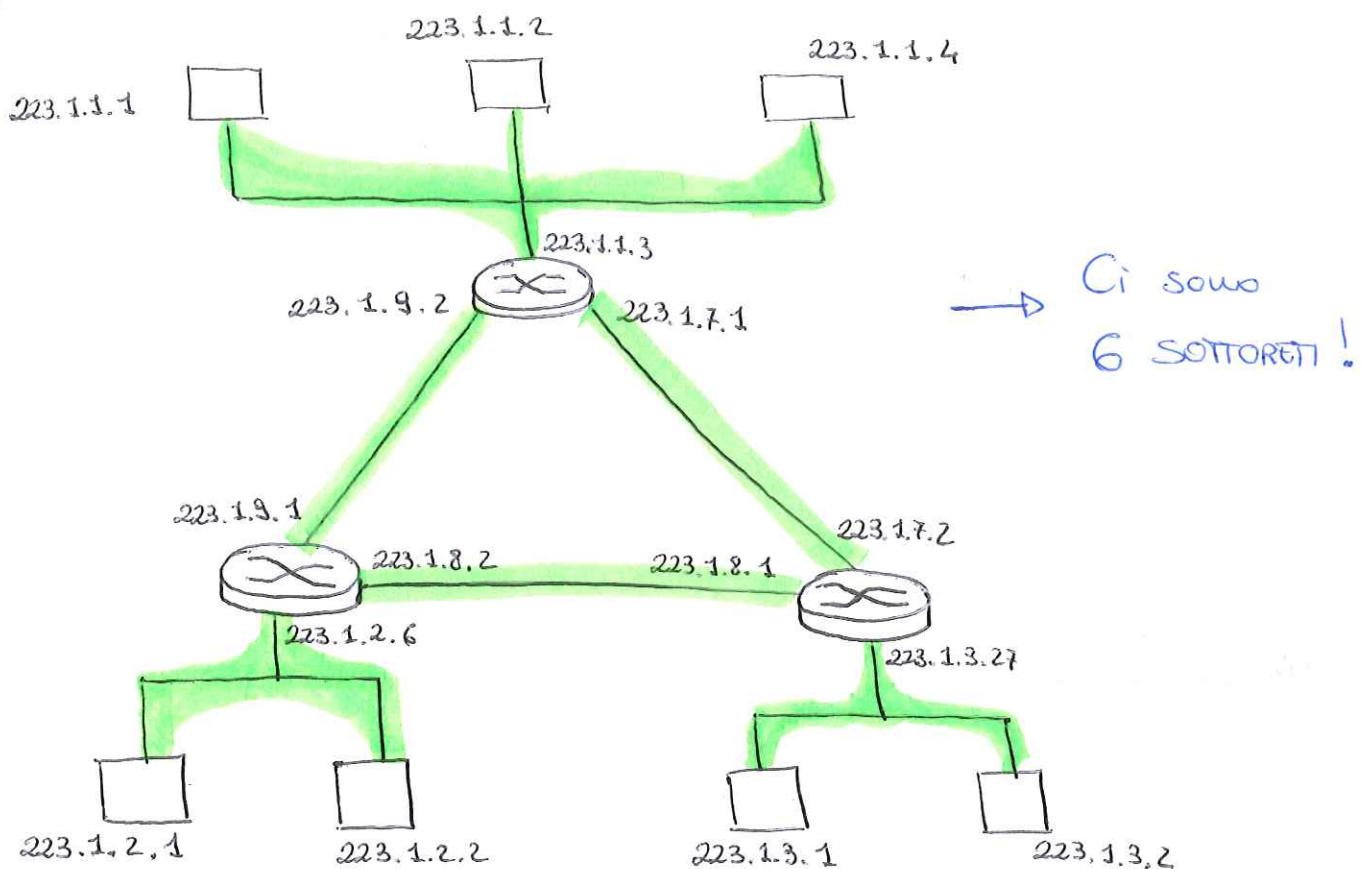
- NetID: identifica la rete
- HostID: identifica l'host all'interno della rete
- Host appartenenti alla stessa rete hanno lo stesso NetID.

SOTTORETE

Parte delle reti all'interno delle quali ciascun componente può comunicare con gli altri componenti, SENZA PASSARE PER UN ROUTER.
Una sottorete è detta anche "Rete IP".

- * In questo scenario, i **ROUTER** consentono la comunicazione tra host di sottoreti diverse; sono appunto dei "portelli" (GATEWAY).

- Ogni link tra 2 router è concettualmente un'ultra SOTTORETE!



"CLASS-FULL" ADDRESSING:

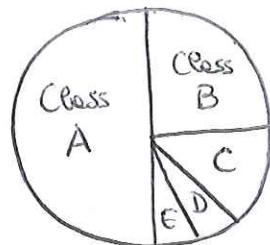
Si distinguono 4 classi di indirizzi, a seconda dei bit iniziali:

CLASS

	network	host			de	1.0.0.0
A	[0	network		host		0.0.0.0 - 127.255.255.255
B	[10	network		host		128.0.0.0 - 191.255.255.255
C	[110	network		host		192.0.0.0 - 223.255.255.255
D	[1110	multicast	address			224.0.0.0 - 239.255.255.255

Ampliamento dell'indirizzamento a classi:

- IP address è 32 bit $\Rightarrow 2^{32} = 4.294.967.296$ possibili indirizzi IP
- classe A : $2^7 - 2 = 126$ reti ($0.0.0.0$ e $127.0.0.0$ riservate)
 $2^{24} - 2 = 16.777.214$ massimi hosts per rete
 $\hookrightarrow 2.113.928.964$ host indirizzabili ($\approx 50\%$ del totale)
- classe B : $2^{14} = 16.384$ reti
 $2^{16} - 2 = 65.534$ massimi hosts per rete
 $\hookrightarrow 1.073.709.056$ host indirizzabili ($\approx 25\%$ del totale)
- classe C : $2^{21} = 2.097.152$ reti
 $2^{8} - 2 = 254$ massimi host per rete
 $\hookrightarrow 532.676.608$ host indirizzabili ($\approx 12,5\%$ del totale)

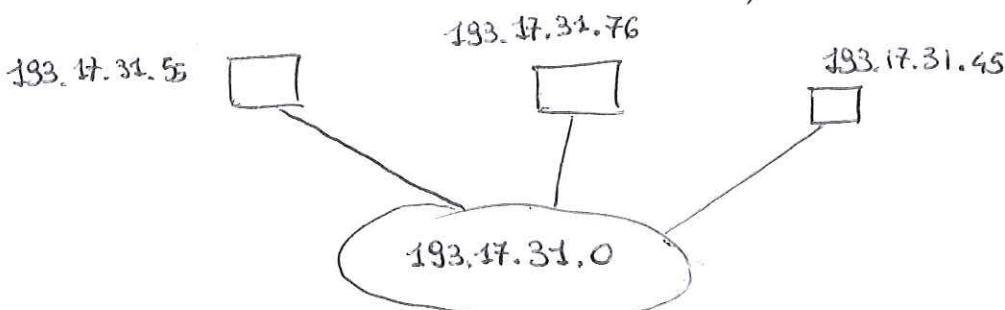


→ Strutture troppo RIGIDA!

Inoltre 4 RLD di indirizzi non bastano più. bisogna trovare un modo per renderli dinamici.

• INDIRIZZI SPECIALI:

- NETWORK ADDRESS: un indirizzo con HostID = 0 identifica la rete con il NetID corrispondente; usato nelle ROUTING TABLES.



- DIRECT BROADCAST ADDRESS: un indirizzo con tutti i bit della HostID pari a 1; è l'indirizzo BROADCAST DELLA SOTTORete indicato nel NetID

Invia pacchetti a tutti i nodi della sottorete; ad esempio:

193.17.31.255

*Nota: Non esiste un indirizzo di BROADCAST di tutte le reti Internet, a livello 2 questo è possibile, a livello di rete LAN (locale).

Non è previsto per motivi legati alla SICUREZZA: esporrebbe la rete Internet ad una multitudine di potenziali attacchi!

IP Addressing: CIDR

CIDR = Classless InterDomain Routing

- CLASSLESS: gli indirizzi IP non sono più divisi rigidamente in classi, ma le lunghezze del NETID (Subnet Mask) è arbitraria.
- Segue il formato: **a.b.c.d./x**, dove x è il numero di bits delle indirizzi che identificano la sottorete, cioè le lunghezze del NetID.
Esempio: 192.68.1.32/24 \Rightarrow /24 implica: netmask = 0xffffffff
net = 192.68.1.0

CHE FA UN Host AD OTTENERE UN INDIRIZZO IP?

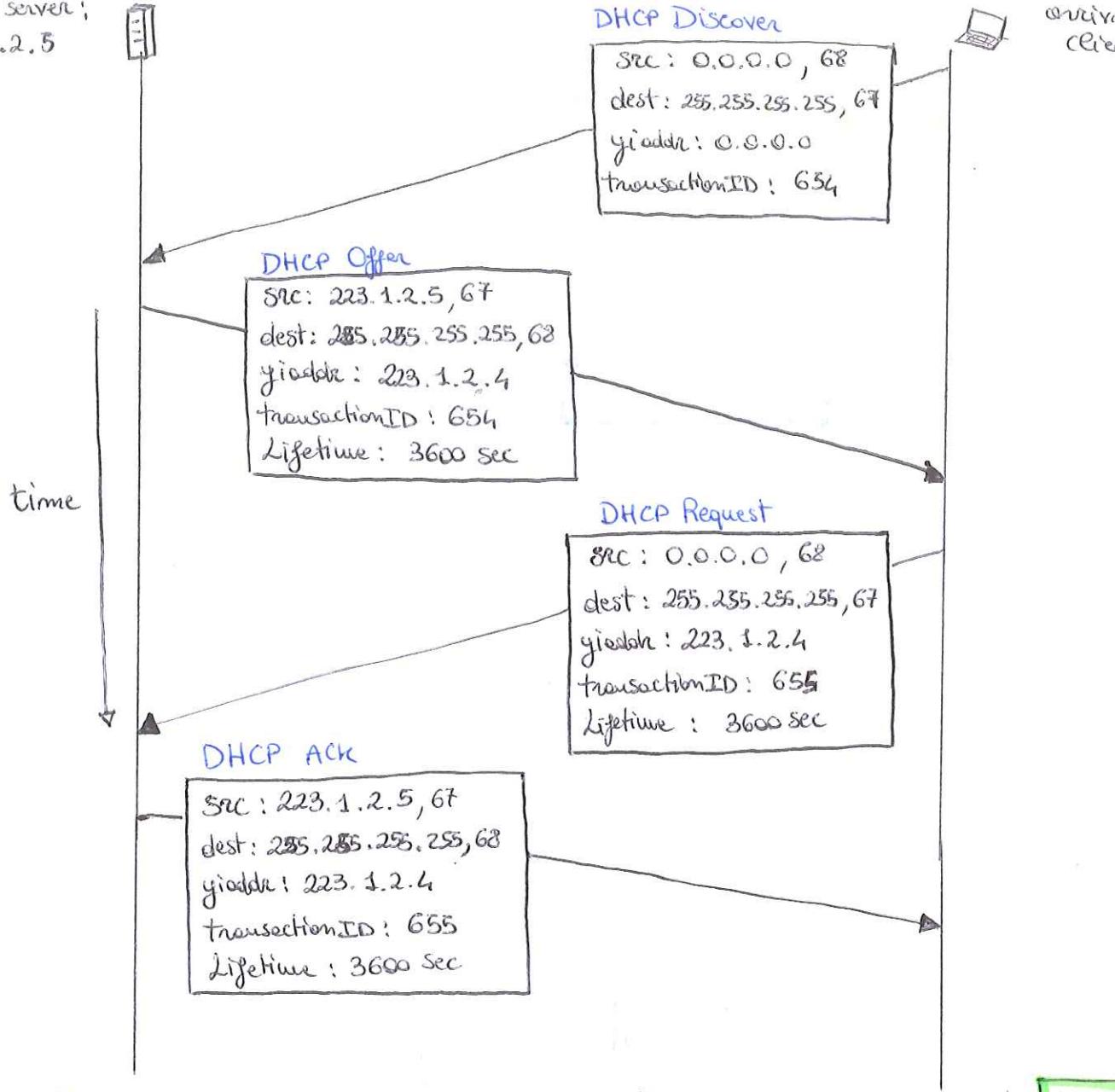
- 1) O viene scritto "hard-coded" in un file da un amministratore di sistema; così ognuno si configura il proprio, ma potrebbero esserci problemi.
- 2) **DHCP** = Dynamic Host Configuration Protocol:
ottenere dinamicamente indirizzi IP da un server, ("plug-and-play").

DHCP

E' un PROTOCOLO DI LIVELLO APPLICATIVO! E' previsto un server DHCP all'interno di una sottorete

- OBIETTIVO: consentire agli host di ottenere **DINAMICAMENTE** il proprio indirizzo IP della rete (del DHCP server) quando si aggiunge alla rete.
 - un host può rinnovare il suo ultimo indirizzo in uso
 - Consente il RIVISO DEGLI INDIRIZZI \rightarrow permette di gestire più utenti dei 2^{32} indirizzi possibili di IPv4
 - Supporta bene gli utenti mobili che vogliono aggiungersi alla rete, ma vi permaneggi per tempi brevi
- Il client comunica col server DHCP sulla **PORTA 67**
- Il server comunica col client sulla **PORTA 68**
- Il protocollo DHCP a livello di trasporto utilizza UDP!

DHCP server:
223.1.2.5



- **DHCP Discover:** Il client, non avendo un IP, usa come IP sorgente 0.0.0.0 e la porta 68. Non sapendo chi contattare, invia datagrammi in BROADCAST LOCALE (255.255.255.255) sulla porta 68. Inoltre, genera pseudo-random un transactionID per farsi riconoscere nei messaggi successivi.
- **DHCP Offer:** Il server risponde con un'offerta di indirizzo IP (yaddr) e me dà un LIFETIME, invia sempre in broadcast locale, su porta 67!
- **DHCP Request:** Il client invia l'effettiva richiesta dell'IP address proposto
- **DHCP ACK:** Il server invia l'ACK di risposta, alla cui ricezione il client ottiene l'indirizzo IP richiesto per un tempo pari al Lifetime!
- * **Nota:** è il LIFETIME ad essere fondamentale per poter RIVISARE gli indirizzi!

ROUTING

Supponiamo che ci siano 3 host: A, B appartengono alla stessa sottorete, mentre C fa parte di un'altre sottorete.

- Se un host, diciamo A, vuole inviare un pacchetto, ci sono 2 possibilità, a seconda se è l'host destinatario:

- Si trova nelle stesse sottorete (e.g. B):

A consulta le proprie tabelle di indirizzamento, vede che B è nella sua stessa rete e usa direttamente la RETE LAN A LIVELLO 2

- Si trova in un'altre sottorete (e.g. C):

A consulta le proprie forwarding table e vede che deve inviare il datagramma al router, la cui interfaccia è nella stessa rete di A → Use il LIVELLO 2

FORWARDING TABLE IN A

Dest. Net	Next Router	Nhops
223.1.1.0/24		1
223.1.2.0/24	223.1.1.4	2
223.1.3.0/24	223.1.1.4	2

FORWARDING TABLE NEL ROUTER

Dest. Net	Next router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27

Quale interfaccia il router deve usare per comunicare con un altro router o con altri host.

Pertanto QUALE LINK D'USCITA USARE!

- In realtà, le forwarding table dei router non servono il mapping sul singolo indirizzo di rete, ma lavorano per INTERVALLI di indirizzi!

Per riconoscere le reti usano i PREFISSI!

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

- Notare che il 3° prefisso è a sua volta un prefisso del 2° prefisso. Se un indirizzo matche con entrambi che si fa? → LONGEST PREFIX MATCHING

Si sceglie il 2° prefisso, quindi link d'interfaccia = 1.

- * Questo meccanismo si può implementare in HARDWARE → Permette di gestire otto numeri di pacchetti al secondo.

COME FA UNA SOTTORETE AD OTTENERE IL PROPRIO INDIRIZZO IP?

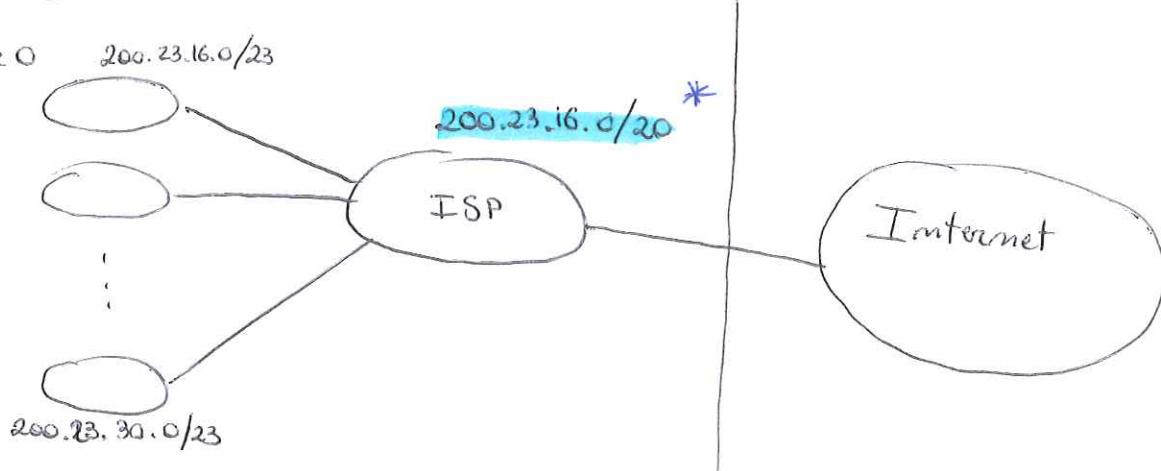
Glielo fornisce l'ISP (Internet Service Provider), assegnandogli una porzione, generalmente contigua, di indirizzi del proprio spazio di indirizzamento.

Noterete le seguenti cose:

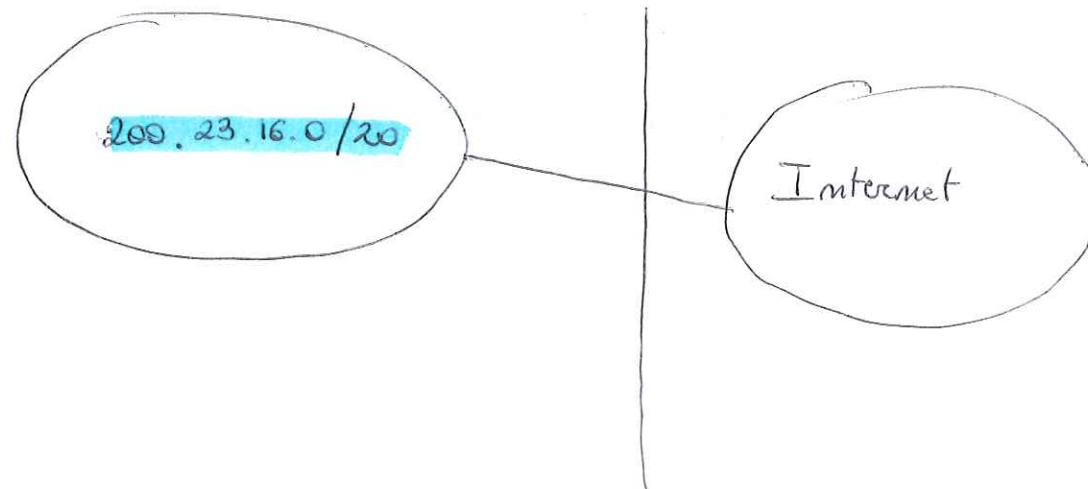
(1)

Organizzazione 0 200.23.16.0/23

Org. 1



(2)



* Del punto di vista dell'indirizzamento, i 2 scenari sono UGUALI, poiché il prefisso è lo stesso.

* L'unica differenza è che nel 1° caso c'è l'ISP e dovrà fare dei transiti tra le organizzazioni e il resto di Internet.

* Un PREFISSO può rappresentare una SINGOLA RETE (caso particolare), ma, nel caso generale rappresenta un INSIEME DI RETI !

Questo è possibile perché gli indirizzi hanno una STRUTTURA GERARCHICA.

* Ciò permette di fare ROUTE AGGREGATION: aggregare un numero grande di indirizzi tramite i prefissi.

$$\text{PREFISSO} = \text{INSIEME DI RETI} = \text{INSIEME DI INDIRIZZI}$$

• Un ISP come ottiene gli indirizzi IP?

C'è una corporazione che gestisce ciò:

ICANN = Internet Corporation for Assigned Names and Numbers

- alloca indirizzi
- gestisce DNS
- assegna nomi di dominio e risolve le dispute

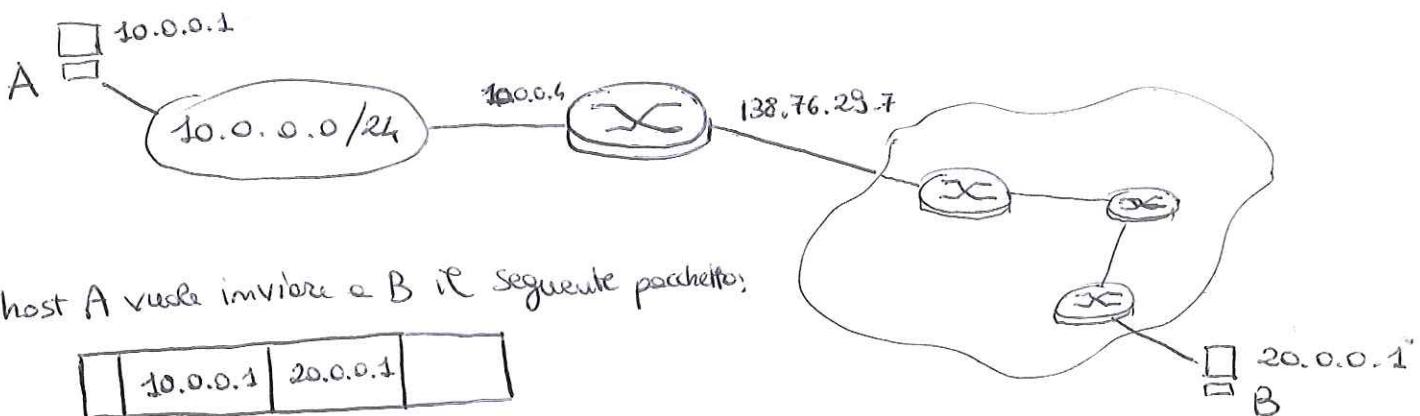
* Problema: Lo spazio di indirizzamento di IPv4 è FINITO!

Si "risolve" tramite ROUTER NAT e i conseguenti concetti di:

- Rete IP Pubblica: mato e tutti
- Rete IP Privata: mato solo nelle LAN (Local Area Network).

• NAT : Network Address Translation

Permettono di usare stessi indirizzi IP in sottoreti diverse (IP Privati).



L'host A vuole inviare a B il seguente pacchetto:

10.0.0.1	20.0.0.1
----------	----------

• Un router NAT esegue la seguente operazione:

per ogni datagramma inviato da un host della Rete IP Privata verso il resto di Internet, **SOSTITUISCI l'IP sorgente con l'IP della PROPRIA INTERFACCIA del link d'uscita** (quello verso Internet)

Il pacchetto modificato risulterà:

138.76.29.7	20.0.0.1
-------------	----------

* Il pacchetto risulta generato dal ROUTER!

VANTAGGI

- Risparmio di indirizzi non necessari dell'ISP: me basta uno per tutti i dispositivi delle sottoreti
- Possibilità di combinare gli indirizzi IP dei dispositivi nelle LAN senza dover modificare il mondo esterno

- Si può cambiare ISP senza modificare gli indirizzi dei dispositivi nella LAN
- I dispositivi nella LAN non sono direttamente indirizzabili / visibili alla rete esterna ("pubblica") → Migliore la SICUREZZA.

IMPLEMENTAZIONE NAT:

Il router NAT deve:

- OUTGOING DATAGRAMS: ricomporre (source IP addr., port #) di ogni datagramma in uscita con (NAT IP addr., new port #)
 - ... clients/servers remoti risponderanno usando (NAT IP addr., new port #) come indirizzo destinazione
- REMEMBER (in NAT Translation Table): ogni (source IP, port #) to (NAT IP addr., new port #) mapping tracciato.
- INCOMING DATAGRAMS: sostituire ogni (NAT ip addr., new port #) nel CAMPO DESTINAZIONE col corrispondente (source IP addr., port #) memorizzato nella NAT ~~forwarding~~ ^{translation} table.

NAT Translation Table	
WAN side addr	LAN side addr
138.76.29.7 , 5001	10.0.0.1 , 3345
---	---

- (1) Il client non si accorge che c'è un router di mezzo che fa da NAT!
- (2) Nemmeno il server se ne accorge! Dal suo punto di vista agisce come se il client fosse il router!

Osservazioni:

- * Se arriva dalla WAN un pacchetto che non ha una corrispondenza nelle tabelle NAT, viene SCARTATO, perché considerato un potenziale attacco.
- * Il numero di connessioni simultanee che la LAN può avere è limitato dal numero di porte del router NAT:
«16-bit port # ≈ 65.536 possibili connessioni simultanee»
Si può risolvere o dando più di 1 indirizzo al router NAT oppure disponendo più router NAT!
- * Per funzionare, ha bisogno di guardare e memorizzare i NUMERI DI PORTA
 - HA BISOGNO DI GUARDARE AL LIVELLO 4

- * Modificando gli indirizzi IP negli header dei datagrammi, VIOLA il percorso END-to-END: crea diversi problemi, specialmente in applicazioni P2P o in applicazioni che si scambiano info, come ad esempio il proprio indirizzo IP.
- * Tuttavia, è un meccanismo che consente di avviare oltre le limitazioni dello spazio di indirizzamento offerto da IPv4, dovrebbe però essere risolto da IPv6.
- * Inoltre NAT, modificando gli header, devono RICALCOLARE IL CHECKSUM: costo prestazionale potenzialmente elevato!

• NAT TRAVERSAL PROBLEM :

Se nella rete privata c'è un Server, a cui i clients vogliono connettersi, come si fa ad "attraversare" le NAT? L'indirizzo del server è privato?!

(1) PORT FORWARDING: configurare STATICAMENTE il mapping tra IP privato e IP pubblico del server nella NAT Translation Table.

↳ PROBLEMA: Se ci sono più server di una stessa organizzazione (stesso IP), una volta usato un port number per 1 server, non posso usarlo anche per un altro nella tabella!

(2) Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol:

Consente ai NATted hosts di:

- impostare l'indirizzo IP pubblico
- aggiungere/rimuovere i mapping di porte

Sostanzialmente automatizza la configurazione statica del mapping di porte NAT.

↳ PROBLEMA: Come fanno i clients delle app (P2P) a comunicare e contattarsi tra loro? Non funziona più niente!

(3) RELAYING (usato in Skype):

- C'è un server esterno che fa da RELAY, da intermediero tra 2 clients.
- I client, NATted o meno, si connettono al server relay
- Il server RELAY deve avere un indirizzo IP PUBBLICO!

↳ Un'app P2P si trasforma in un'app CLIENT-SERVER per poter funzionare.

- * Il server relay potrebbe fare "SNIFFING" del traffico dei client; per questo motivo il traffico viene CIFRATO e DECIFRATO negli end-host, ma essendo i proprietari dell'app a gestire il server e le cifrature, non ci sono grosse gerende di sicurezza.

ROUTING ALGORITHMS

Si usa l'estrazione di GRAFO $G = (N, E)$, dove N è l'insieme dei nodi della rete ed E è l'insieme dei link che connettono i vari nodi.

- Tipicamente, i costi degli archi, ad esempio $C(x, y)$, dovrebbero essere pari a 1, oppure dipendere inversamente dalle lunghezze di buona delle link oppure dipendere proporzionalmente alla congestione sul link.
- ROUTING ALGORITHM = algoritmo che trova il **CAMMINO DI COSTO MINIMO**!
- CLASSIFICAZIONE:

Una prima suddivisione riguarda come l'informazione è presente ai nodi, se è un'informazione globale o decentralizzata:

- **GLOBALE**: tutti i router hanno informazione sulla **COMPLETA TOPOLOGIA** della rete e su tutti i costi dei link
 - **LINK STATE ALGORITHM** (e.g. Dijkstra)
- **DECENTRALIZZATA**: ogni router conosce solo i router adiacenti ("vicini") e i costi dei link con cui è collegato fisicamente a tali router.
Il processo di computazione è iterativo e c'è scambio di informazioni tra router vicini.

- **DISTANCE VECTOR ALGORITHM**

Un'altra suddivisione si fa a seconda se l'algoritmo sia statico o dinamico:

- **STATICO**: i cammini e i costi dei link cambiano molto lentamente nel tempo.
- **DINAMICO**: i cammini cambiano molto velocemente, in risposta al cambio di costo dei link; richiede un update periodico di informazioni.
 - * È molto utile in reti Wireless, specialmente in contesti militari, anche perché permette di essere molto agili in caso di guasti.
(AD HOC NETWORKS)

* Gli algoritmi che vedremo, anche se non è strettamente necessario, tendono ad essere tutti dinamici!

• ALGORITMO DI DIJKSTRA:

Le topologie delle reti e i costi dei link sono noti ad ogni nodo; questo è realizzato mediante il "LINK STATE BROADCAST": tutti i nodi si scambiano info periodicamente sullo stato dei link cui sono connessi; le informazioni vengono diffuse tramite il meccanismo del FLOODING (visto in P2P), in cui ogni nodo invia le info che riceve ai propri vicini.

- Ogni nodo, tranne l'algoritmo, calcola il cammino di costo minimo da se stesso a TUTTI gli altri nodi della rete, ricevendone così le proprie FORWARDING TABLE
- È ITERATIVO: dopo K iterazioni si conoscono K cammini minimi verso k destinazioni!

NOTAZIONI: $c(x,y)$ = costo link (x,y) ; $D(v)$ = valore corrente del costo del cammino dalla sorgente alla destinazione v .
 $p(v)$ = nodo predecessore di v nel cammino; N' = insieme di nodi il cui cammino minimo è definitivamente determinato.

• ALGORITMO:

Initialization:

```

 $N' = \{u\}$ 
for all nodes  $v$ 
  if  $v$  adjacent to  $u$ 
    then  $D(v) = c(u,v)$ 
  else  $D(v) = \infty$ 

```

Loop:

find w not in N' such that $D(w)$ is a minimum

add w to N'

update $D(v)$ for all v adjacent to w and not in N' :

$$D(v) = \min(D(v), D(w) + c(w,v))$$

until all nodes in N'

- Calcolando tutti i percorsi di costo minimo, ogni nodo ottiene un albero, il cosiddetto MINIMUM SPANNING TREE, tramite le quali puo' settare le forwarding table!
- Nonostante l'algoritmo calcoli tutto il percorso, nelle forwarding table si registra solo le NEXT HOP, il prossimo link da usare!

COMPRESSITÀ ALGORITMO:

m modi; ogni iterazione deve controllare tutti i modi w mon im N^t

$$\Rightarrow \frac{m(m+1)}{2} \text{ confronti} \rightarrow O(m^2)$$

- Sono possibili implementazioni più efficienti: $O(m \log m)$

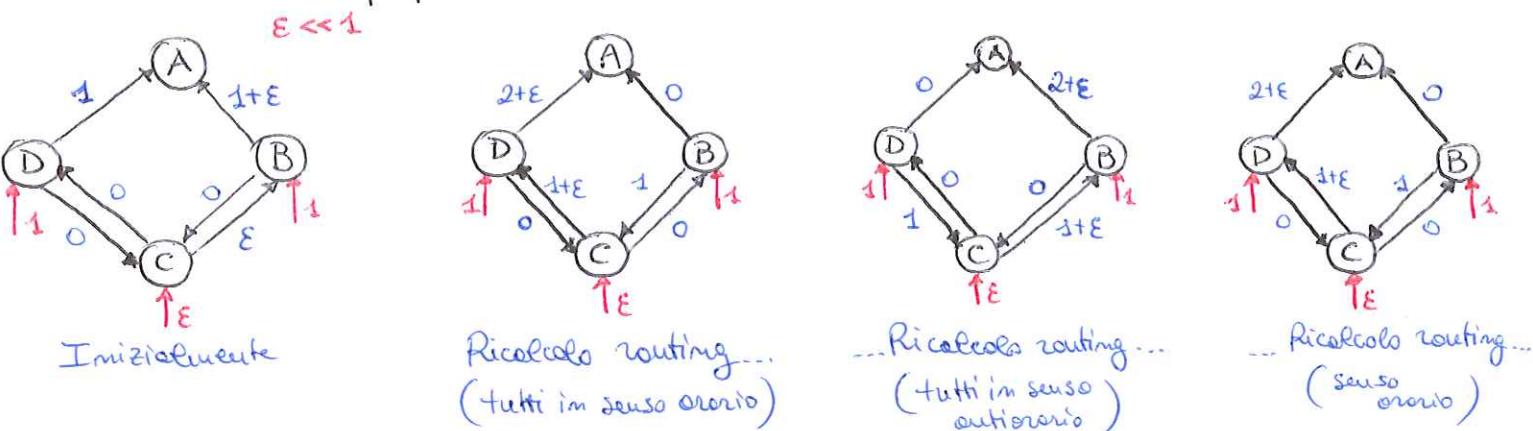
Ora però sorge la seguente domanda: in base a cosa stabilire il costo del link?

In realtà, quello che più impatta il "costo" di un link sarebbe il RITARDO DI CODA, principale cause di congestione.

Tuttavia, viene suggerito di usere $\frac{1}{C}$, con C capacità del link. Ma perché?

OSCILLAZIONI:

Supponiamo che i nodi B, C, D debbano inviare traffico al nodo A. Assumiamo che il costo sia proporzionale alle quantità di traffico!



* Si entra in LOOP, ottenendo un' OSCILLAZIONE DELLA RETE

Ci sono 2 problemi principali:

- 1) Nel ricalcolo del routing, non si considera l'impatto del proprio traffico nel costo dei link; tuttavia, in una rete con un'enormità di router che trasmettono traffico, tende ad essere trascurabile e a non impattare.
- 2) TUTTI I NODI RAGIONANO ALLO STESSO MODO, fanno tutte le stesse scelte e si crea congestione!

↳ Il traffico tende ad oscillare!

E' per questo motivo che si preferisce non usare funzioni di costo che siano dinamiche in funzione del carico, in quanto funzionano bene dal punto di vista del singolo nodo, ma non se applicate a tutti i modi della rete!

DISTANCE VECTOR ALGORITHM:

Sie $d_x(y) :=$ il costo del percorso di costo minimo da x a y ;

si cerca di sfruttare l' EQUAZIONE DI BELLMAN - FORD:

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

* Viene scelto solo tale modo v , tanto interesse solo il next hop! Notare che l'algoritmo non calcola quale sia il cammino, ma solo il costo minimo!

- L'idea è di usare l'equazione di Bellman-Ford in maniera **ITERATIVA**, secondo delle **STIME**.

Notazione: $D_x(y) =$ stima del costo minimo
da x a y ; $c(x,y) =$ costo dell'arco (x,y)

- Il modo x mantiene il **DISTANCE VECTOR** $D_x = [D_x(y) : \forall y \in N]$
- Il modo x mantiene anche i **DISTANCE VECTOR di tutti i suoi vicini**!

Idee Base:

- Periodicamente, si invia il proprio DV delle stime ai propri vicini; lo si fa in maniera **ASINCRONA**, indipendentemente degli altri nodi.
- Quando un nodo riceve un DV da un proprio vicino, **USA L'EQUAZIONE DI BELLMAN-FORD** per aggiornare il proprio DV:

$$D_x(y) \leftarrow \min_v \{ c(x,v) + d_v(y) \} \text{ for each modo } y \in N$$

* Sotto normali condizioni, queste stime $D_x(y)$ CONVERGE AL VALORE OTTIMO $d_x(y)$.

- Notare che i DV sono comunicati solo ai propri vicini! Questo riduce di molto il costo che invece si paga in caso di **FLOODING**.

ITERATIVE, ASYNCHRONOUS

Ogni iterazione locale (ad un nodo) è costituita da:

- cambio del costo di un link locale
- messaggio da parte di un nodo vicino di fare DV update

DISTRIBUTED

Ogni nodo notifica i vicini solo quando il proprio DV cambia!

I vicini notificano a loro volta i propri vicini SOLO SE NECESSARIO!

* NOTA: Dijkstra richiede più messaggi di DV, ma ha subito un risultato ottimo; invece, con DV si ha una **FASE DI TRANSITORIO**, in cui i costi e i cammini potrebbero non essere ottimi.

- Nelle fasi di inizializzazione, tipicamente si comunica il proprio DV ai vicini, mentre che ogni nodo, a regime, conosce sia il proprio DV che quello comunicato gli dai suoi vicini precedentemente.

COSA SUCCIDE SE UN LINK CAMBIA COSTO?

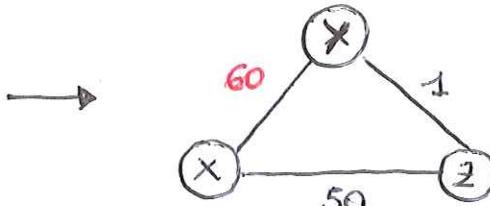
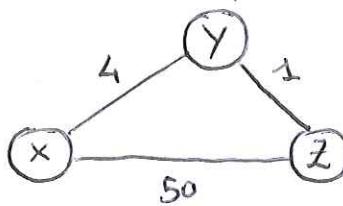
- 1) Il nodo rileva il cambiamento
- 2) Ricalcola il DV e, se le distanze cambiano, aggiorna le informazioni di routing
- 3) Se il DV è cambiato, inviatolo ai vicini

- "GOOD NEWS TRAVEL FAST": se il costo del link DIMINUISCE, allora tutti i nodi si aggiornano velocemente.

BAD NEWS TRAVEL SLOW:

Se il costo di un link AUMENTA, ci vorranno molte iterazioni per far sì che il nuovo costo venga propagato correttamente ai vari nodi → COUNTLESS TO INFINITY PROBLEM

Vediamo un esempio:



(Y)	X	Y	Z
D _X	0	4	5
D _Y	4	0	1
D _Z	5	1	0

(Y)	X	Y	Z
D _X			
D _Y			
D _Z			

$$D_Y(x) = \min \{ c(y,x) + D_X(x), c(y,z) + D_Z(x) \}$$

$$= \min \{ 60+0, 1+5 \} = 6$$

↓ Comunica nuovo DV ai vicini

(Z)	X	Y	Z
D _X	0	4	5
D _Y	4	0	1
D _Z	5	1	0

(Z)	X	Y	Z
D _X			
D _Y			
D _Z			

$$D_Z(x) = \min \{ 50+0, 1+6 \} = 7$$

↓ Comunica ai vicini

* Y e Z si riempiono messaggi errati, fino a trovare il percorso giusto, aumentando il costo di 1 unità alle volte → Servono 44 ITERAZIONI!

PROBLEMI:

- 1) L'algoritmo ci mette tanto a convergere!
- 2) Durante le transizioni, i nodi sono indotti all'errore, si creano dei Loop nella rete che ne impedisce il corretto funzionamento (riempimento di datagrammi)

• Poisoned Reverse:

In italiano è "l'AVVELENAMENTO del PERCORSO INVERSO":

Quando un nodo comunica il proprio DV ad un vicino, se il costo che gli comunica, entry per entry, è ottenuto passando per quel vicino, allora lo imposta ad OO, eccetto se è proprio il percorso verso il vicino.

Ad esempio, per il nodo Y, si avrà:

(Y)	x	y	z
Dx	0	4	OO
Dy	4	0	1
Dz	OO	1	0

$$D_y(x) = \min \{ 60 + 0, 1 + \text{OO} \} = \underline{\underline{60}}$$

(Z)	x	y	z
Dx	0	4	5
Dy	60	0	1
Dz	50	1	0

$$D_z(x) = \min \{ 50 + 0, 1 + 60 \} = \underline{\underline{50}}$$

* Con 3 iterazioni è già tutto risolto!

ALGORITMO LS vs DV

	LS	DV
Message complexity	Con m nodi, E archi: $O(m^2)$ msg inviati	Scambio solo tra vicini: il tempo di convergenza varia
Speed of Convergence	$O(m^2)$ l'algoritmo e richiede $O(m^2)$ msg: - ci possono essere oscillazioni	Tempo di convergenza variabile: - si possono formare Loop - count to infinity problem
Robustness (in caso di gesti)	- i nodi possono comunicare il costo errato - ogni nodo calcola solo le sue tabelle	- i nodi possono comunicare il costo di un percorso errato - ogni nodo ha una tabella usata da altri nodi → l'errore si propaga nelle reti

* Essendo la rete reale molto grande, l'algoritmo DV, anche con Poisoned Reverse, ci metterebbe tanto a convergere!

* Inoltre, le varie reti private vogliono essere autonome e gestire loro il routing all'interno delle proprie reti.

Si lavora su ROUTING GERARCHICO → HIERARCHICAL ROUTING

HIERARCHICAL ROUTING

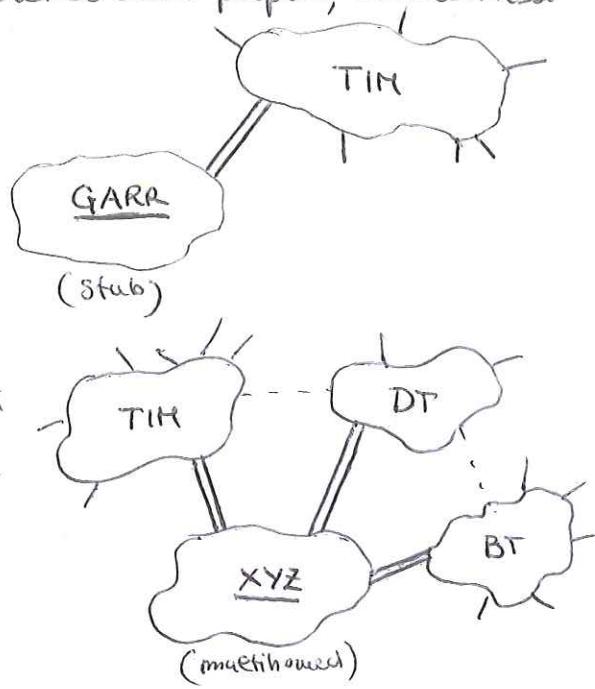
- AUTONOMOUS SYSTEM (AS): è un aggregato di routers in una "regione"
- Router nello stesso AS eseguono lo STESO ROUTING PROTOCOL:
 - "intra-AS" routing protocol
 - routers in AS differenti possono eseguire protocoli di routing diversi!
- ROUTER DI GATEWAY:
 - sono router "speciali" in un AS
 - eseguono l' intra-AS routing protocol con gli altri routers dello stesso AS
 - sono responsabili per il routing verso destinazioni ESTERNE dell' AS; eseguono l' inter-AS routing protocol con gli altri routers di gateway.
- * Le FORWARDING TABLE di TUTTI i ROUTER (non solo i gateway) vengono compilate in base sia all' inter-AS protocol che all' intra-AS protocol.
 - Intra-AS → destinazioni interne dell' AS
 - Intra-AS + Inter-AS → destinazioni esterne, verso altri AS.
- INTER-AS TASKS:
Deve svolgere i seguenti compiti:
 - 1) LEARNING: far imparare quali destinazioni sono raggiungibili attraverso gli altri AS vicini, a cui si è collegati.
 - 2) DISSEMINATION: propagare tali informazioni di routing a tutti i routers nell' AS considerato.
- I router di gateway fanno da tramite tra gli altri router nello AS e le destinazioni esterne.
- Se più AS possono raggiungere la stessa rete esterna X, per quale AS passare per raggiungere X?
E' compito dell' Inter-AS protocol stabilire, lo fa adottando il cosiddetto principio dell' HOT POTATO ROUTING:

si invia il pacchetto al router di gateway più "vicino", cioè con costo minore per raggiungerlo.
Ovviamente router di gateway del PROPRIO AS!

• ROUTING IN INTERNET :

Internet consiste in diversi AS, ognuno con caratteristiche proprie, interconnessi l'uno all'altro :

- **STUB AS** : per piccole corporazioni.
1 connessione verso altri AS;
non fa da transito tra AS
per il traffico!
- **MULTIHomed AS** : per le aziende/corporazioni più grandi; si hanno PIÙ connessioni verso PIÙ AS :
 - più affidabilità in caso di guasti
 - motivi commerciali: evitare di vedersi imposto un prezzo del singolo operatore
 * NON permettono transito di traffico esterno
- **TRANSIT AS** : i provider (ad esempio Tim); collegano diversi AS fra loro, permettendone lo scambio di TRAFFICO!



• TWO-LEVEL ROUTING :

- **INTRA-AS** : ognuno adotta un proprio routing algorithm nelle reti, indipendentemente dagli altri AS
 - **INTER-AS** : c'è un unico standard → **PROTOCOLLO BGP**
- * I router di gateway vengono detti BORDER ROUTERS (router di bordo); quelli non gateway "INTERNAL ROUTERS".

• INTRA-AS ROUTING :

Sono protocolli conosciuti come **IGP** (Interior Gateway Protocol); i più comuni ed usati sono:

- RIP = Routing Information Protocol
- OSPF = Open Shortest Path First
- IGRP = Interior Gateway Routing Protocol (proprietà della Cisco).

* OSPF è il più usato; vediamo di analizzarlo in breve sia RIP che OSPF.

RIP

- È un protocollo che usa l'algoritmo DISTANCE VECTOR
- È incluso nella distribuzione BSD - Unix di Berkeley del 1982
- DISTANCE METRIC = numero di hops (max = 15 hops) → Praticamente ogni link ha costo 1 (unitario)!

RIP ADVERTISEMENT:

- I DV's sono scambiati con i vicini OGNI 30 SECONDI (PERIODICAMENTE!), attraverso messaggi chiamati "Advertisement".
- In ogni advertisement ci possono essere fino a 25 informazioni di routing del tipo (Destination Subnet, Num. hops to dest).

LINK FAILURE AND RECOVERY:

I messaggi si possono perdere oppure un router o un link si può rompere;

Se non si riceve alcun advertisement da un vicino per 180 secondi (3 minuti = 6 advertisements), lo si reputa guasto:

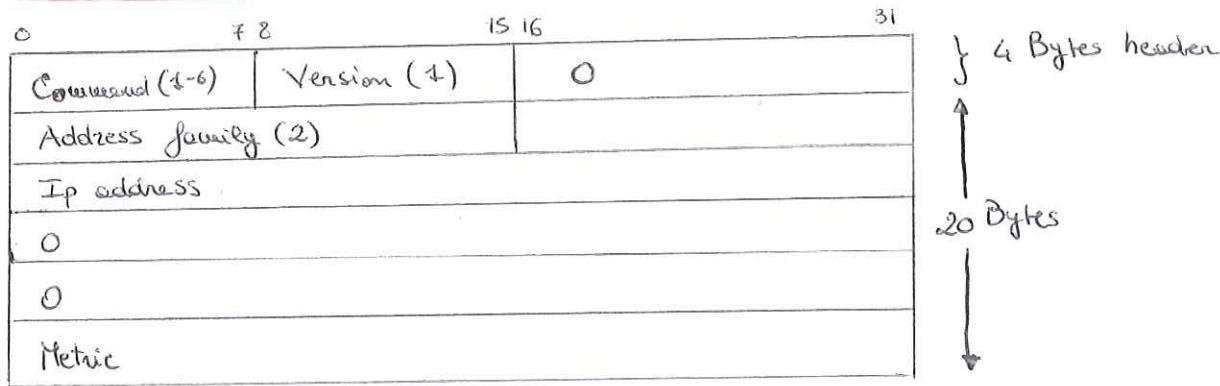
- i percorsi per quel vicino vengono invalidati
- nuovi advertisement aggiornati vengono inviati ai vicini
- i vicini a loro volta inviano nuovi advertisements, se le tabelle di routing sono complete.

- Le informazioni si trasmettono velocemente nell'intera rete
- Vicini usano Poisoned Reverse per prevenire i ping-pong loops (cos = 16 hops)

RIP TABLE PROCESSING:

- La gestione delle tabelle di routing in RIP avviene a LIVELLO APPLICATIVO, tramite processi (dewari di sistema) chiamati route-d.
- Gli advertisements sono inviati in pacchetti UDP → Impossibile!

• RIP MESSAGE:



- COMMAND: 1 = request to send all or parts of the routing table
2 = reply
3-6 ; obsolete or not documented

- ADDRESS FAMILY: 2 = IP Addresses

- METRIC: distance of emitting routers from the specified IP address in number of hops (valid from 1 to 15; 16 = infinity ∞).-

• MESSAGE SIZE:

8 UDP header + 4 RIP header + 20 Bytes x fino a 25 entries = max **512 Bytes**

è il massimo di un datagramme UDP

* 25 entries \rightarrow di solito non abbastanza per un'intera routing table
 \hookrightarrow Spesso servono più messaggi UDP!

• INIZIALIZZAZIONE:

Quando un route-d devoce parte, invia messaggi di richieste RIP speciali su tutte le sue interfacce:

- command = 1 (request)
- un'unica entry con tutti bit a 0 } Richieste delle tabelle di routing complete ai vicini

\rightarrow Questo consente di scoprire quali sono i routers vicini!

OSPF

- Utilizza l'algoritmo **LINK STATE**:
 - LS packet dissemination
 - ogni nodo ha l'intera topologia della rete
 - calcolo del percorso usando Dijkstra
- L'advertisement di OSPF ha 1 entry per ogni router vicino
- Gli advertisements sono disseminati in TUTTO l'AS (trunk flooding)
 - * Sono trasportati direttamente su IP (non pesce per TCP/UDP → No port #)
- FUNZIONI "AVANZATE" (NON IN RIP):
 - SECURITY: Tutti i messaggi OSPF vengono autenticati dai routers, per evitare intrusioni esterne.
 - MULTIPLE SAME-COST PATHS ALLOWED: permette di inviare pacchetti su più percorsi che hanno lo stesso costo, così da avere meno congestione e meno perdite
 - * → I pacchetti possono arrivare non in ordine
 - COSTI DEI LINK SETTABILI: usa i bit **TOS** (Type Of Service) dell'header IP, permette di associare **METRICHE DIVERSE** a seconda dei TOS, così da veicolare il traffico su link differenti.
Utile, ad esempio, per un link settefiltrato usato per traffico di backup ; ottimo per supplire ai guasti sulle linee terrestri.
↳ PERCORSI ALTERNATIVI, MA DI COSTI DIVERSI, con traffico di cui va differenziata l'importanza
 - SUPPORTA IL TRAFFICO MULTICAST: consente di inviare pacchetti a GRUPPI di nodi ; Multicast OSPF (MOSPF),
 - HIERACHICAL OSPF: molto usato, soprattutto in domini ampi.
 - * I costi dei link sono chiamati "METRIC", una metrica è nel range $[0, 2^{16}]$, differente da RIP ↔ Le metrliche possono essere **ASIMMETRICHE**

• LSA (Link State Advertisement) :

Vediamo un esempio di LSA di un router 10.1.1.1 collegato con altri 2 router:

Link State ID :

Advertising Router :

Number of Links :

Description of Link 1 :

Description of Link 2 :

Description of Link 3 :

10.1.1.1	→ RouterID
10.1.1.1	→ RouterID
3	→ 2 routers + se stesso
LinkID = 10.1.1.2	, Metric = 4
LinkID = 10.1.1.3	, Metric = 3
LinkID = 10.1.1.1	, Metric = 0 → Router stesso

- Ogni router mantiene gli LSA ricevuti in un proprio database detto LINK STATE DATABASE:

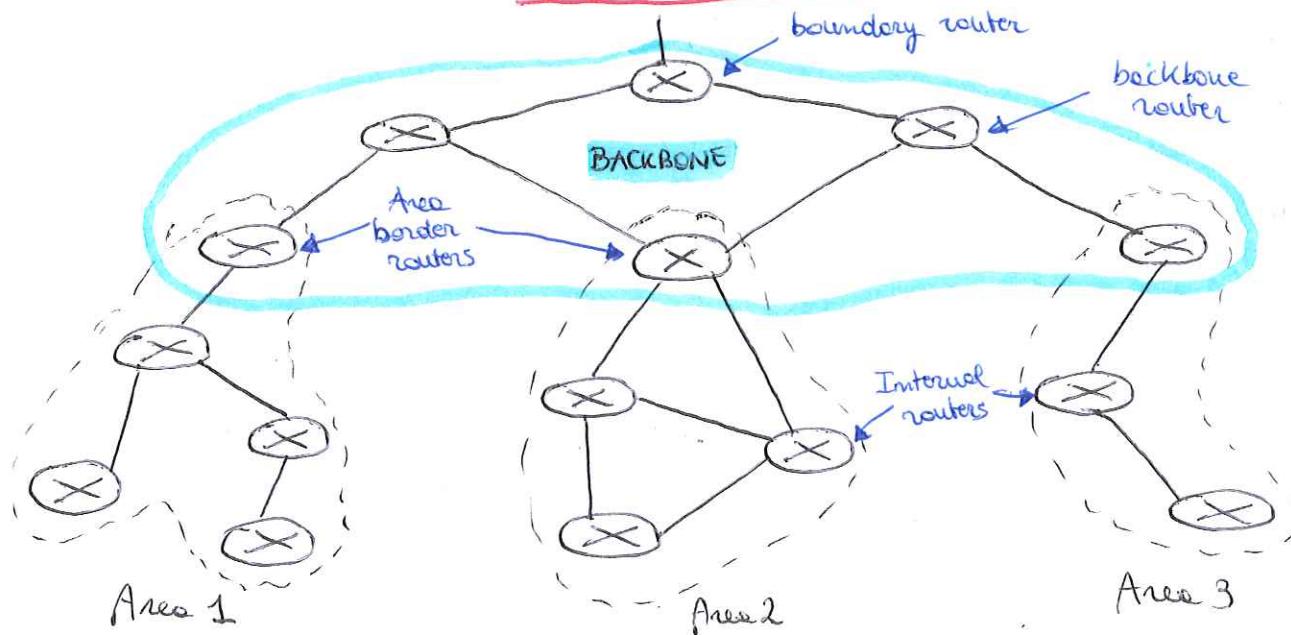
- ogni router ha lo STESO link state database → utile per il debugging
- se nodi vicini si "conoscono" per la 1^a volta, si scambiano i propri link state database
- i link state databases sono SINCRONIZZATI mediante RELIABLE FLOODING: i pacchetti flooded vengono riscontrati tramite "Link State Acknowledgement" (ACK) packets.

* I messaggi OSPF non viaggiano con UDP (né con TCP). OSPF ha il proprio protocol number da inserire nell'header IP: IP protocol number = 89

• DISSEMINATION OF LSA - UPDATE :

- Un router invia e ritrasmette gli LSA solo se le topologie delle reti o i costi dei link cambiano.
- Tuttavia, OGNI 30 MINUTI, un router ritrasmette gli LSA anche se niente è cambiato!

HIERARCHICAL OSPF



- GERARCHIA A 2 LIVELLI: LOCAL AREA + BACKBONE
 - LSA inviati solo nelle local aree (flooding limitato)
 - ogni nodo ha una topologia delle reti dirette dello SHORTEST PATH verso le RETI in altre aree.
- AREA BORDER ROUTERS: "RIASSURGONO" (SUMMARY) le distanze delle proprie reti alle altre reti, e le comunicano agli altri aree border routers
- BACKBONE ROUTERS: eseguono OSPF limitatamente al backbone
- BOUNDARY ROUTERS: si connettono ad altri AS's.

VANTAGGIO DEL SUMMARY: I cambi di link state si propagano solo nelle local aree!
All'esterno, viene propagato solo un Summary LSA!

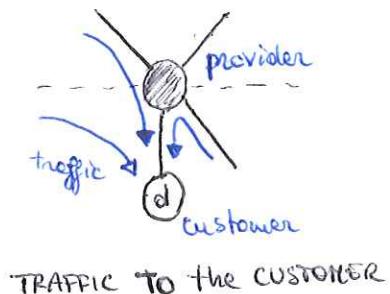
RELAZIONI DI BUSINESS TRA AS:

- AS vicini hanno contratti di business;
- quanto traffico trasportare
 - quali destinazioni raggiungere
 - quanto pagare

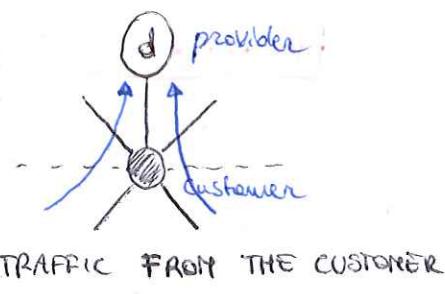
Relazioni di Business comuni sono: Customer-Provider, Peer-Peer, --

CUSTOMER - PROVIDER:

- Il customer deve essere raggiungibile da chiunque → il provider deve assicurare che tutti i vicini possano raggiungere il customer
- Il customer non vuole offrire SERVIZIO DI TRANSITO → No traffico esterno dal provider



TRAFFIC TO THE CUSTOMER

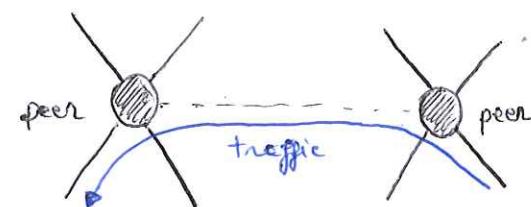


TRAFFIC FROM THE CUSTOMER

PEER-PEER:

I peer si scambiano traffico tra customers;

la relazione è spesso SETTLEMENT-FREE
(non pagano tra loro)



TRAFFIC TO/FROM THE PEER AND ITS CUSTOMERS
(OTHER PEERS)

• AS Structure: TIER-1 PROVIDERS

Sono le due classi gerarchiche di Internet:

- non hanno un upstream provider
- tipicamente hanno un ampio backbone (inter)nazionale
- ce ne sono circa 10-12: AT&T, Sprint, ...

• TIER-2 PROVIDERS:

Forniscono transito di traffico solo DOWNSTREAM verso i customers.

- hanno bisogno di almeno 1 provider tier-1 in upstream
- tipicamente hanno ambito nazionale o regionale (e.g. TIM)
- Includono poche migliaia di AS's

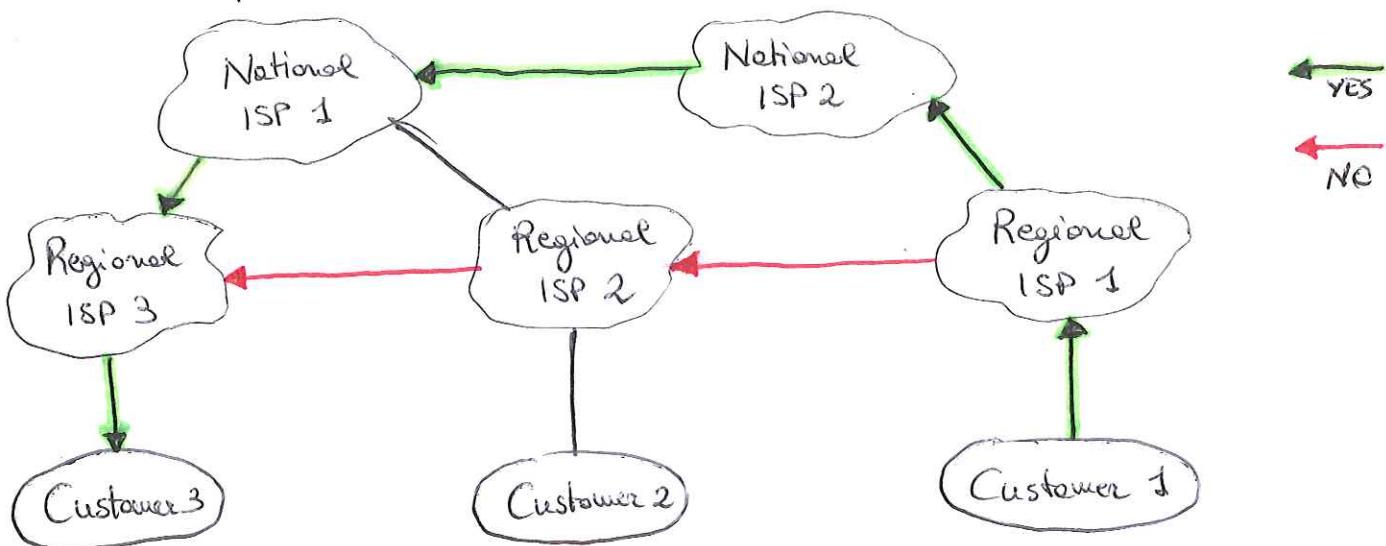
• STUB AS':

- non forniscono SERVIZIO DI TRANSITO
- si connettono agli upstream provider(s)
- Sono la maggior parte degli AS (85-90%)

PATH-VECTOR ROUTING

Lo Shortest-Path Routing visto fin'ora è troppo restrittivo perché:

- tutto il traffico deve passare sugli shortest paths
- tutti i nodi hanno bisogno di una mazzone comune di costo dei Cirkle
- è incompatibile con le relazioni commerciali (no policy)



* L'ISP Regionale 2 non vuole fare da tramite per il traffico di 2 suoi competitor! Vuole trasmettere/ricevere traffico solo da/per i suoi customers!

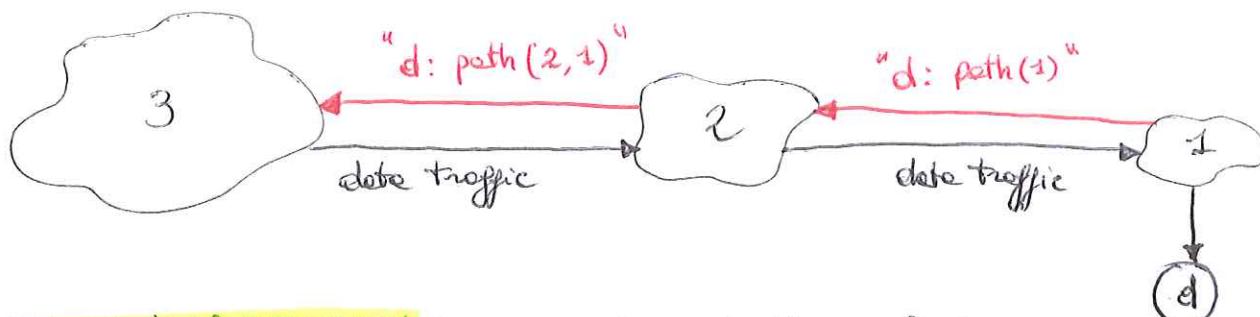
• PATH-VECTOR ROUTING ALGORITHM:

È un'estensione del distance-vector routing:

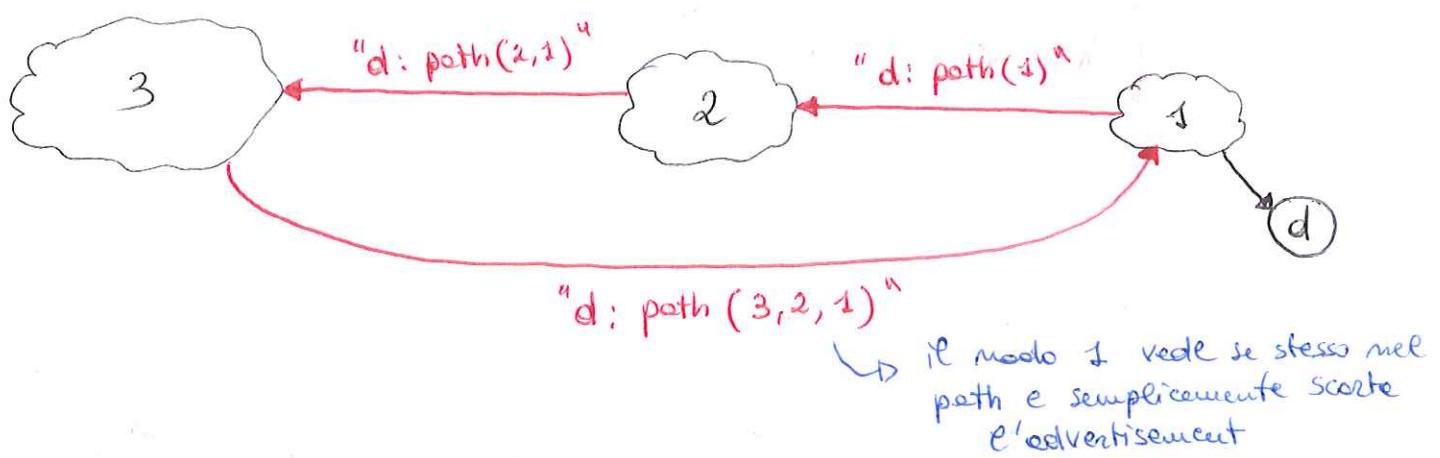
- supporta politiche di routing flessibili (POLICY BASED)

- converge più velocemente, in quanto evita il count-to-infinity problem conoscendo l'intero path

- anziché inviare le distanze metriche (costo) migliore per una destinazione d , si invia l'intero percorso, costituito dagli IP degli AS; negli advertisements:



- **FASTER LOOP DETECTION**: un modo può rilevare facilmente un loop, verificando se il proprio AS number è presente in un path ricevuto; in tal caso, il modo semplicemente scorre tale path;



BGP

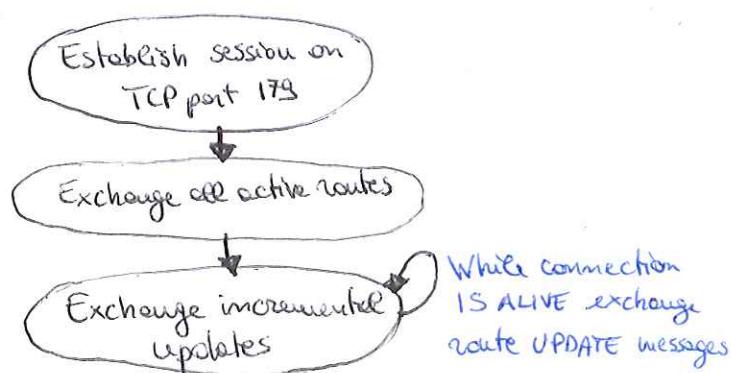
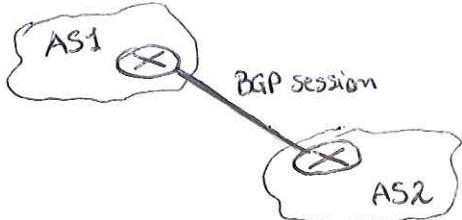
BGP = Border Gateway Protocol → E' uno standard "de facto" come
PROTOCOLLO INTER-AS DI INTERNET

- Sfrutta un algoritmo Path-Vector basato sui PREFISSI delle reti
- E' un routing Policy-BASED basato sui percorsi tra AS
- BGP consente ad ogni AS di:
 - 1) Ottener informazioni sulle raggiungibilità di sottoreti degli AS vicini
 - 2) Proporre le informazioni sulle raggiungibilità ai router interni all'AS
 - 3) Determinare delle "BUONE" ROTTE (non ottime!) verso le sottoreti, basandosi sulle informazioni di raggiungibilità e sulle policy dell'AS.
- Consente alle sottoreti di notificare la propria esistenza al resto di Internet:
"I am here" / "I'm alive".
- BGP Basics:
 - Copie di router (BGP peers) si scambiano le informazioni di routing su una CONNESSIONE TCP SEMI-PERMANENTE, usando il port # = 179 → BGP SESSION
- Ci sono 2 tipi di sessioni BGP:
 - 1) eBGP (external) → tra router di bordo di 2 AS differenti
 - 2) iBGP (interior) → tra router di bordo dello stesso AS, per propagare nell'AS le info di raggiungibilità ottenute
- Quando un AS decide di comunicare un PREFISSO ad un altro AS:
 - * PROMETTE che si farà carico di fare da tramite, quindi di INSTRADARE il traffico verso quel prefisso, tramite il path comunicato.
 - nel comunicarlo, si puo' decidere di AGGREGARE i PREFISSI

* NOTA: se non vuoi fare da tramite, puoi anche decidere di NON COMUNICARO!

↳ POLICY !!!!

BGP OPERATIONS:



• BGP MESSAGES:

OPEN: apre una connessione TCP verso un peer, fornendo l'autenticazione del sender

UPDATE: annuncia nuovi percorsi o revoca vecchi percorsi

KEEPALIVE: mantiene aperta la connessione TCP in sostanza di updates; usato anche per fare da ACK alle richieste di OPEN

NOTIFICATION: riporta errori nel messaggio precedente; usato anche per chiudere la connessione

• INCREMENTAL PROTOCOL:

- Un modo impone PIÙ DI 1 PATH verso una destinazione e:

- gli salva tutti nelle routing table

- applica le policy per selezionare 1 SINGOLA ROTTA ATTIVA

- ... e PUÒ decidere di annunciare le rotte ai suoi vicini

- INCREMENTAL UPDATES → contengono solo le differenze / il cambiamento rispetto all'update precedente

- Announcement → quando si seleziona una nuova rotta attiva, si aggiunge il proprio nome id al path ... e si può comunicare ai vicini

- Withdrawal → se la rotta attiva non è più disponibile, la si revoca comunicando ai vicini con dei withdrawal messages

• BGP Route:

È formato da:

PREFISSO DESTINAZIONE + ATTRIBUTI DI ROUTE

Tra gli attributi, ci sono:

- AS path → e.g. "7018 82"

- Next-Hop IP address → e.g. 12.127.0.121

• BGP Route Selection:

Caso più semplice:

- Shortest AS path → si sceglie la rotta con numero minore di AS da attraversare

- In caso di parità, si può scegliere arbitrariamente

* Tuttavia, BGP è POLICY BASED! → Le policies influenzano la scelta!

Vedremo che ci sono:

- import policies
- export policies

che influenzano le informazioni di routing da salvare e da comunicare ai vicini.

- Abbiamo visto che un router può imporre più rotte verso una destinazione, ma deve scegliere 1. Vediamo alcune regole di scelta:

- 1) valore degli attributi di preferenze locali → policy decision
- 2) shortest AS path
- 3) closest NEXT-HOP router → "hot potato routing"
- 4) criteri aggiornati → **POLICY PROGRAMMABILI !**

* Tipicamente, la metrice per gli attributi di preferenze usata è $\begin{cases} 100 \rightarrow \text{STANDARD} \\ 300 \rightarrow \text{PREFERITO} \end{cases}$

Import Policy and Export Policy:

Le Import Policy vengono applicate quando si riceve un BGP update; invece, le Export Policies quando c'è la possibilità di inviare un Announcement ai vicini.

Import Policy:

- Filtra le rotte non volute dai vicini → e.g. prefissi che non appartengono ai propri customers
- Manipola gli attributi per influenzare le scelte → e.g. assegnare le local preference alle rotte preferite

Export Policy:

- Filtra le rotte che non si vogliono comunicare ai vicini → e.g. non annunciare ad un peer una rete appresa da un altro peer (per non essere sfruttati)
- Manipola gli attributi per controllare cosa gli altri vedono → e.g. far sembrare un path più lungo di quanto effettivamente sia

Perché differenti INTRA-AS e INTER-AS Routing?

Policy:

- Inter-AS: l'utente vuole controllare le rotte del traffico e chi paga per le sue reti
- Intra-AS: singolo datore, quindi non ci bisogna di decidere policy

Scalabilità: il routing gerarchico previene "l'esplosione" delle dimensione delle tabelle e riduce il traffico per gli update

Performance:

- Intra-AS: focus sulle performance
- Inter-AS: policy (business) dominano sulle performance!

