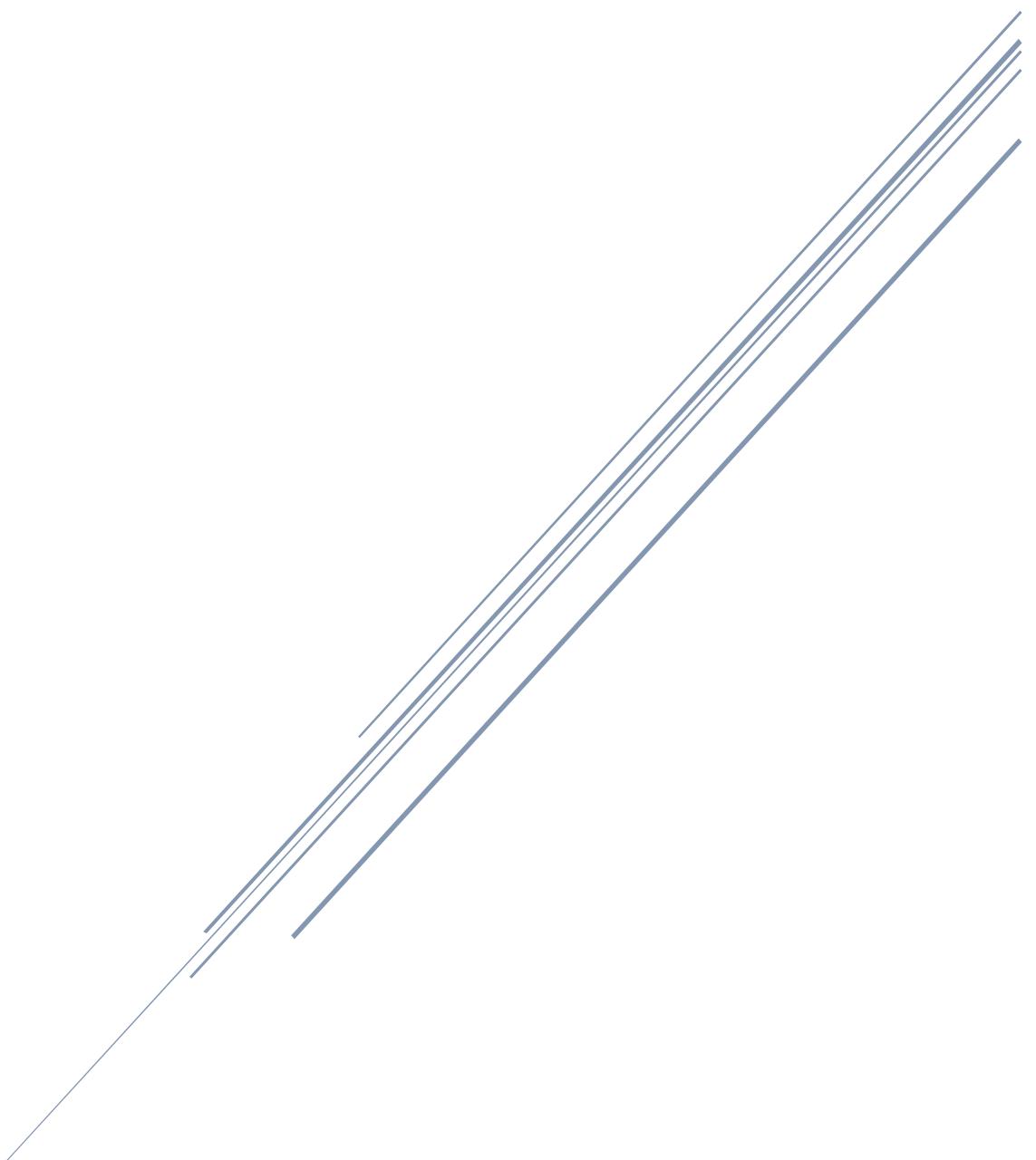


# APPUNTI DI INFRASTRUTTURE DELLE RETI DI CALCOLATORI

Appunti del corso del professor Di Battista nell'anno  
2014/2015



## Appunti di Infrastrutture delle Reti di Calcolatori

Nota importante: questi appunti sono stati presi da me a lezione, e integrati ove necessario con materiale reperito su siti web, libri, etc. Questi appunti non sono assolutamente da ritenere come sostituto di materiale didattico fornito dal professore del corso (libri, dispense etc.), ma come supporto allo studio personale; non sono inoltre né visionati né approvati dal Docente del Corso. Non escludo che siano affetti da errori e/o imprecisioni, e declino ogni responsabilità sull'uso che si farà di questo fascicolo.

## Indice delle lezioni

Lezione di introduzione al corso.....	8
<i>Programma</i> .....	8
<i>Cos'è Netkit</i> .....	8
Il punto di vista delle applicazioni.....	9
<i>Pila ISO-OSI</i> .....	9
<i>Internet Protocol Suite</i> .....	11
<i>Classificazione delle architetture</i> .....	12
Client-server (due livelli) .....	12
Client-server in cascata (tre livelli) .....	12
Peer-to-peer .....	13
Cloud .....	13
<i>Altri criteri di classificazione</i> .....	13
<i>Servizi di rete e livelli di astrazione</i> .....	14
Qualità dei servizi di rete .....	15
<i>Servizi</i> .....	17
Punto di accesso al servizio.....	17
<i>Classificazione delle metriche per la QoS</i> .....	18
Tempo di provisioning .....	18
Gestione dei disservizi.....	18
Metriche tipiche dei servizi di trasporto .....	19
<i>Sistemi di misura della QoS</i> .....	20
Sistemi passivi: ipanema technologies .....	22
<i>Soglie e penali per la QoS</i> .....	24
Progettazione di servizi basati sul web con esigenze stringenti in termini di prestazioni: architetture, modelli e algoritmi .....	25
<i>Obiettivi</i> .....	25
<i>Definizioni preliminari</i> .....	25
<i>Motivazioni</i> .....	25
<i>Semplice richiesta al server</i> .....	26
Problemi e soluzioni .....	26
<i>Sistemi basati su più server</i> .....	27
HTTP request: caso statico .....	27
HTTP request: caso dinamico.....	27
<i>Classificazione di riferimento</i> .....	28
<i>Distribuzione Locale – Web Cluster</i> .....	29
Web cluster .....	29
Web switch .....	30
<i>Distribuzione globale: sistemi Web Distribuiti</i> .....	38
Siti mirror .....	38
Server distribuiti .....	38

Content delivery (distribution) networks.....	44
<i>Akamai</i> .....	44
Redirezione DNS.....	45
Problemi e tecniche generali delle CDN.....	48
URL rewriting .....	49
CDN peering.....	49
Billing e Logging .....	49
Il routing nelle overlay networks .....	50
<i>Reti di overlay</i> .....	50
Primitive .....	50
Efficienza delle operazioni .....	51
Architetture.....	51
<i>DHT</i> .....	54
Memorizzazione diretta e indiretta .....	54
Qualità delle DHT .....	55
Can .....	61
Kademlia .....	65
Architettura di un internet data center .....	67
<i>Sicurezza logica</i> .....	68
<i>Sicurezza fisica</i> .....	68
Socket .....	70
<i>Client</i> .....	70
Client in C .....	72
<i>Server</i> .....	75
Server in C .....	75
Server TCP che fa uso di fork .....	78
<i>Client e server UDP</i> .....	81
Client.....	81
Server .....	83
Tecnologie e metodologie per lo strato di trasporto: affidabilità ed efficienza di TCP .....	85
<i>Instaurazioni di connessioni</i> .....	85
<i>Efficienza</i> .....	88
TCP per servizi interattivi .....	90
<i>Controllo di congestione in TCP</i> .....	92
Finestra di controllo di congestione.....	92
Algoritmo di slow start.....	93
<i>Congestion avoidance</i> .....	95
<i>Timeout e ritrasmissione</i> .....	96
Algoritmo di Karn .....	97
<i>Fast recovery e fast retransmit</i> .....	97
Comprendere il TCP.....	99

<i>TCP onnisciente</i> .....	99
Self-clocking .....	100
<i>TCP in una rete stabile</i> .....	103
<i>TCP reno</i> .....	105
AIMD .....	105
AIAD .....	109
MIMD .....	109
MIAD .....	109
Algoritmi per l'instradamento per l'infrastruttura di una rete fissa .....	111
<i>Qualità degli algoritmi di instradamento</i> .....	111
<i>Tassonomia degli algoritmi</i> .....	111
Algoritmi statici .....	112
Algoritmi dinamici .....	113
<i>Algoritmi distance vector</i> .....	114
Efficacia in condizioni di raggiungibilità .....	117
Lemma delle distanze .....	118
Efficienza degli algoritmi di distance vector .....	120
Algoritmi Link-State-Packet .....	122
Protocolli di routing e la rete internet .....	124
<i>Comunicazioni tra protocolli di routing</i> .....	126
<i>Protocolli IGP ed EGP</i> .....	130
EGP .....	130
IGP .....	130
Bridge: calcolo dello spanning tree .....	133
<i>Fase 1</i> .....	135
<i>Fase 2</i> .....	136
<i>Fase 3</i> .....	136
<i>Fase 4</i> .....	136
<i>Esempio di utilizzo dell'algoritmo</i> .....	137
Software Defined Networks .....	139
<i>Architetture delle SDN e OpenFlow</i> .....	140
OpenFlow .....	141
<i>Messaggi</i> .....	143
Messaggio PacketIn .....	143
Messaggio PacktOut .....	143
Messaggio FlowMod .....	144
Messaggio FlowRemoval .....	145
<i>Interazione con la topologia</i> .....	145
Border Gateway Protocol (BGP) .....	146
<i>Routing interdominio</i> .....	146
Singolo protocollo di instradamento .....	147

Rotte statiche .....	148
Exterior gateway protocol.....	149
<i>Border Gateway Protocol (BGP)</i> .....	151
Aspetti essenziali .....	151
Autonomous system .....	151
Peering .....	152
e-BGP e i-BGP .....	152
<i>Zebra - Quagga</i> .....	154
Comandi di configurazione .....	154
Esempio di peering.....	155
Annuncio .....	156
Policy .....	157
Attributi .....	163
<i>Selezionare la rotta migliore</i> .....	164
Il processo decisionale.....	164
<i>Modifica degli attributi</i> .....	166
<i>Una piccola rete internet</i> .....	167
Classificazione dei customer .....	168
Stub .....	168
Multi-homed stub network: two links to the same isp .....	171
Multi-homed stub network: two links to two different isp.....	175
Piccola rete internet .....	179
<i>Scalabilità di BGP</i> .....	182
<i>Route refresh</i> .....	182
<i>Next-Hop</i> .....	182
<i>iBGP peering</i> .....	183
<i>iBGP mesh</i> .....	183
<i>Migrazione</i> .....	186
<i>BGP Communities</i> .....	187
no-export, esempio di utilizzo.....	189
no-peer, esempio di utilizzo .....	190
<i>Il grafo di internet</i> .....	191
<i>Cos'è il grafo di internet</i> .....	191
<i>Sorgenti informative</i> .....	191
Interner Routing Registry (IRR) .....	192
Looking glasses.....	193
Route views.....	196
<i>Damping algorithm</i> .....	199
<i>Ancora sorgenti informative</i> .....	201
Routing information service .....	201
<i>Analisi dei dati di Internet</i> .....	203

<i>Crescita delle tabelle di instradamento</i> .....	203
<i>Tempeste di BGP update</i> .....	206
<i>Esaурimento degli indirizzi IPv4</i> .....	206
<i>Variazioni sulla gerarchia di internet</i> .....	207
<i>Robustezza dell'ecosistema di internet</i> .....	209
<b>Stabilità di un BGP</b> .....	211
<i>Disagree</i> .....	212
<i>Aspetti computazionali della stabilità di BGP (SPP, Stable Path Problem)</i> .....	215
<b>Reti locali virtuali (VLAN)</b> .....	218
<b>VLAN su un solo switch</b> .....	218
Filtering database .....	220
VLAN simmetriche e asimmetriche .....	220
<b>Connessioni virtuali asimmetriche</b> .....	221
<b>VLAN su reti con più switch</b> .....	223
Tag IEEE 802.1Q.....	225
La LAN di un edificio .....	226
<b>MPLS VPN</b> .....	227
<b>Motivazioni</b> .....	227
<b>Vincoli</b> .....	228
<b>MPLS</b> .....	228
How-to .....	230
LDP .....	232
Riferimenti tecnologici .....	232
<b>VPN</b> .....	236
<b>Route target</b> .....	236
<b>NAT (Network Address Translation)</b> .....	239
<i>Private address allocation</i> .....	239
<i>NAT (Network Address Translation)</i> .....	240
Binding degli indirizzi .....	241
<i>PAT (Port Address Translation)</i> .....	242
Problemi di nat e pat .....	243
<b>Il protocollo ipv6</b> .....	244
<i>Ipv6</i> .....	244
Header ipv4 .....	244
Header ipv6 .....	245
Gestione delle opzioni .....	246
<i>Indirizzi ipv6</i> .....	249
Tipi di indirizzi .....	249
Indirizzi per ogni host .....	256
<i>Icmpv6</i> .....	257

Protocollo e tipi di pacchetto.....	257
Path MTU Discovery .....	258
<i>Neighbor discovery</i> .....	259
Redirection.....	259
Neighbor solicitation .....	259
Neighbor unreachability detection .....	261
Neighbor discovery.....	262
Duplicate address detection .....	263
<i>Procedura di autoconfigurazione</i> .....	263
Esempio di autoconfigurazione.....	263
<i>Renumbering</i> .....	265
<i>Gestione avanzata degli indirizzi</i> .....	265
Router renumbering .....	265
Configurazione stateful .....	265
<i>Transizione ipv4-ipv6 e meccanismi di compatibilità</i> .....	265
Il processo di transizione .....	265
Meccanismi basati su tunnel .....	267
Meccanismi basati su traduzione di protocollo .....	272
<i>Source address selection multihoming</i> .....	275
Source address selection .....	275
Multihoming .....	276

## Lezione di introduzione al corso

### Programma

Gli argomenti trattati dal corso sono:

- il rapporto tra le applicazioni e la rete;
- l'interfaccia fra il livello applicativo ed il livello di trasporto;
- TCP e il controllo di congestione;
- routing intradominio;
- routing interdominio;
- software defined networks;
- ipv6;
- tecniche di virtualizzazione per le reti.

In particolare *Software Defined Networks* è un approccio al computer networking che permette agli amministratori di rete di gestire servizi di rete attraverso l'astrazione di funzionalità di livello inferiore. Questo viene fatto grazie al disaccoppiamento del sistema che prende le decisioni su dove il traffico viene inviato (il piano di controllo o control plane) dai sistemi sottostanti che inoltrano il traffico verso la destinazione selezionata (il piano dati o data plane).

Un'altra definizione viene data da *Telecom Italia*:

---

I paradigma SDN (Software Defined Networking), secondo l'accezione più diffusa, propone l'ambiziosa visione di rendere i nodi di rete (ad es. router e switch) programmabili, introducendo opportuni livelli di astrazione, ai quali accedere attraverso l'uso di interfacce di controllo (API). Nell'ambito di questo paradigma, assume particolare rilievo anche il concetto di virtualizzazione di rete, ovvero l'idea di creare delle partizioni virtuali dell'infrastruttura di rete fisica, in modo da permettere a più istanze di controllo e le rispettive applicazioni di utilizzare le partizioni assegnate: questo permetterebbe la coesistenza di più reti virtuali, completamente isolate, che insistono sulla medesima infrastruttura hardware.

La centralizzazione logica del controllo, potrebbe permettere di attuare più facilmente azioni di configurazione e ottimizzazione delle risorse di rete. L'adozione di questo paradigma potrebbe abilitare lo sviluppo di nuovi ecosistemi.

---

### Cos'è Netkit

Netkit è una piattaforma che consente di emulare il comportamento della rete.

## Il punto di vista delle applicazioni

### Pila ISO-OSI



Figura 1 - Schema concettuale di dialogo tra elaboratori nel modello OSI

- **Fisico:**
  - Si interfaccia direttamente con i mezzi fisici di trasmissione.
  - Offre allo strato superiore una comunicazione indipendente dal particolare mezzo trasmittivo.
  - Servizi forniti allo strato di collegamento: gestione della connessione fisica, identificazione della connessione fisica, trasmissione delle unità dati, consegna in sequenza delle unità dati, notificazione di malfunzionamenti.
  - Qualità dei servizi tasso d'errore, disponibilità, frequenza di cifra, ritardo di trasferimento.
- **Data link (collegamento):**
  - Condividere lo stesso fisico e trasmettono un pacchetto alla volta.
  - Obiettivo: fronteggiare i malfunzionamenti dello strato fisico mediante la rivelazione e correzione degli errori.
  - Servizi offerti allo strato di rete: trasferimento di informazioni senza vincoli di formato, codice o contenuto, e selezione di una certa qualità di servizio.
  - Utilizzo di due code nelle due direzioni.
- **Rete:**
  - Si occupa della trasmissione di un pacchetto da una macchina posta sulla rete ad una macchina destinazione (posta ovunque sulla rete). Per fare questo bisogna introdurre uno schema di indirizzamento che individua qual'è la destinazione; si applica un meccanismo di rimbalzo (una sorta di "passaparola" finché non arriva a destinazione).
  - Conosce la tipologia della rete (instradamento).
  - Se connesso, instaura, mantiene e rilascia le connessioni di reti.

- Il 3-servizio consente di trasferire le informazioni da un estremo all'altro, selezionare (non sempre è possibile) una certa qualità in termini di tempo di attraversamento, disponibilità, ecc.
- Commutazione: circuito, pacchetto a datagramma, circuito virtuale.

➤ Trasporto:

- Trasmissione di un flusso tra due processi posti ovunque nella rete.
- Lo schema di indirizzamento del livello di trasporto, non deve individuare solo la macchina, ma anche il processo: mentre a livello di rete va da indirizzo IP a indirizzo IP, al livello di trasporto va da porta sorgente a porta destinazione.
- Colma defezioni e fluttuazioni del grado di servizio delle connessioni di rete ed ottimizza l'uso della rete dal punto di vista dei costi.
- È il primo strato estremo-estremo (risiede solo nei nodi terminali).
- Servizi offerti allo strato 5: instaurazione di una connessione, trasferimento dati e gestione della connessione, rilascio.
- Sincronizzazione tra i due sistemi per mezzo di conferma.

➤ Sessione:

- L'utente vuole avere una sessione di comunicazione, mentre la rete gli propone una rete di connessione al livello di trasporto. Rende indipendente l'utente dalla connessione ai livelli più bassi.
- Sincronizzazione del dialogo tra due processi.

➤ Presentazione:

- Traduzione nel linguaggio locale della macchina.
- Scambio di messaggi indipendentemente dalla sintassi della trasmissione.

➤ Applicazione:

- L'applicazione vera e propria utilizzata dall'utente.
- Mezzo per accedere alla rete per un processo applicativo.

## Internet Protocol Suite

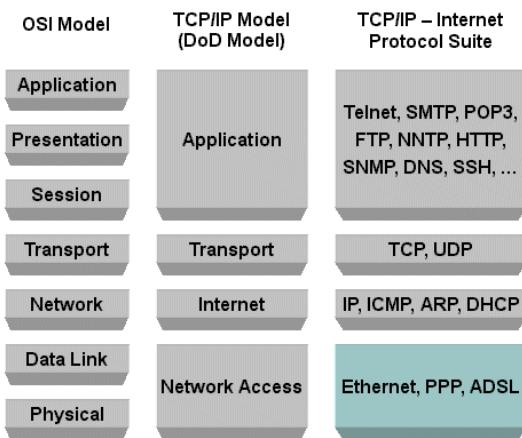


Figura 2 - Modello Internet Protocol Suite

Internet Protocol Suite è un insieme di protocolli di rete su cui si basa il funzionamento della rete Internet. Tale suite può essere descritta per analogia con il modello OSI, che descrive i livelli (layer) della pila di protocolli. In una pila di protocolli ogni livello risolve una serie di problemi che riguardano la trasmissione di dati e fornisce un ben definito servizio ai livelli più alti. I livelli più alti sono logicamente più vicini all'utente e funzionano con dati più astratti lasciando ai livelli più bassi il compito di tradurre i dati in forme mediante le quali possono essere fisicamente manipolati e trasmessi infine sul canale di comunicazione. Il modello Internet è stato prodotto come una soluzione ad un problema ingegneristico pratico in quanto si è trattato di aggiungere via via strati protocollari all'architettura di rete delle reti locali per ottenere un'interconnessione efficiente ed affidabile. Il modello OSI, in un altro senso, invece è stato l'approccio più teorico-deduttivo ed è stato anche prodotto nel più vecchio modello di rete.

Ricapitolando:

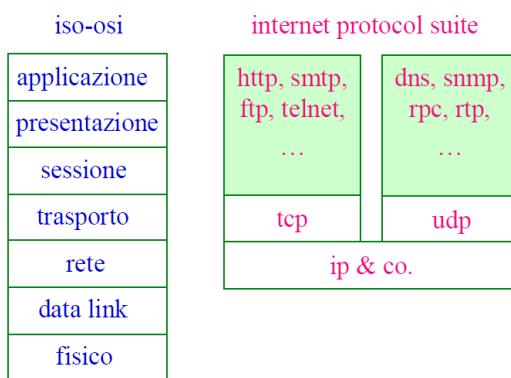


Figura 3 - Schema della Pila ISO-OSI e dell'Internet Protocol Suite

C'è stata una progressiva evoluzione dei servizi tradizionali, generalmente offerti con filosofia best effort, in servizi più sofisticati e con stringenti esigenze di performance. Viene sempre utilizzata più banda. Si pretendono reti efficienti e veloci per servizi come:

- consultazione di pagine web;
- querying di repositories online;
- streaming audio e video;
- file download e file sharing;
- giochi online;
- software-as-a-service (SaaS);
- social networking.

## Classificazione delle architetture

Per far fronte a questa tipologia di servizi, si può fare una classificazione delle architetture a più livelli:

- client-server (due livelli);
- client-server in cascata (tre livelli);
- peer-to-peer;
- cloud.

### Client-server (due livelli)

È la più semplice: sono presenti due attori, l'utente e il fornitore della risorsa, cioè il detenente della risorsa e colui che la richiede (es: posta elettronica, risoluzione dei nomi DNS, FTP, http). Ha come vantaggi la facilità dell'implementazione e della gestione della rete; viceversa, gli svantaggi stanno nella poca resistenza ai guasti e la poca scalabilità a causa dell'unico server.

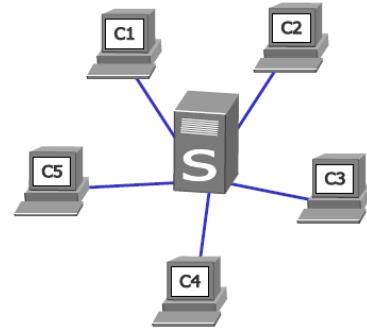


Figura 4 - Architettura Client-Server

### Client-server in cascata (tre livelli)

Si usa una tipologia a tre livelli per sistemi più complessi. Gli attori in questo caso sono tre: l'utente delle risorse (il sistema con cui interagiamo, per esempio il browser), un punto di accesso alla risorsa che intermedia tra l'utente e il fornitore (per esempio il web server), e il fornitore stesso. Questo punto di accesso è facilmente accessibile (24h), gradevole ed usabile; inoltre è sicuro, cioè in grado di selezionare tra più fornitori, e interagire con diversi di essi. C'è sempre il problema della scalabilità.

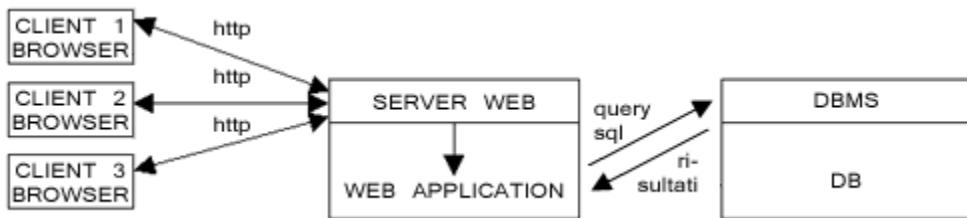


Figura 5 - Architettura Client-Server a tre livelli

### Peer-to-peer

Questo sistema è in fase calante. Ha una sola tipologia di attore che fa da utente e da server. I peer sono nodi equivalenti e paritari che possono fungere sia da cliente che da servente verso gli altri nodi terminali (*host*) della rete. Un insieme di peer costituisce una rete di connessione virtuale che permette lo scambio di dati (*overlay network*). La rete non è stabile (è transiente), quindi non si ha una risorsa continua, ma si abbattono i costi dovuti all'acquisto/gestione del server ed alla scalabilità. Garantiscono inoltre una velocità maggiore in quanto posso prelevare la medesima informazioni a frammenti (ossia prendo un pezzo da più peer).

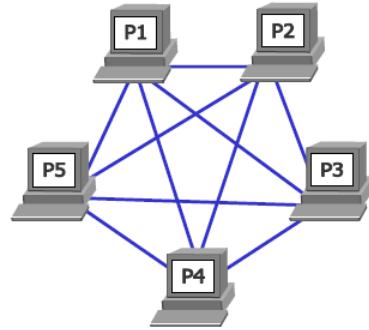


Figura 6 - Architettura Peer-to-Peer

### Cloud

Gli attori sono tre: client, cloud service provider, e server di vario tipo, spesso virtuali, spesso in cascata. I vantaggi sono la scalabilità ed economicità. I problemi però sono due: lo storage, cioè la criticità della privatezza, e la limitata affidabilità. I servizi offerti sono vari, come storage e backup di dati, accesso a piattaforme virtuali remote, accesso a programmi di office automation come servizi, storage, identity, payment, live chat, search, integration, mapping.

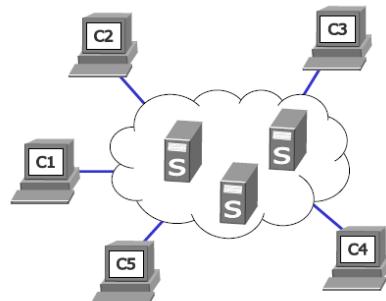


Figura 7 - Architettura Cloud

### Altri criteri di classificazione

- *Pull and push technology*: nel pull il client “cerca” il server, mentre nel push il server “cerca” il client.
- Livelli di replicazione e distribuzione: *proxy* e *caching*.
- Bilanciamento del carico tra client e server o tra più server (lavoro distribuito).

## Servizi di rete e livelli di astrazione

I servizi di rete sono accessibili al programmatore tramite delle librerie software istallate sulle macchine. I programmatori di applicazioni utilizzano API (*application program interface*) per afferire a tali servizi. Diverse librerie modellano i servizi di rete a diversi livelli di astrazione: alcune librerie consentono di controllare il singolo byte inviato, ma non offrono servizi più sofisticati, mentre altre librerie offrono servizi più astratti, confezionando esse stesse i pacchetti dei più comuni protocolli del livello di applicazione.

È previsto un supporto dei linguaggi al trasferimento dei messaggi: nei linguaggi C e C++ tramite la libreria socket, che consente la trasmissione di byte tramite connessioni TCP o invii UDP (il programmatore compone e trasmette i messaggi nei protocolli di applicazione come per esempio HTTP e FTP); per il linguaggio Java, le API Java (anch'esse chiamate socket) consentono trasferimenti di tipi più complessi (come oggetti serializzati qualsiasi, anche non atomici) e offrono primitive per protocolli applicativi (per esempio primitive HTTP).

È inoltre previsto un supporto dei linguaggi all'invocazione di procedure remote: C e C++ supportano RCP (*remote procedure call*), mentre in Java supporta RMI (*remote method invocation*).

La rete può essere vista come popolata da oggetti in base alle tecnologie utilizzate. *Distributed objects* sono moduli software progettati per lavorare insieme che risiedono su piattaforme diverse, connesse tramite una rete. Esempi di tecnologie per distributed object computing sono *common object request broker architecture* (CORBA), che consente di vedere applicazioni scritte in linguaggi diversi, su piattaforme diverse, in modo unificato, come oggetti (dati + metodi) invocabili in vari linguaggi, *distributed component object model* (DCOM) (analogo di CORBA in ambiente Microsoft) e *ddobjects*, un framework per *distributed objects* che usa *Borland Delphi*.

## Qualità dei servizi di rete

Il web sta diventando ogni giorno di più fonte di servizi critici quali l'home banking e dinamici quali ad esempio i quotidiani online, questo genere di servizi richiede alla rete un'affidabilità ed una sicurezza prima non necessari, più in generale richiede un incremento della QoS (Quality of Service).

Le *Quality of Service* sono le tecniche e misure atte a garantire prestazioni predibibili dei servizi di rete. Alcune parole fondamentali sono:

- *Servizio*: offerto da un fornitore ad un fruitore;
- *Metriche*: misurano la qualità del servizio offerto;
- *Specifiche della QoS*: pone delle soglie ai valori delle metriche;
- *Sistema di misura*: misura la qualità del servizio e la riscontra con le soglie;
- *Hit*: richiesta di un singolo contenuto da parte di un client.
- *Page request*: richiesta di un pagina, composta solitamente da diversi hit.
- *Session*: sequenza di page request consecutive da parte dello stesso client.

L'idea di QoS è un'idea che nel mondo delle reti non è nuova, e non nasce con il web.

La qualità del servizio in questo ambito si divide in:

- *Quality of Network Service (QoNS)*: dipendente dai tempi di latenza, dal numero di pacchetti persi e dalla disponibilità del servizio.
- *Quality of Web Service (QoWS)*: dipendente dalla disponibilità del servizio (il sistema è accessibile per l'X% del tempo) e dall'efficienza del servizio (una percentuale X% delle richieste è servita in un tempo inferiore ad un soglia Y).

Salendo di livello abbiamo:

- *Quality of Experience*: misura soggettiva della soddisfazione dell'utente ad un servizio fruito.  
Questa comprende:
  - *Quality of Content*: misura soggettiva del livello di apprezzamento dei contenuti e delle risorse offerte.
  - *Quality of Service*: misura oggettiva della capacità della rete di rendere fruibile il servizio.

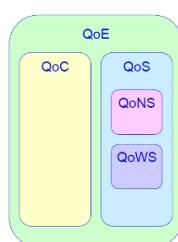


Figura 8 - Schema riassuntivo delle varie QoS

Quali sono i parametri tipici di QoNS (*Quality of Network Service*) nel caso delle reti?

- *Tempi di latenza*: sono i tempi di attesa di un pacchetto. Per essere un po' più precisi, che tipi di misure sono ragionevoli?
- *Numero di pacchetti scartati*: si guarda più che altro alla *percentuale* di pacchetti scartati nell'unità di tempo;
- *Tempo di servizio*: tempo in cui il servizio è effettivamente disponibile (in termini percentuali).

Effettivamente se guardiamo un contratto di servizi di livello 3, firmato da un'azienda qualunque, non mancano mai le caratteristiche di QoNS, espresse tipicamente in questo modo: tempo di latenza al più pari a 60 ms, numero di pacchetti scartati al più pari allo 0,01%, tempo di servizio pari al 99,9% del tempo.

Nel contratto per la fruizione di un servizio tra fornitore e fruitore è sempre presente lo SLA (*Service Level Agreement*): è un accordo commerciale tra le due parti dove viene descritta la qualità attesa del servizio offerto e vengono stabilite le penali per il fornitore al superamento delle soglie stabilite, cioè se il servizio non rispetta le condizioni concordate nel contratto (ci sono naturalmente delle unità di misura per verificare la qualità del servizio). Contiene:

- definizione rigorosa dei servizi offerti;
- definizione formale delle metriche adottate;
- specifica della QoS (soglie);
- penali proporzionali alla violazione delle soglie;
- descrizione del sistema di misura;
- reportistica periodica.

Che cosa succede per il web? Si fa QoS anche per il web?

Si parla in questo caso di QoWS (*Quality of Web Service*). Anche qui di ogni servizio si sceglie una metrica e una soglia. Come si misura? Nel caso di servizi di questo genere verificare che la media dei valori osservati sia inferiore ad una soglia è poco significativo. Infatti questo nel caso del web paga poco: non importa sapere questa media se poi ho degli scostamenti significativi. Di norma, per essere più stringenti si può verificare che tutti i valori osservati siano inferiori alla soglia, o che lo siano il 90%.

Gli SLA per il web sono fatti in questo modo:

- disponibilità:
  - service level agreement: il sistema funziona per X% del tempo:
    - se X=99% non funziona per 7,2 ore al mese (h24\*365);
    - se X=99,9% non funziona per 43 minuti al mese;

- X=99,999% non funziona per 26 secondi al mese;
  - attenzione alla manutenzione;
- efficienza:
  - X% delle richieste (tutti o alcuni tipi) ha un tempo di risposta inferiore a Y secondi:
    - il 95% delle richieste ha un tempo di risposta inferiore a 4 secondi;

## Servizi

Che cosa succede veramente sul mercato? Quando sono diffusi e articolati i contratti che prevedono QoNS e QoWS? Nessuna azienda accetta di sottoscrivere SLA di rete poco articolati; per il web gli SLA di rete articolati sono una pratica poco diffusa poiché il provider di rete, quando offre il servizio di interconnessione di rete si basa su un'infrastruttura, proprietaria, di cui ha il pieno controllo. Quando offri invece un servizio basato sul web i fattori in gioco sono tantissimi, e quello che succede e che chi offre in generale applicazioni distribuite controlla solo alcune componenti architettoniche. Facciamo un esempio concreto: Telecom Italia offre servizi web based in data center di primissima qualità. Se ti deve offrire un servizio con SLA web base end-to-end qualche problema c'è, poiché la latenza di una pagina dipende anche da quello che c'è in mezzo, tra il datacenter e casa. I servizi distribuiti di questo genere sono un collage di pezzi. Si pensi alla componente DNS: la risoluzione di www.servizio.it passa attraverso un'infrastruttura DNS che è sotto la responsabilità di Telecom Italia soltanto per una minima parte. Offrire un servizio profondamente articolato di SLA per il web quindi è molto difficile. Sul web si prendono delle precauzioni sulla SLA: viene caratterizzata solo per la parte di responsabilità del provider.

Le tipologie di servizi offerti sono:

- trasporto:
  - connettività tra le sedi del cliente;
  - accesso ad internet;
- telefonia VoIP (voice over IP);
- sicurezza;
- hosting/housing di server.

## Punto di accesso al servizio

I servizi vengono modellati tramite la definizione del punto di accesso al servizio (PAS) attraverso cui vengono fruitti: il PAS costituisce la cesura tra le responsabilità del fornitore e quelle del fruitore. Alcuni esempi di PAS sono:

- *Servizio di trasporto always-on*: interfaccia interna del default gateway verso la quale le macchine inviano (e dalla quale ricevono) i pacchetti. Il default gateway può essere di proprietà del fruitore o del fornitore; alcuni fornitori però si rifiutano di erogare servizi con router del fruitore per

problemi di sicurezza, perché il fruitore potrebbe manometterlo in qualche modo e in quel caso sarebbero difficili da stabilire le panali dello SLA; altri fornitori preferiscono il rischio, perché offrire anche il default gateway costa di più.

- *Servizio di trasporto WiFi*: access point a cui la piattaforma dell'utente si connette.
- *Servizio di telefonia VoIP*: apparecchio telefonico sul quale l'utente riceve una chiamata (oppure digita il numero del destinatario della chiamata).

## Classificazione delle metriche per la QoS

- Metriche generali:
  - *Tempo di provisioning*: tempo necessario al fornitore per installare, modificare o integrare il servizio richiesto dal fruitore. Una volta che si fa una richiesta, si deve essere reattivi rispetto alle richieste e alle modifiche.
  - Gestione e qualità dei disservizi.
- *Metriche specifiche del servizio*: misurano le prestazioni del servizio in questione (come il packet loss per i servizi di trasporto).

### Tempo di provisioning

È il tempo intercorrente tra la richiesta di un servizio e la sua effettiva disponibilità; ecco alcune:

- tipicamente espresso in giorni solari;
- provisioning per la prima attivazione a partire dalla data di firma del contratto o dalla data di consegna prevista;
- provisioning per successive modifiche di configurazione dal momento della richiesta al momento dell'implementazione della variazione.

### Gestione dei disservizi

Disservizio è l'indisponibilità (totale o parziale) del servizio e può essere:

- *Bloccante*: il servizio non può essere fruito (non accedo alla rete).
- *Non bloccante*: il servizio viene fruito con difficoltà.
- *Anomalia*: il servizio non corrisponde alla descrizione prevista dal contratto.
- *Disponibilità*: percentuale del tempo durante il quale il servizio non è affetto da disservizi bloccanti. Quanto tempo realmente il servizio è disponibile (è considerato disponibile anche in caso di disservizio non bloccante o anomalia)
- *Tempo di ripristino*: dopo quanto tempo il servizio è ripristinato. Parte dall'istante nel quale si concorda che ci sia un disservizio.
- *Ripetitività dei guasti*: i guasti non si dovrebbero ripresentare in modo ripetuto.

### Metriche tipiche dei servizi di trasporto

- *Packet loss*: percentuale dei pacchetti trasmessi che non arrivano a destinazione. Le cause sono:
  - malfunzionamenti dei mezzi fisici: linee “errorate”, cioè che presentano una probabilità significativa di errore sul bit;
  - congestione: apparati di rete che non hanno sufficienti risorse per gestire il volume di pacchetti entranti o esaurimento della memoria dedicata alle code di uscita delle interfacce;
  - transitori: configurazioni temporanee ed incoerenti della rete dovuti alla convergenza di protocolli di routing (i pacchetti vengono scartati per azzeramento del TTL (Time To Live)). Ad esempio, se ci sono cambiamenti della rete, prima della riconfigurazione delle tabelle di istradamento ci potrebbero essere trasferimenti errati di alcuni pacchetti.

È inutile lavorare sull'aumento della memoria di coda sul packet loss perché se il pacchetto si trova tanto in coda, per il ricevitore potrebbe essersi perso, e quindi può richiedere comunque la ritrasmissione. Bisogna invece lavorare sui transitori.

- *Delay*:
  - *one-way-delay*: tempo che intercorre tra la spedizione di un pacchetto e la sua ricezione;
  - *round-trip-delay*: tempo che intercorre tra la spedizione di un pacchetto e la ricezione del corrispondente riscontro.

Cause:

- ritardo di inoltro: tempo che intercorre tra la ricezione di un pacchetto da parte di un apparato ed il suo accodamento sull'interfaccia di uscita (tempo necessario per la richiesta di inoltro);
- ritardo di accodamento: tempo in cui il pacchetto rimane in coda su una interfaccia in attesa della sua trasmissione;
- ritardo di immissione: tempo necessario a “scrivere” il pacchetto sul mezzo trasmisivo; dipende dalla dimensione del pacchetto e dal bit rate della tecnologia e può essere diminuito acquistando più risorse;
- ritardo di propagazione: tempo che impiega il segnale a coprire la distanza tra sorgente e destinazione; dipende dalla velocità della luce e non può essere diminuito in alcun modo;
- ritardo di ritrasmissione: quando sono coinvolte tecnologie di livello 2 che prevedono riscontri e ritrasmissioni; tempo impiegato a rilevare la perdita di un frame e provvedere alla sua ritrasmissione.

Per migliorare il delay bisogna lavorare per ridurre il ritardo di immissione.

- *Jitter*: variazione del ritardo tra due pacchetti successivi.

Cause:

- cambiamenti di routing;
- ritardi di ritrasmissione;
- accodamento dei pacchetti a flussi eterogenei.

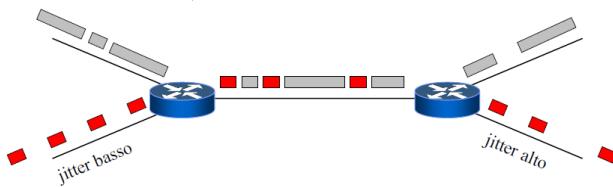


Figura 9 - Rappresentazione dello Jitter

- Banda garantita: Il numero di bit al secondo garantiti dalla connessione (difficile da misurare, infatti è raramente oggetto di misure).

## Sistemi di misura della QoS

Il fornitore del servizio generalmente si impegna ad effettuare misure costanti della QoS per verificare la qualità dei sistemi che possiede: il fruitore può, a sua discrezione, eseguire misure indipendenti della QoS. I risultati delle misure vengono comunicati dal fornitore tramite report periodici: per esempio, report mensili tecnici + report trimestrali “contrattuali”, cioè contenenti anche il calcolo delle penali. Il sistema e le modalità di misura sono negoziate tra fornitore e fruitore e descritte nello SLA: vengono individuati anche gli apparati coinvolti (per esempio, tutte le misure avvengono dal punto di accesso al servizio posto nella rete del fruitore ad uno specifico router di frontiera del fornitore). Il fruitore può chiedere gli strumenti e metodi usati dal fornitore per fare le misurazioni e controllare se l’infrastruttura è gestita in modo corretto.

Tipi di misure:

- Misure attive: il sistema di misura fruisce del servizio per rilevarne la qualità; la misura deteriora la qualità del servizio.
- Misure passive: la qualità del servizio viene dedotta osservandone la fruizione; la misura è disponibile solamente quando il servizio viene fruito.

Si possono classificare in:

- ✓ Misure intrusive: prevedono un uso della rete da parte del sistema di misura per lo scambio di informazioni di controllo.
- ✓ Misure non intrusive: la rete non viene utilizzata neanche per trasportare informazioni di controllo.

Le misure possono essere anche divise in:

- Misure campionate: prendono in considerazione solo un sottoinsieme delle richieste di servizio. Per esempio solo le richieste generate dal sistema stesso di misura (se attivo) e non tutte le richieste di servizio vengono utilizzate per dedurre la qualità del servizio (se il sistema di misura è passivo).
- Misure non campionate: prendono in considerazione la totalità delle richieste di servizio.

I sistemi di misura descritti usualmente negli SLA sono attivi, intrusivi e campionati: viene definita una tipologia di fruizione del servizio (esempio per un servizio di trasporto: invio di un pacchetto di ping di 100 KB ogni 10 secondi verso un target specifico). Viene definito un intervallo di rilevazione: all'interno di ogni intervallo di rilevazione le misure vengono aggregate e confrontate con le soglie.

I sistemi di misura per servizi di trasporto possono essere:

- sistemi attivi, intrusivi, campionati;
- sistemi passivi, intrusivi, campionati;
- sistemi passivi, non intrusivi, non campionati (basati su comunicazione fuori banda, replicazione, ed economicamente onerosi).

Altre definizioni di Jitter:

- Definito come la deviazione standard del one-way delay di trasferimento dei pacchetti (radice quadrata della varianza, dove la varianza è la media dei quadrati degli scostamenti dalla media).
- *Inter-packet delay variation*: definito nell'rfc 3393, la media della differenza dei tempi di interarrivo (denotato  $t_i$  il tempo necessario al pacchetto  $i$  per arrivare a destinazione, viene calcolata la media dei  $|t_{i+1} - t_i|$ ).

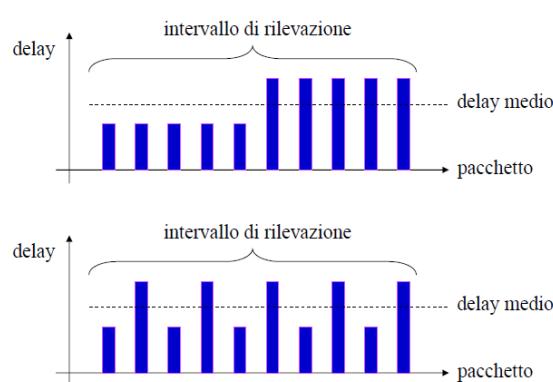


Figura 10 - Misure del jitter a confronto

Queste due trasmissioni hanno lo stesso jitter matematico ma diversa inter-packet delay variation:

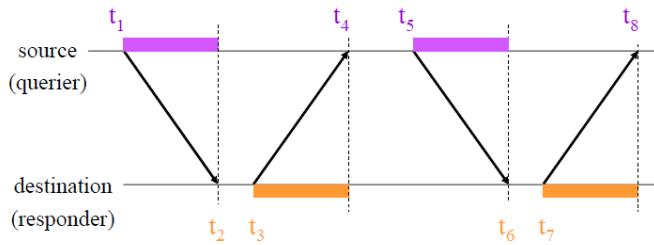


Figura 11 - Esempio di jitter

Un pacchetto parte all'istante  $t_1$  e parte con la risposta. Il jitter è:

$$\begin{aligned} \text{jitter source-destination} &= (t_6 - t_5) - (t_2 - t_1) \\ \text{jitter destination-source} &= (t_8 - t_7) - (t_4 - t_3) \end{aligned}$$

Il calcolo delle formule qui sopra è affetto dagli errori di sincronizzazione tra il querier ( $t_1, t_4, t_5, t_8$ ) e il responder ( $t_2, t_3, t_6, t_7$ ).

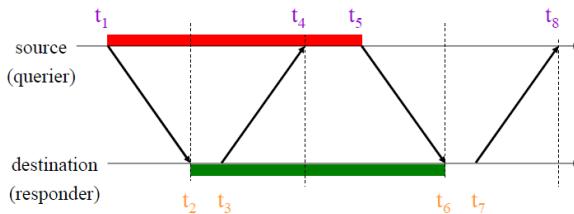


Figura 12 - Misura del jitter di Cisco IP-SLA

Per calcolare il jitter:

$$\begin{aligned} \text{jitter sd} &= (t_6 - t_5) - (t_2 - t_1) \\ &= t_6 - t_5 - t_2 + t_1 \\ &= (t_6 - t_2) - (t_5 - t_1) \\ &= (t_6 - t_2) - \Delta \end{aligned}$$

dove  $\Delta$  è ora un parametro di configurazione

Si calcolano i tempi in cui arrivano i pacchetti al ricevitore ( $t_6 - t_2$ ). ( $t_5 - t_1$ ) che si dovrebbe misurare dall'altra parte viene assunta come costante perché solitamente l'invio è equispaziale e quindi si considera come un delta.

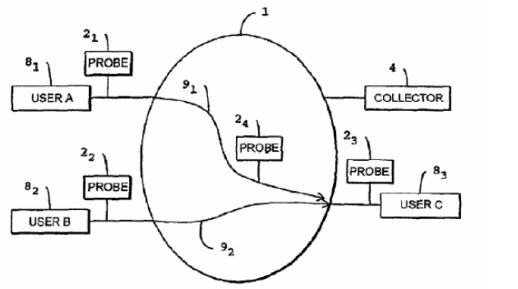
#### Sistemi passivi: ipanema technologies

Degli apparati di misura, chiamati *observing probes* vengono distribuiti nella rete. Di pacchetto  $p_i$  viene calcolata una signature  $h(p_i)$  -funzione hash-. Un sottoinsieme di pacchetti viene selezionato (per esempio: tutti i pacchetti  $p_i$  tali che  $h(p_i) \bmod 16 = 0$ ). Per ogni pacchetto selezionato, ogni observing probe realizza un ticket, contenente:

- identificatore dell'*observing probe*;
- il tempo di passaggio del pacchetto  $p_i$ ;

- il numero di pacchetti che separano  $p_i$  da  $p_{i-1}$ .

I ticket vengono inviati periodicamente ad un *collector* che calcola i tempi di invio dei singoli pacchetti prendendo le informazioni dei vari observer probes sparsi nella rete.



1 = Internet; 2<sub>i</sub> = probes; 4 = collector; 8<sub>i</sub> = clients; 9<sub>i</sub> = routes

Figura 13 - Rappresentazione dell'ipanema technologies

## Soglie e penali per la QoS

Specifiche della QoS:

- come specificare un livello di qualità:
  - scegliere un servizio;
  - scegliere una metrica (tempo di ritardo round-trip);
  - scegliere una soglia X;

si parla di:

- specifica poco stringente:
  - la media dei valori osservati deve essere inferiore a X (non penalizza grandi scostamenti dei valori da X);
- specifiche più stringenti:
  - tutti i valori osservati devono essere inferiori a X;
  - il 90% dei valori osservati deve essere inferiore a X.

Per esempio alcune soglie possono essere:

- Soglie tipiche per la disponibilità: il sistema funziona per x% del tempo ( $h24 * 365$ ):
  - x=99% non funziona per 7,2 ore al mese;
  - x=99,9% non funziona per 43 minuti al mese;
  - x=99,999% non funziona per 312 secondi al mese.
- Esempio di penale: per ogni punto percentuale in più il fornitore restituisce lo 0,1% del costo mensile del servizio.
- Soglie tipiche per il servizio always-on:
  - one-way delay: < 40 ms per il 99% dei pacchetti;
  - packet loss: < 0.15% dei pacchetti;
  - jitter (inter-packet delay variation): valore medio < 15 ms.
- Esempio di penale: per ogni punto percentuale in più il fornitore restituisce lo Z% del costo mensile del servizio.

## Progettazione di servizi basati sul web con esigenze stringenti in termini di prestazioni: architetture, modelli e algoritmi

### Obiettivi

Gli obiettivi che si propone questa parte sono l'identificazione dei problemi a cui si va incontro nell'ambito dell'analisi delle prestazioni di un web server, la classificazione dei Web-Server scalabili, la distinzione tra architetture localmente distribuite e architetture distribuite, la classificazione e comparazione degli algoritmi usati e l'identificazione delle alternative di progettazione.

### Definizioni preliminari

Diamo alcune definizioni preliminari:

- *Hit*: richiesta di un singolo oggetto al server da parte del client (ad esempio un'immagine o un testo);
- *Page Request*: richiesta di una “pagina” composta di solito da vari hit; Session: sequenza di page request consecutive da parte dello stesso client.
- *Tipi di oggetto*: ciascun oggetto può essere statico, dinamico, o sicuro.

### Motivazioni

Prima generazione	Seconda generazione
<ul style="list-style-type: none"><li>• canale economico per info non critiche;</li><li>• 90% testo e immagini.</li><li>• prestazioni fortemente variabili.</li><li>• nessuna garanzia sulla disponibilità del servizio.</li><li>• Scarsa importanza della sicurezza.</li></ul>	<ul style="list-style-type: none"><li>• canale per informazioni critiche;</li><li>• crescente percentuale di contenuto dinamico;</li><li>• valutazione di una organizzazione in base al suo sito</li><li>• necessità di Quality of Service (QoS)</li></ul>

Perché ci concentriamo sulle prestazioni dei web server? Si calcoli che siti come Google o Lycos hanno più di 50 milioni di hit al giorno, e solo nell'edizione 1999 di Wimbledon sono stati fatti quasi un miliardo di hit in soli 14 giorni sul sito apposito, con una giornata di picco con 125 milioni di hit. Inoltre il web della nuova generazione è ben diverso dalla prima, dove non giravano tante informazioni “critiche” e la maggior parte delle pagine era formata da testo. Al giorno d'oggi un'organizzazione si valuta in base al proprio sito, le informazioni sul web sono spesso di livello critico e crescono gli elementi dinamici. Per esempio il sistema funziona per X% del tempo o X% delle richieste ha un tempo di risposta di Y secondi o meno.

Chi offre SLA sono più i Network Provider che le organizzazioni, perché i primi hanno pieno controllo della propria dorsale, mentre gli ultimi controllano le componenti dell'architettura solo in parte.

## Semplice richiesta al server

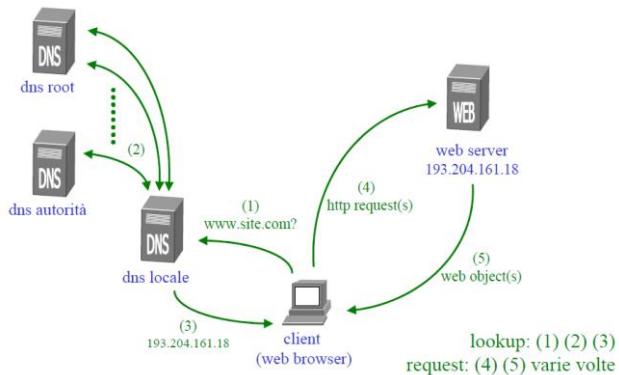


Figura 14 - Semplice richiesta al server

Chi vuole offrire un servizio sul web, controlla il *web server* e in alcuni casi anche il DNS autorità.

Vediamo cosa succede in una semplice richiesta al server.

1. La macchina client chiede l'accesso a un sito www.site.com e quindi dialoga con il DNS locale per farsi dire l'IP corrispondente al nome specificato.
2. Il DNS locale dialoga (eventualmente) con il DNS autorità per farsi dire tale indirizzo, e poi lo riporta alla macchina client.
3. Quest'ultima, finalmente in possesso dell'indirizzo IP, instaura la connessione con il server ed effettua vari HTTP request seguiti ciascuno da un oggetto restituito.

In pratica nella prima fase c'è un lookup dell'IP, nella seconda ci sono i request.

I principali componenti dell'architettura sono:

- Rete;
- Sistema (o sistemi) che erogano il servizio;
- Altre componenti dell'infrastruttura come DNS o cache.

## Problemi e soluzioni

I possibili problemi che possono verificarsi sono a livello di:

- *DNS*: potrebbero esserci indirizzi IP non più validi in cache e il DNS potrebbe andare in timeout;
- *web server*: potrebbero esserci problemi di sovraccarico o non raggiungibilità;
- *internet*: sovraccarico dei link o router;
- *proxy*: problemi di oggetti non più validi.

Le soluzioni possono essere:

- a livello di rete;
- a livello di sistema che eroga il servizio (overprovisioning di CPU, memoria o numero di server con distribuzione locale o geografica); in poche parole, sistema con più memoria e con CPU veloce

(le scelte di overprovisioning sono spesso preferite alle scelte progettuali articolate) e sistema con molti server (distribuzione locale e geografica).

- a livello di infrastruttura (DNS e cache).

## Sistemi basati su più server

I requisiti dei sistemi basati su più server comprendono accesso rapido, trasparenza dell'architettura, scalabilità (sono previsti meccanismi e algoritmi di scheduling per stabilire il miglior server), robustezza, disponibilità e affidabilità.

### HTTP request: caso statico

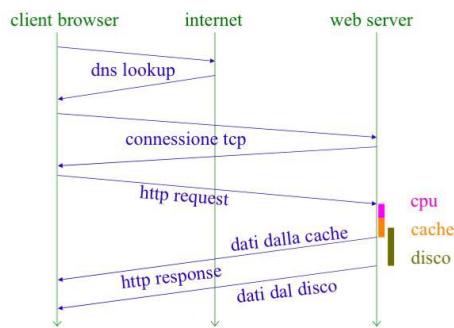


Figura 15 - HTTP request: caso statico

1. Fase di lookup dal client verso internet.
2. Avviene prima la connessione TCP con il web server trovato.
3. Ci sono le richieste HTTP, a cui il web server risponde prima con i dati della cache e poi quelli del disco.

### HTTP request: caso dinamico

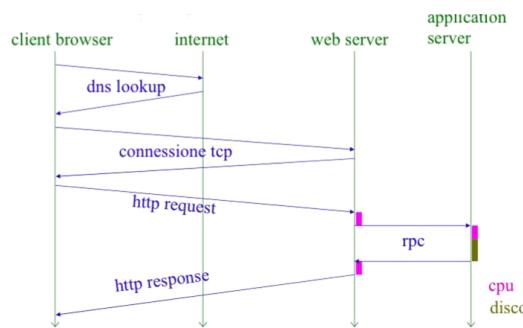


Figura 16 - HTTP request: caso dinamico

1. Fase di lookup dal client verso internet.
2. Avviene prima la connessione TCP col server trovato.
3. Nel caso dinamico all'HTTP request segue una comunicazione con l'application server tramite *rpc*.

Nella progettazione di un servizio bisogna analizzare in dettaglio le caratteristiche del carico sul servizio, secondo modelli qualitativi (solo la media) e quantitativi (termini di probabilità). Gli aspetti da considerare sono:

- i tempi delle richieste;
- dimensione, interesse e tipo degli oggetti richiesti;
- tipo di servizio (statico, dinamico o sicuro).

In genere i tempi di interarrivo hanno distribuzione poissoniana, cioè è più probabile che in un certo intervallo di tempo  $T$  ci siano un certo numero di arrivi piuttosto che ce ne siano di meno o di più (ad esempio è difficile che nell'arco di un minuto, ci siano meno di una decina di arrivi ed è difficile che ce ne siano mille! Il numero di arrivi più probabile è la cima della “campana” di poisson).

Invece la distribuzione della durata delle sessioni e anche del think-time dell’utente tra una richiesta e l’altra è cosiddetta *heavy tailed*, ovvero valori alti sono presenti con probabilità non trascurabili (difatti è possibile che un utente resti sul sito durante una stessa sessione anche per molto tempo come un’ora, o due ore così come cinque minuti, e lo stesso vale per il tempo che ci mette a fare nuove richieste).

Per quanto riguarda la dimensione degli oggetti richiesti, la distribuzione è ancora *heavy tailed*, ma molti oggetti sono piccoli. Per l’interesse, gli oggetti seguono la legge di Zipf: frequenza  $x$  rango = costante, ovvero più è importante un oggetto (rango basso), più sono frequenti gli accessi a quell’oggetto (in realtà in questo caso sembra molto banale questa legge). Per quanto riguarda il tipo, spesso gli oggetti più acceduti sono html e immagini.

Per quanto riguarda le caratteristiche del tipo di servizio, basta fare considerazioni logiche: oggetti statici prevedono meno lavoro di quelli dinamici in termini di CPU e disco. Oggetti sicuri prevedono ancora più lavoro in termini di CPU e di rete, ma non necessariamente di disco.

### Classificazione di riferimento

Un web server “scalabile” è basato su vari server distribuiti. Ogni volta che arriva una richiesta, viene fatto uno scheduling e scelto il server migliore. Lo scheduling prevede tre cose: un meccanismo (per assegnare una richiesta al server “migliore”), un algoritmo (per stabilire quale sia il server “migliore”) e una entità (che esegua l’algoritmo di scheduling e il relativo meccanismo).

meccanismo	entità	livello
name resolution locale/globale	dns	sessione
http redirection locale/globale	web server	page request
packet redirection locale	web switch	sessione / page request / hit

Figura 17 - Overview della tassonomia dei meccanismi di scheduling

- Name resolution: metodo per permettere l'utente di collegarsi ad un certo indirizzo IP; l'entità che se ne occupa è il DNS a livello di sessione.
- HTTP redirecton: posso reindirizzare il servizio; l'entità è il web server con un livello di page request.
- Packet redirection: posso indirizzare ogni singolo pacchetto; l'entità che se ne occupa è il web switch (scatola che ha come obiettivo di scegliere un server e reindirizzare i pacchetti delle richieste che riceve) a livello di sessione / page / request / hit.

Gli algoritmi di scheduling si dividono in:

- statici: si dicono così perché lavorano senza informazioni (potrebbero scegliere un server a caso, oppure ragionare in maniera round-robin, oppure far scegliere all'utente, tipo mirror).
- dinamici: hanno informazioni sul client e/o sullo stato dei server.

In particolare si distinguono meccanismi applicabili a distribuzioni di server locali e globali (geografiche):

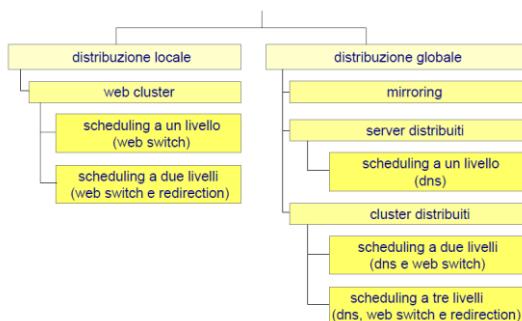


Figura 18 - Tassonomia dei web server scalabili

Se sono distribuiti geograficamente, il web server viene scelto e acceduto tramite il DNS, altrimenti attraverso il web switch.

### Distribuzione Locale – Web Cluster

Web cluster

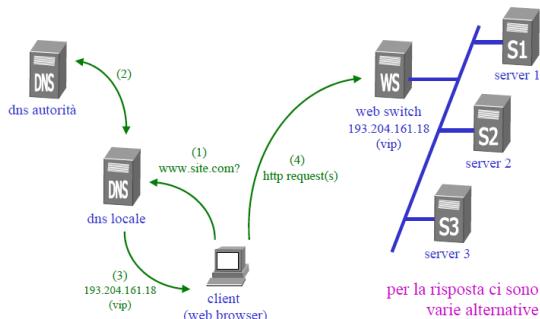


Figura 19 - Esempi di web cluster

Un'architettura web cluster è caratterizzata dal fatto che tutti i server si trovano nella stessa locazione (ad esempio un Internet Data Center – IDC), e sono raggiungibili tramite URL, in un unico indirizzo IP (detto VIP: Virtual IP) mostrato all'esterno, mentre i singoli server hanno indirizzi IP privati, che possono essere assegnati a diversi livelli della pila ISO-OSI.

Quindi il DNS locale riceve dall'autorità un unico indirizzo IP (il VIP) da riferire al client, con cui poi instaura la connessione TCP. Tale connessione è instaurata con un web switch che si occupa dello scheduling (algoritmo e meccanismo) e comprende richieste e risposte. La richiesta dunque è fatta al web switch, per la risposta invece ci sono varie alternative divise in "a-due-vie" (la risposta la ridà lo stesso switch) e "a-una-via" (la risposta la dà direttamente poi il server scelto).

### Web switch

Cos'è di preciso il web switch? E' il componente che smista le richieste, effettuando il mapping tra VIP e indirizzo dei server. Può essere un hardware "special purpose" (fatto apposta) oppure un modulo software di un SO. Il web switch effettua un controllo fine sulle richieste e può farlo a livello di hit, page request o sessione.

Alcune alternative del web switcher sono:

- I web switch di livello 4 non conoscono il contenuto delle pagine (content-blind) e si basano su indirizzo IP di sorgente e destinazione, numero di porta TCP, e i bit syn e fin dei pacchetti TCP che indicano l'apertura e la chiusura di connessioni (ovvero lavorano a livello di trasporto/rete).
- I web switch di livello 7 sono content-aware e sono basati su contenuto dell'URL, cookies e SSL ID (valore usato per connessioni sicure).

Quindi, l'attuale tassonomia riscontrata è composta da quattro tipologie, che sono tutte le combinazioni possibili tra web switch di livello 4 o 7 con architettura ad una o due vie. All'interno di queste quattro tipologie, sono possibili vari algoritmi di scheduling, statici o dinamici, e vari meccanismi.

### Switch di livello 4

Gli switch di livello 4 operano dunque a livello di TCP/IP. I pacchetti relativi ad una stessa connessione (identificati tramite il syn) sono fatti corrispondere ad uno stesso server tramite una tabella connessione-server mantenuta dallo switch. Per identificare nuove sessioni, si usa il bit syn (appena arriva il syn dell'inizio della connessione lo switch parte direttamente con l'assegnazione del server, non serve aspettare il completamento del tree-way-and-shake perché a questo livello non occorre conoscere il contenuto del pacchetto).

## DEI WEB CLUSTERS CON LB L4

La classificazione avviene in base a:

- uso di diversi meccanismi per ridirigere i pacchetti in arrivo;
- flusso dei pacchetti: **USCITA**
  - due-vie: sia i pacchetti in arrivo che quelli in partenza ~~attraversano~~ dal web switch;
  - una-via: solo i pacchetti in arrivo ~~attraversano~~ dal web switch.

PASANO

PASANO

### Web switch di livello 4 – Architetture a due vie

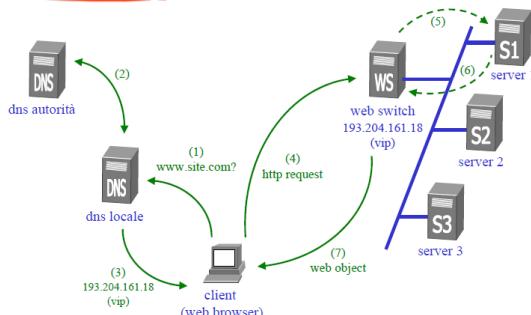


Figura 20 - Web switch di livello 4 – Architetture a due vie

Nell'architettura a due vie lo switch “riscrive” anche i pacchetti in partenza. Tale riscrittura segue l'approccio NAT (*Network Address Translating*). Significa che il VIP comune gestito dallo switch viene tradotto nell'IP della rete interna dei server e viceversa per i pacchetti in uscita. Questo significa però dover ricalcolare le checksum TCP. Nello switch di livello 7 in questo caso avviene tutto in modo più lento e la macchina è molto sollecitata.

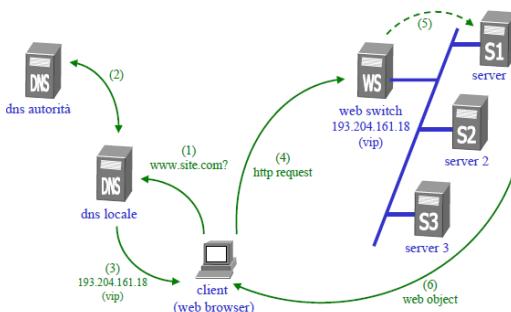


Figura 21 - Web switch di livello 4 – Architetture ad una via

Nell'architettura ad una via i pacchetti in uscita non passano per lo web switch (lo switch modifica solo i pacchetti in ingresso -al vip viene sostituito l'IP del server prescelto-), e i server modificano i pacchetti in uscita ponendo come mittente l'indirizzo VIP. Un'alternativa dell'architettura ad una via è evitare la riscrittura a livello IP e usare gli indirizzi MAC per il reindirizzamento ai server:

- VIP è definito sull'interfaccia di loopback di ogni server (IP alias);
- il forwarding dallo switch è effettuato a livello MAC (uso del mac del server prescelto);
- vincolo: web switch e server devono far parte della stessa sottorete fisica.

Dal punto di vista degli algoritmi:

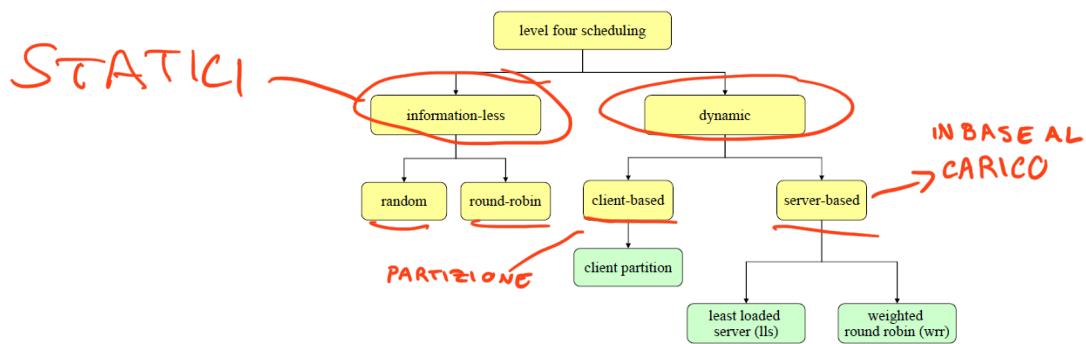


Figura 22 - Schema degli algoritmi usati nel web switch

Gli algoritmi possibili sono statici (informaton-less) e dinamici. Tra gli statici ci sono random (assegnazione casuale: nessuna informazione sullo stato dei cluster né sulla storia delle assegnazioni) e round-robin (assegnazione "a turno", l'unica informazione è l'assegnazione precedente). Tra gli algoritmi dinamici annoveriamo la partizione dei client (assegnazione effettuata in base all'indirizzo dei client, in questo modo è possibile controllare la Quality of Web Service per gruppi di utenti), assegnazione in base al carico dei server (osservato dallo switch in termini di connessioni attive, appreso dal server stesso tramite informazioni su uso di CPU e disco o ricavato tramite emulazioni di richieste dallo switch ai server). Esempi di architetture web switch di livello 4 a due vie sono Cisco Local Director, Cabletron lsnat, Alteon Web Systems, Linux Virtual Server. A una via sono IBM TCP router & network dispatcher e lsmac.

### Switch di livello 7

Gli switch di livello 7 operano invece a livello di applicazione. Client e switch in questo caso hanno bisogno di instaurare una connessione TCP, altrimenti lo switch non può gestire la GET, e deve quindi gestire l'intera instaurazione per accedere al contenuto della HTTP. Ispezionano il contenuto dei pacchetti HTTP per decidere sull'assegnamento, quindi lo switch abbia un parser delle richieste HTTP, e questo ci consente anche di lavorare per connessione, cioè gestire anche i *keep alive*. Possono dunque indirizzare le richieste su server specializzati in funzione del contenuto della richiesta HTTP. Dato che possono arrivare al pacchetto HTTP solo dopo averlo extrapolato da quello TCP, devono gestire per intero la connessione TCP e occuparsi degli ack. Anche gli switch di livello 7 si classificano in base ai meccanismi usati e all'architettura a due o una via. In particolare stavolta ci sono quattro particolari architetture possibili, due a due vie (*TCP gateway* e *TCP splicing*) e due a una via (*TCP handoff* e *TCP connection hop*).

### TCP gateway

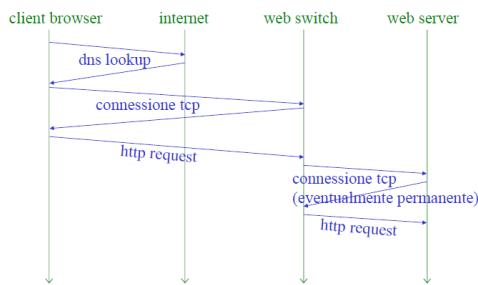


Figura 23 - Architetture a due-vie TCP gateway

Nel due-vie TCP gateway, lo switch si comporta come un proxy (rappresentante) per entrambe le parti e fa da intermediario alla connessione. In questo modo l'overhead è altissimo: lo switch deve mantenere due connessioni TCP e tutta la pila ISO-OSI è concentrata su di esso.

### TCP splicing

Nel TCP splicing, che ne è una ottimizzazione, dopo aver instaurato le due connessioni TCP, lo switch effettua una traduzione di indirizzi e numeri di sequenza operando esclusivamente a livello 3 (di rete) in modo tale da “incollare” le due connessioni TCP in un'unica connessione a livello 4.

### Architettura a due vie: TCP splicing

- *Fase uno:* le due connessioni client-switch e switch-server sono state instaurate.
- *Fase due:* si effettua una traduzione di indirizzi e numeri di sequenza in modo tale che le due connessioni siano “saldate” in una connessione unica .

I campi *ACK number* e *sequence number* si riferiscono ai singoli byte trasmessi nella connessione. Il valore del campo *ACK number* è il numero del byte che si sta attendendo.

Il valore del campo *sequence number* è il numero del primo byte dei dati contenuti nel pacchetto. Gli algoritmi usati sono molteplici. Per abbassare i tempi di risposta il content switch può essere configurato per anticipare i client e preparare delle connessioni pronte con i server: pool di connessioni attive con i server pronte all'uso.

I tre attori sono: client, (content) switch e server, e questo è il diagramma temporale di ciò che succede:

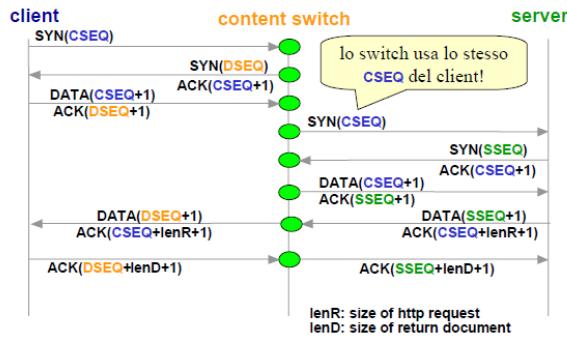


Figura 24 - Architetture a due-vie TCP splicing

Finora ad  $ACK(DSEQ+1)$  abbiamo semplicemente il 3-way-handshake. Per faticare un po' meno sui pacchetti in transito lo switch può specificare come numero di sincronizzazione col server il *CSEQ* specificato dal client. Questo consente di usare gli stessi numeri di sequenza. Se non abbiamo fatto tutto il 3-way-handshake non possiamo scegliere il server (non abbiamo visto il contenuto della richiesta). Il primo SYN non può essere immediatamente forwardato al server. Il server specifica poi un suo numero iniziale di sequenza; questo vuol dire che quando gli invio i dati la numerazione dei dati va benissimo in questa direzione (da sinistra verso destra) ma nell'altro verso devo specificare i numeri di sequenza del server (*SSEQ*). In pratica si risparmia solo la metà del lavoro di rinumerazione, i pacchetti inviati verso il client vanno rinumerati.

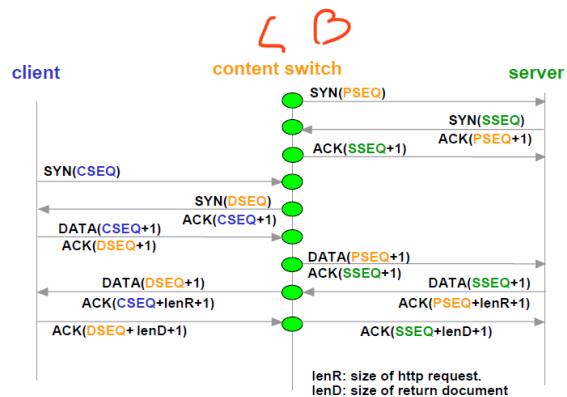


Figura 25 - TCP splicing - connessioni pronte

Questa è una variante di TCP splicing, con connessioni pronte; in questo caso lo switch ha una connessione pronta verso i server. Quando arriva la richiesta da parte dei client si sceglie il server e gli si spedisce la

richiesta senza aspettare la fine del 3 way handshake. Un'altra variante è quella con server preallocato; il prossimo e il disegno “sbagliato”, in cui si pensa che questo sia uno switch livello 4 invece che livello 7.

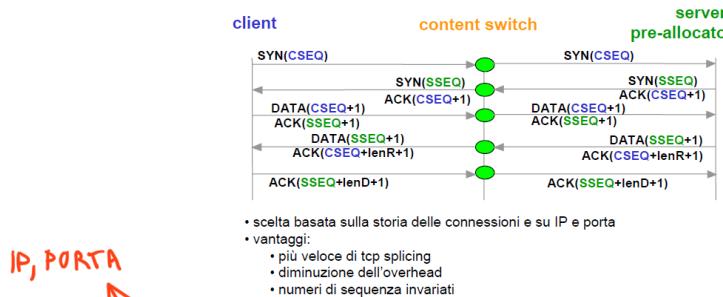


Figura 26 - TCP splicing - server pre-allocato

Lo switch, in funzione delle scelte del passato, non aspetta di guardare il contenuto, ma lo spedisce ad un certo server; potrebbe aver fatto la scelta sbagliata, quindi va gestito il fallimento: si manda un FIN sul server preallocato, e si fa partire la connessione con un altro server (quello giusto).

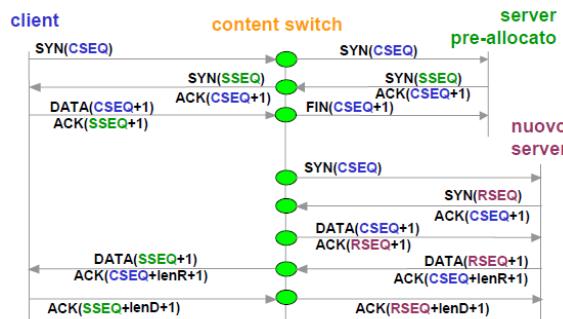
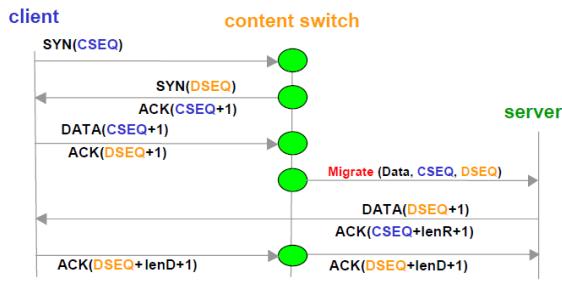


Figura 27 - TCP splicing - server pre-allocato (riallocazione)

Quello che succede e che non si trovano sempre tutte queste tecniche su tutti gli switch possibili.

#### TCP handoff

Nell'una-via TCP handoff, la connessione instaurata tra client e switch viene trasformata in una connessione tra client e server, mentre nella variante connection hop la connessione viene trasferita al server con un protocollo apposito. Quindi c'è un protocollo apposito che migra la connessione dal client al server. Questo protocollo deve fare un trasferimento di stato da una macchina all'altra, dopodiché il server gestisce il nuovo stato come se lui stesso fosse con quella connessione attiva. Questo implica che si entri nel kernel di questi oggetti.



- migrazione della connessione dallo switch al server
  - con il server non si fa il three-way handshake
- quando la connessione è passata al server lo switch deve inviargli i pacchetti

Figura 28 - Una-via TCP handoff

Trasformazione della connessione instaurata tra client e switch in una connessione tra client e server.

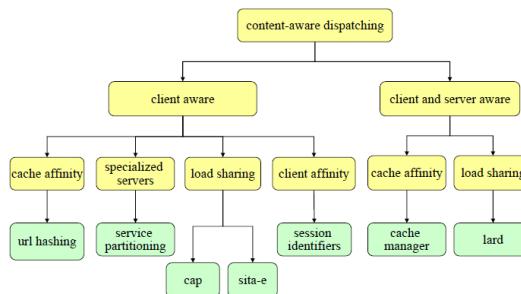


Figura 29 - Web Switch di livello 7 - tassonomia degli algoritmi di scheduling

### Algoritmi

- Algoritmi in funzione della sessione: richieste HTTP con lo stesso ssl id o gli stessi cookie sono assegnate allo stesso server, così si evita di identificare più volte lo stesso client;
- Algoritmi in base al contenuto stesso: Siccome a questo punto conosciamo il tipo del contenuto richiesto, possiamo usare server specializzati per ciascun tipo di risorse; possiamo altrimenti assegnare i server in funzione della dimensione del file richiesto (site-e, site interval task assignment with equal load), con l'obiettivo di privilegiare i task più leggeri. Un'altra scelta è partizionare il file space tra i server utilizzando funzioni di tipo hash: si prende il contenuto della GET e ci si applica una funzione hash. Questo fa sì che quando si accede alla stessa URL si è sempre direzionati verso lo stesso server, e si massimizza l'hit rate sulle cache dei server;
- CAP – Client Aware Policy: classificare le risorse in funzione dell'impatto sulle singole componenti dei server (cpu, dischi, rete), e quindi in base al tipo di richiesta (una query sql avrà impatto sui dischi, un download di un grosso file sulla rete, richieste con alti requisiti di sicurezza sulla cpu). Questa politica è accompagnata da un'assegnazione ciclica delle classi ai server, per diversificarne l'impegno:
  - basso impatto: piccoli file statici;
  - impatto sulla rete: download di grandi file;
  - impatto sui dischi: query su database;

- impatto sulla CPU: richieste con requisiti di sicurezza;
- *Identificazione di sessione*: richieste http con lo stesso SSL ID o con lo stesso cookie assegnate allo stesso server. L'obiettivo: evitare di identificare più volte lo stesso client. Le richieste HTTP con lo stesso SSL ID o con lo stesso cookie vengono assegnate allo stesso server. A livello 7 possiamo direzionare tutte le richieste che hanno lo stesso identificatore dentro un cookie verso lo stesso server. Stiamo utilizzando HTTP dentro SSL; quando si usa SSL viene fornito un ID di sessione a chi la mette in piedi, e molto vantaggioso (a volte indispensabile) che si continui a lavorare dentro la stessa connessione sicura.
- *Locality-Aware Request Distribution (LARD)*: la prima richiesta di una certa risorsa è assegnata al server meno carico (con criterio: numero di connessioni). Successive richieste della stessa risorsa sono assegnate allo stesso server, per aumentare la località e quindi lo hit rate delle cache. L'obiettivo è aumentare la località e quindi lo hit rate delle cache dei server.
- *Cache Affinity*: tecnica basata su un cache manager a cui è nota la situazione delle cache di tutti i server;

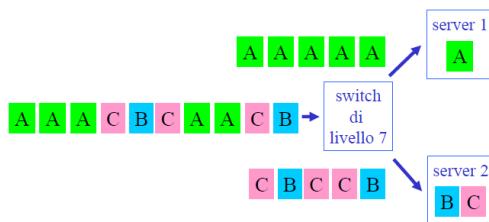


Figura 30 - Web Switch di livello 7 - algoritmi di scheduling con conoscenza di informazioni sul client e sul server

Esempi disponibili sul mercato sono: per TCP gateway *ibm network dispatcher*, per TCP splicing *alteon*, *arrowpoint*, e *foundry nets*, per TCP handoff *lard*, per TCP connection hop *resinate central dispatcher*.

In conclusione confrontiamo le architetture dei web cluster di livello 4 e 7:

Livello 4	Livello 7
<ul style="list-style-type: none"> <li>• Operazioni di switch veloci;</li> <li>• Controllo a livello di hit per HTTP 1.0;</li> <li>• Controllo a livello di page request per http 1.1 (se i vari hit corrispondenti appartengono alla stessa connessione TCP);</li> <li>• informazioni sul client solo a livello TCP/IP.</li> </ul>	<ul style="list-style-type: none"> <li>• operazioni di switching più lente;</li> <li>• controllo a livello di hit per HTTP 1.0;</li> <li>• Controllo a livello di hit o page request per HTTP 1.1;</li> <li>• Informazioni sul client a livello HTTP.</li> </ul>

## Distribuzione globale: sistemi Web Distribuiti

Qui si parla di distribuzione geografica (server di uno stesso sistema anche molto distanti fra loro). Le architetture possibili sono: siti mirror, server distribuiti, cluster distribuiti. Gli algoritmi di scheduling possono essere a uno, due o tre livelli. Sono possibili vari modelli e metriche.

### Siti mirror

I siti mirror prevedono una stessa informazione replicata su più web distribuiti geograficamente. L'indirizzamento prevede nomi multipli, uno per ciascun sito replicato, e lo scheduling è lasciato all'utente che sceglie il sito da una pagina di partenza. Nonostante sia molto semplice come architettura, la replicazione è visibile, bisogna mantenere la consistenza fra i vari siti e non c'è alcun controllo sulla distribuzione del carico. Questo significa che non si può fare bilanciamento di carico: se tutti fanno la stessa scelta non possiamo farci molto. Il vantaggio di questa architettura è che è molto semplice.

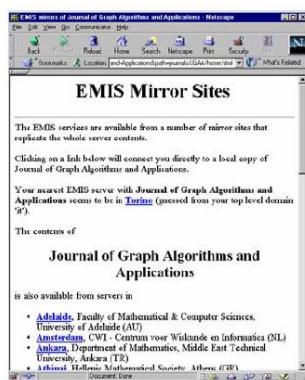


Figura 31 - Esempio di sito mirror

### Server distribuiti

Poi ci sono i server distribuiti, che sono più server distribuiti geograficamente e replicati, con lo stesso nome ma con indirizzi IP diversi.

### Scheduling

Lo scheduling è effettuato dal name server autorità per quel sito. Quindi è il DNS autorità per il sito (ma non necessariamente) che sceglie il server al primo livello di scheduling. Il name server autorità per la zona può essere un po' più furbo di un semplice name server. Questo name server può avere informazioni sullo stato dei serve ovunque distribuiti, e può darmi un indirizzo IP in funzione di questo stato. Questo può darmi algoritmi statici (*information-less*) o dinamici. Quelli dinamici sono basati su:

- conoscenza di informazioni sul client;
- conoscenza dello stato dei server;
- conoscenza di entrambi.

Cosa vuol dire "informazioni sul client"? Al name server arriva la richiesta del client, e il name server autorità della zona, in funzione ad esempio della posizione geografica per esempio dei diversi server ne

assegna uno. Ma che ne sa il name server autorità della zona dove sto, come fa a capirlo? Quando sono sul mio browser, e scrivo [www.site.com](http://www.site.com), il name server autorità della zona, in quel momento, quale informazione può sfruttare? Chi parla con chi in questa storia? Il browser contatta il local DNS, a questo punto il name server è impostato per andare a chiedere ad un dato name server; chi e che fa la richiesta al name server autorità per [site.com](http://site.com)? È il mio local DNS, e quello che vede il name server autorità non è il mio IP, ma l'IP del local DNS, e allora pensa: se quella richiesta mi arriva da x o da y, posso pensare che quell'utente sia più o meno nei paraggi, e allora gli assegno un server ragionando in quel modo. Nel caso di richiesta ricorsiva, cosa succede? Un local DNS fa una richiesta ricorsiva in un solo caso: quando si vuol far prendere a male parole dai name server intermedi, perché non potrebbero mai sobbarcarsi quel carico. Questo è un secondo livello di scheduling molto più difficile, perché il carico, quando si lavora su scala geografica, magari e funzione di fusi orari, giorno e ora, dopodiché possiamo avere problemi di caching dell'indirizzo IP nei DNS intermedi per tutto il TTL (in caso di uso del DNS). Questo scheduling viene implementato tramite HTTP redirect; il client quindi ad ogni richiesta riceve dal DNS autorità la coppia indirizzo IP (scelto) + TTL (time to live, ovvero quanto tempo tale indirizzo deve rimanere in cache nei DNS intermedi). Gli algoritmi previsti possono essere al solito statici o dinamici.

Possiamo anche avere problemi a fare delle buone stime di prossimità. Tutte cose che rendono difficile questo scheduling, ma questo è lo scheduling che si usa per i servizi internet importanti. Come si stima la prossimità in internet? Una possibilità è stimare il tempo di latenza roundtrip, e questo è largamente dipendente dalla prossimità geografica. Rispetto a che cosa si fa la stima? Non possiamo fare ping sui singoli utenti (non scalerebbe), ma possiamo farlo sui name server locali. Quindi, chi fa ping su chi? La cosa è abbastanza articolata: abbiamo i local DNS come rappresentativi di una classe di utenti vicini, poi abbiamo i server distribuiti geograficamente, che però non vedono i local DNS. Dobbiamo riuscire a dire ai server locali qual'è l'elenco dei local DNS che periodicamente lo contattano, e ogni server fa i ping su quei local DNS. Queste informazioni sui ping si possono poi inviare ai local DNS della zona. L'alternativa è che, invece di fare ping, i server facciano le richieste ai name server usando non il proprio name server, ma questi name server della zona. Questa cosa prevede di avere una infrastruttura piuttosto solida.

Lo scheduling dinamico può essere complesso perché funzione del carico e quindi di giorno, ora, fusi orari, distribuzione degli utenti in internet ma anche prossimità tra client e server, e caching nei DNS intermedi. A ciò si aggiunge che i tempi di latenza spesso sono indipendenti dalla vicinanza geografica e i link non hanno tutti la stessa capacità.

Posso poi far uso di informazioni statiche. Posso guardare l'IP del client, e in funzione di questo IP posso cercare di capire più o meno dove sta quel client. Le informazioni statiche che si possono ottenere staticamente sono indirizzo IP del client, dal quale si può ricavare la zona di internet (*l'Autonomous System*) in cui esso risiede, numero di network hop (router tra i due nodi) e numero di AS hop ("zone

internet” tra i due nodi). Una cosa che si fa tantissimo è cercare di capire quale sia l'autonomous system in cui è allocato il client. Normalmente un autonomous system corrisponde ad un provider; lo scheduler furbo ha una mappa di internet in cui sa dove sono gli autonomous system, e se è proprio furbissimo sa anche come sono collegati fra loro. Anche se non sappiamo quest'ultima cosa, conosciamo l'indirizzo del name server che mi sta facendo la richiesta, e sappiamo che il prefisso di questo name server appartiene ai prefissi di un certo provider, che sta in un certo paese. Se incappiamo però su link con una banda troppo limitata comunque non offriamo un servizio così buono.

Nel caso in cui si faccia valutazione dinamica della prossimità abbiamo necessità di tempo addizionale e traffico per la valutazione. Le informazioni dinamiche invece sono roundtrip di latenza (tempo che intercorre tra l'invio di un segmento e la ricezione del relativo ack, ottenibile tramite *ping*), disponibilità di banda sui link e tempo di latenza di una richiesta HTTP (ottenibile tramite un'emulazione). Si noti che per effettuare tali valutazioni serve tempo e traffico addizionale sulla rete. Pare che ci sia un correlazione tra hop (informazione statica) e latenza (informazione dinamica).

Alcune considerazioni sul TTL: si potrebbe porlo uguale a 0 per affidare al DNS il pieno controllo dello scheduling, ma non è sicuro che i DNS collaborino a questo scopo (perché potrebbero ignorare TTL molto piccoli), inoltre si sovraccaricherebbero troppo, e infine la cache dei browser ignorano comunque il TTL. L'altra alternativa è un TTL dinamico che si adatta al carico attuale dei server e alla frequenza di richieste da un certo dominio. La verità è che gli algoritmi attuali usati dai DNS sono molto complessi (a differenza dei web switch). Alternative prevedono di non utilizzare i DNS bensì redirect fin dal primo livello di scheduling. Oppure utilizzare i DNS in primo livello e HTTP redirection (o anche IP tunneling) in secondo livello.

Esperimenti con *nslookup*.

---

```
> server mail.dia.uniroma3.it SERVER
      ALL' UNIVERSITA' DI ROMA3
Default Server: mail.dia.uniroma3.it
Address: 193.204.161.133
> www.google.com
Server: mail.dia.uniroma3.it
Address: 193.204.161.133
Non-authoritative answer:
Name: google.lb.google.com
Address: 64.208.32.100
Aliases: www.google.com
> server neddy.newcastle.edu.au SERVER IN AUSTRALIA
Default Server: neddy.newcastle.edu.au
Address: 134.148.24.1
> www.google.com
```

---

```
Server: neddy.newcastle.edu.au
Address: 134.148.24.1
Non-authoritative answer:
Name: google.lb.google.com
Address: 216.239.35.100
Aliases: www.google.com
```

---

Il comando *nslookup* permette di trovare quali indirizzi IP restituisce il DNS per un certo nome ad un certo client specificato. Per specificare il server a cui chiedere la risoluzione si digita “*server xxx.yyy.zzz*”, dopodiché si digita il nome del sito (ad esempio [www.google.com](http://www.google.com)) e il programma restituisce il nome del sito e i vari alias (come ad esempio [www.l.google.com](http://www.l.google.com)) e gli indirizzi IP restituiti dal server specificato. Quando dice *non-authoritative answer* significa che gli indirizzi IP che sta dando sono cachati in qualche DNS intermedio.

Se facciamo due query a distanza di un minuto l'una dall'altra:

---

```
> www.yahoo.com
Server: mail.dia.uniroma3.it
ANSWERS:
-> www.yahoo.akadns.net
    internet address =
        64.58.76.178
        ttl = 40 (40S)
Non-authoritative answer:
    Name:
        www.yahoo.akadns.net
    Addresses: 64.58.76.178,
        216.32.74.50, 64.58.76.177,
        216.32.74.52
        64.58.76.176,
        216.32.74.51, 216.32.74.55,
        64.58.76.179, 216.32.74.53
    Aliases: www.yahoo.com
```

```
> www.yahoo.com
Server: mail.dia.uniroma3.it
ANSWERS:
-> www.yahoo.akadns.net
    internet address =
        216.32.74.51
        ttl = 300 (5M)
Non-authoritative answer:
    Name:
        www.yahoo.akadns.net
    Addresses: 216.32.74.51,
        64.58.76.176, 216.32.74.52,
```

```

64.58.76.177
216.32.74.53,
216.32.74.55, 216.32.74.50,
64.58.76.179, 64.58.76.178
Aliases: www.yahoo.com

```

In queste risposte non c'è un solo indirizzo, ma un insieme di indirizzi. Se guardo questi insiemi scopro che sono uguali (potrebbero anche non esserlo). Yahoo sta sfruttando un'altra furbata che si può fare in questi casi: l'ideale sarebbe che il local DNS scegliesse quale è il server migliore tra i tanti disponibili. I local DNS fanno una cosa: quando arriva come risposta ad una query un pool di indirizzi se ne sceglie uno a caso. Questo va molto bene a Yahoo, perché usa questa cosa per fare uno scheduling sui propri server. Io sto vedendo nelle risposte due indirizzi diversi soltanto perché ne è stato privilegiato uno. A causa del caching i DNS controllano solo una parte del traffico destinato ad un certo sito, e ridurre di molto il TTL pone diversi problemi:

- il TTL spesso non ha effetto sulla cache dei browser (il browser rimane “inchiodato” ad un name server);
- DNS non cooperativi ignorano i TTL molto piccoli.

Il name server, che non gestisce a questo punto poi così tante richieste, ha tutto il tempo per usare algoritmi più sofisticati.

Le possibili soluzioni per i problemi posti dallo scheduling basato su DNS sono:

- il DNS non si occupa più dello scheduling; ci pensa un'altra entità;
  - HTTP redirection fin dal primo livello di scheduling;
- uso di un secondo livello di scheduling dopo il DNS;
  - HTTP redirection;
  - IP tunneling;
- rimpiazzamento dei server con dei cluster.

#### HTTP redirect

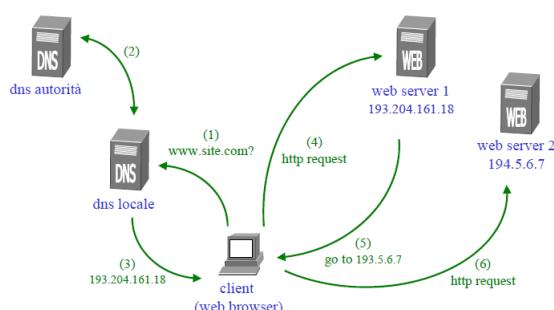


Figura 32 - HTTP redirect

Qualche parola sull'HTTP redirect, che è di fatto il secondo livello di scheduling. Esso prevede uno scheduling distribuito (ciascun server può fare un redirect, assegnando la richiesta a qualcun altro). È completamente trasparente all'utente (ma non al client), e la re-direzione può essere verso un nuovo nome o un nuovo indirizzo IP. Le politiche di redirect possono essere molteplici. Bisogna scegliere chi può effettuare redirect (solo il DNS o tutti i server), quali pagine ridirigere (tutte, solo quelle che superano una certa soglia in dimensione o numero di hit), e a chi redirigere (round robin, server meno carico, funzione hash, prossimità). L'HTTP redirect è ottimo ma si possono redirigere solo le richieste HTTP e aumenta traffico e tempo di risposta, dato che bisogna mantenere due connessioni HTTP per richiesta). Sul mercato scheduling a un livello sono *server ncsa*, *cisco distributed director*, *sun scalar* mentre a due livelli c'è *sweb*.

PRO	CONTRO
<ul style="list-style-type: none"> <li>• Piena compatibilità con i client e con i server.</li> <li>• Non introduce un punto critico la cui caduta rende indisponibile il sistema.</li> </ul>	<ul style="list-style-type: none"> <li>• consente di ridirigere solo le richieste http;</li> <li>• Può aumentare traffico e tempo di risposta: due connessioni HTTP per richiesta.</li> </ul>

#### *Web cluster distribuiti geograficamente*

Per ottenere il terzo livello di scheduling basta introdurre anche i web cluster. In pratica sono web cluster distribuiti geograficamente. Il nome è unico ma c'è un indirizzo IP (ovvero VIP) per ogni cluster, con indirizzi privati dei server all'interno. I tre livelli di scheduling sono:

- DNS o altro durante il lookup;
- web switch;
- redirect dai web server (o dal web switch se è di livello 7).

## Content delivery (distribution) networks

Le CDN sono delle strutture fatte per migliorare le prestazioni del web. Lo strumento che si usa è quello di distribuire server in giro per il mondo, e di attribuire poi a ciascun client il server migliore da qualche punto di vista. L'atteggiamento è lo stesso di quello dei server web propriamente distribuiti, ma su scala immensa. A questo punto, con tecniche analoghe a quelle viste per i servizi web distribuiti si fa l'allocazione delle richieste al server migliore. Chi deve distribuire dei contenuti stipula un contratto con una o più CDN per distribuire i propri contenuti. Si trova tra il livello backbone IP e sotto il web-application e web-server.

### Akamai

È un'enorme infrastruttura della rete a cui accediamo senza saperlo. Senza Akamai, molti servizi importanti (Yahoo, Amazon, BBC) non riuscirebbero a distribuire i GB di dati al secondo che sono richiesti da immagini, video e animazioni flash presenti nelle loro pagine. Passare da una CDN ad un'altra richiede una configurazione quasi marginale. Possiamo pensare che Akamai (e questo vale anche per altre CDN) formi un'enorme rete di overlay sopra il backbone IP<sup>1</sup> e sotto le applicazioni. Hanno dei server in giro per il mondo, tengono costantemente sotto controllo la rete con quei server e stabiliscono i percorsi migliori.



Figura 33 - Esempio di indirizzamento Akamai

Prendiamo due città e vediamo come vanno le cose tra due città A e B. Viene indicata la latenza e il packet loss che loro forniscono. Avendo tanti server in giro per il mondo, quindi si può scoprire che facendo giri più larghi si può guadagnare in latenza e packet loss (*path optimization*).

Quali sono i segreti di Akamai? I meccanismi di misura della rete, di scelta del percorso e distribuzione delle cache sono algoritmi proprietari, e non sono noti, ma i meccanismi che permettono ad Akamai di redirezionare le richieste dei client sono pubblicamente noti. Akamai funziona con redirezione DNS. Il meccanismo è grossomodo sempre quello: redirezione DNS e riscrittura della URL. Qualche tecnica di base: dato che ogni CDN dispone di un server in prossimità dei POP (*Point of Presence*, punti di accesso

<sup>1</sup> è un collegamento ad alta velocità di trasmissione e capacità tra due server o router di smistamento informazioni e appartenente tipicamente alla rete di trasporto di una rete di telecomunicazioni.

alla rete) degli ISP, le richieste possono essere inoltrate dinamicamente al duplicato più vicino. Di solito si usa la redirezione DNS o la riscrittura dell'URL. Molti CDN giacciono su sottosistemi per la misurazione delle reti, così possono utilizzare le informazioni dinamiche conosciute per selezionare i server duplicati e determinare i cammini sui quali trasferire i contenuti.

#### Redirezione DNS

Una richiesta DNS di un client viene redirezionata al name server autorità controllato dalla CDN, che risolve il nome del server della CDN con l'indirizzo IP di uno o più server. La redirezione può essere di contenuti completi (redirezione di tutti i request) o parziali (redirezione soltanto di alcuni URL immersi che vengono modificati dal sito d'origine). Nel primo caso, tutte le richieste DNS per il server originale sono redirette alla CDN; nel secondo caso il sito d'origine modifica alcune URL incorporate nella pagina così che solo le richieste per quelle URL siano redirette alla CDN. Akamai spesso utilizza redirezioni di contenuto parziale.

In particolare Akamai traduce la request di un client che sta richiedendo contenuto nel dominio di un cliente di Akamai, nell'indirizzo IP di un server Akamai vicino, detto *Edge Server* (ESI). ESI è un linguaggio di markup usato per definire pagine web, ed è il linguaggio utilizzabile per descrivere i contenuti facendo sì che Akamai si prenda carico della consegna degli oggetti. Può migliorare le prestazioni mettendo in cache parti di codice HTML generate dinamicamente e permettendo l'assemblaggio del contenuto dinamico sui server di bordo area. ESI è una variante di SSI, fatto per la stessa cosa nel mondo di Apache. La sua caratteristica principale è la possibilità di cercare e includere i file di cui è composta una pagina web. Per ciascun file vengono indicati il TTL nella cache e istruzioni di ri-convalida. I file di inclusione possono contenere markup ESI per ulteriori elaborazioni, fino a 15 livelli di inclusione.

#### Passaggi dell'indirizzamento di contenuto parziale (*partial content delivery*)

- Un client web emette un request per un oggetto immerso che risiede nel dominio richiesto; dopodiché richiede al proprio server DNS locale (LDNS) l'indirizzo IP nel classico nel classico modo:
  - Richiesta al Local DNS Server (LDNS).
  - Richiesta al Root DNS Server e risposta su dove proseguire.
  - Richiesta ai server di livello minore e relative risposte.
  - Richiesta al Web Server.
- Quindi l'LDNS tenta una traduzione di nome per conto del client. Quando il name server del dominio è contattato per una traduzione di nome, esso comincia la redirezione DNS ritornando una entry CNAME per il contenuto richiesto; poiché il contenuto è servito da Akamai, il valore di CNAME in questo caso è modificato ed è quello di Akamai.
- L'LDNS esegue nuovamente una traduzione di nome sul nuovo dominio.

- Quindi, una gerarchia di server DNS di Akamai risponde alla richiesta di traduzione di nome, determinando e restituendo gli IP degli Edge server migliori, in base a IP del DNS locale (o del client, a seconda di dove sia stata originata la richiesta di traduzione), nome del customer Akamai e nome del contenuto richiesto.

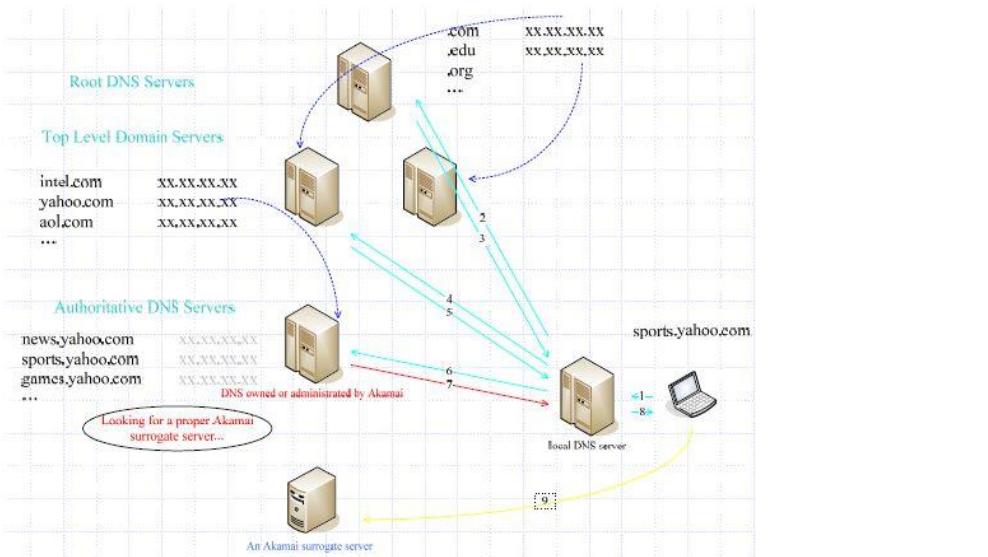


Figura 34 - Esempio redirizione DNS con Akamai

Esempio di redirezione DNS di Akamai.

- Homepage di PCWorld.com.
- Un client web emette un request per un oggetto immerso che risiede nel dominio *images.pcworld.com*; dopodiché richiede al proprio server DNS locale (LDNS) l'indirizzo IP di *images.pcworld.com*.
- Quindi l'LDNS tenta una traduzione di nome per conto del client. Quando il name server di PCWorld è contattato per una traduzione di nome, esso comincia la redirezione DNS ritornando una entry CNAME per *images.pcworld.com*; poiché il contenuto del dominio *images.pcworld.com* è servito da Akamai, il valore di CNAME in questo caso è *images.pcworld.com.Edgesuite.net*; *Edgesuite.net* è un dominio di proprietà di Akamai.
- L'LDNS esegue nuovamente una traduzione di nome, questa volta sul dominio *Edgesuite.net*. Sono in seguito eseguite due nuove redirezioni, prima al dominio *akam.net* (es. *adns1.akam.net*), poi a *a1694.g.akamai.net* (dove 1694 è il numero di customer per PCWorld). In generale, le redirezioni Akamai includono un numero di customer nel nome del dominio.
- Nello stadio finale della traduzione, la rete di Akamai usa una gerarchia di name server Akamai per restituire all'LDNS gli indirizzi IP degli Edge server che dovrebbero garantire un download veloce e devono di solito trovarsi vicini al web client che ha iniziato la request.

- L'LDNS viene direzionato al name server akamai.net che comincia il processo di ricerca di un Edge server vicino inoltrando l'LDNS ad un DNS server Akamai di alto livello, ad esempio, uno nominato za.akamaitech.net.
- Un DNS server Akamai di alto livello è un elemento di un piccolo set globale di server DNS, responsabile di delegare un request DNS ad un appropriato DNS server Akamai di basso livello; generalmente un DNS server Akamai di basso livello è più vicino al LDNS di uno ad alto livello.
- Quindi, il DNS server Akamai di basso livello (attualmente chiamato usando il pattern n#g.akamai.net dove # è una cifra tra 0 e 9) ritorna gli indirizzi IP di due edge server che si aspetta offrano alta performance per il client. Le macchine che operano come DNS server Akamai di basso livello e come edge server possono essere (e di solito lo sono) le stesse.
- Alla fine, l'indirizzo IP dell'edge server viene restituito al web client, che è ignaro di tutte le redirezioni che si sono dovute effettuare.

*Dig* è un'utility che serve per sapere in che modo vengono tradotti i nomi di dominio, molto più sofisticata di *nslookup* ma anche molto chiara. Le entry dell'output sono: nome richiesto, TTL, tipo di entry, valore restituito. Il tipo di entry (che comincia sempre con IN: internet) può essere A (indirizzo IP), CNAME (canonical name, cioè un alias), NS (name server) o MX (mail exchangaer).

Con l'aiuto del comando *dig* si può vedere cosa succede:

---

```
gdb@kubuntu-vm:~$ dig www.adobe.com +trace
; <>> DiG 9.5.0-P2 <>> www.adobe.com +trace
      ; global options: printcmd
      . 779 IN NS f.root-servers.net.
      . 779 IN NS g.root-servers.net.
----->8----- snip -----
      . 779 IN NS e.root-servers.net.
;; Received 500 bytes from 193.204.161.85#53(193.204.161.85) in 6 ms
      ; info from local
      dns
      com. 172800 IN NS m.gtld-servers.net.
      com. 172800 IN NS l.gtld-servers.net.
----->8----- snip -----
      com. 172800 IN NS f.gtld-servers.net.
;; Received 491 bytes from 192.228.79.201#53(b.root-servers.net) in 195 ms
      ; info from
      root server b
      adobe.com. 172800 IN NS adobe-dns.adobe.com.
      adobe.com. 172800 IN NS adobe-dns-2.adobe.com.
      adobe.com. 172800 IN NS adobe-dns-3.adobe.com.
;; Received 155 bytes from 192.41.162.30#53(l.gtld-servers.net) in 135 ms
      ; info from
      generic top level
      domain server
      www.adobe.com. 3600 IN CNAME www.wip4.adobe.com.
      wip4.adobe.com. 86400 IN NS da1gtmoo1.adobe.com.
```

---

```
wip4.adobe.com. 86400 IN NS 3dns-5.adobe.com.  
;; Received 131 bytes from 192.150.22.30#53(adobe-dns-3.adobe.com) in 219 ms - info  
from adobe
```

---

Facendo di nuovo *dig* sull'indirizzo restituito nel record CNAME si può vedere come si viene effettivamente redirezionati verso Akamai:

---

```
gdb@kubuntu-vm:~$ dig www.wip4.adobe.com +trace  
; <>> DiG 9.5.0-P2 <>> www.wip4.adobe.com +trace  
      ; global options: printcmd  
      . 734 IN NS d.root-servers.net.  
----->8----- snip -----  
      . 734 IN NS c.root-servers.net.  
;; Received 500 bytes from 193.204.161.85#53(193.204.161.85) in 6 ms  
      com. 172800 IN NS b.gtld-servers.net.  
----->8----- snip -----  
      com. 172800 IN NS h.gtld-servers.net.  
;; Received 496 bytes from 192.228.79.201#53(b.root-servers.net) in 199 ms  
      adobe.com. 172800 IN NS adobe-dns.adobe.com.  
      adobe.com. 172800 IN NS adobe-dns-2.adobe.com.  
      adobe.com. 172800 IN NS adobe-dns-3.adobe.com.  
;; Received 160 bytes from 192.41.162.30#53(l.gtld-servers.net) in 139 ms  
      wip4.adobe.com. 86400 IN NS 3dns-5.adobe.com.  
      wip4.adobe.com. 86400 IN NS da1gtm001.adobe.com.  
;; Received 113 bytes from 192.150.22.30#53(adobe-dns-3.adobe.com) in 208 ms  
      www.wip4.adobe.com. 30 IN A 192.150.8.60  
;; Received 52 bytes from 192.150.22.46#53(3dns-5.adobe.com) in 209 ms
```

---

Akamai usa molto bene la gestione dei TTL. Usano TTL molto bassi per i Edge Server, che crescono in base alla gerarchia e a come salgono.

#### Problemi e tecniche generali delle CDN

Un modo per classificare le CDN è dire se viene eseguito partial content delivery o full content delivery. Nel primo caso si fa il primo hit all'origine, poi alcune URL sono costruite all'interno della pagina in modo tale che poi si entri nella CDN. Nella full content delivery già dal primo hit si entra nella CDN, il server di origine è invisibile (tranne che per la CDN) e tutte le richieste sono servite dalla CDN. Un altro modo di classificare le CDN è quello dei server surrogati, e anche qui si possono avere due approcci: multi-ISP e single-ISP, cioè si possono dislocare molti server in un *Point of Presence (PoP)* di un ISP oppure un po' meno server ma in più PoP possibili.

### URL rewriting

Anche questo può classificare le CDN. Mentre la maggior parte delle CDN si basa sui DNS, alcune fanno la riscrittura della URL. Il server di origine redireziona i client verso i vari server surrogati riscrivendo i link a URL delle pagine generate. Per automatizzare questo processo le CDN forniscono script speciali che elaborano il contenuto delle pagine e rimpiazzano le URL in maniera trasparente. Questa tecnica viene chiamata anche *content modification*.

### CDN peering

Le peer CDN consegnano i contenuti l'una per conto dell'altra. Una CDN può quindi espandere il suo raggio d'azione ad un insieme di client molto più ampio usando i server di CDN partner. Quando arriva una richiesta del client può non essere possibile trovare gli oggetti richiesti in nessun server surrogato. In questo caso, la CDN può richiedere l'oggetto ad una peer CDN che ha un contratto con la CDN originaria. Un fornitore di contenuti di solito stipula un contratto con una sola CDN, e la CDN stipula contratti con altre CDN per conto del fornitore di contenuti. In generale, le peer CDN consentono prestazioni migliori ad un costo relativamente più basso in confronto ad una singola CDN.

### Billing e Logging

CDN raccogliere i registri con le informazioni sulle richieste degli utenti, processa questi registri (per la fatturazione e la SLA), e consegna le informazioni di fatturazione ai clienti. La fatturazione richiede tutte le informazioni provenienti da ciascun server per ridurre il volume di informazioni da miliardi di righe di log per avere una sintesi mensile. Il ridimensionamento delle infrastrutture di fatturazione back-end è importante quanto scalare server-client fronte, visto il numero di server e clienti che cresce, così come la quantità di informazioni di log da elaborare.

## Il routing nelle overlay networks

### Reti di overlay

Una rete di overlay è un insieme di macchine ed un insieme di connessioni logiche tra queste macchine. Questo è un lavoro di astrazione sulle reti che si fa molto spesso: si parte da una rete fisica su cui c'è piena connettività e su di questa rete si costruisce una rete puramente logica, i cui nodi sono certe macchine collegate tra loro. Questi collegamenti seguono la topologia della rete fisica soltanto quando è conveniente. Ho una macchina da una parte della rete e una dall'altra e la connessione logica sfrutta la connettività fisica della rete sottostante; quando la connessione fisica cambia (perché le condizioni della rete cambiano), la connessione logica segue flussi diversi nel tempo. Questo è una rete di overlay, una rete costruita su un'altra rete esistente. Questo sforzo di astrazione possiamo farlo tutte le volte che vogliamo: possiamo creare una rete logica su una rete logica, e poi un'altra. Le  $n$  macchine di una rete di overlay, se usiamo il linguaggio delle reti p2p, si chiamano *peers*, e possiamo ipotizzare che queste macchine siano su internet (ma possono essere su una rete qualunque). In una struttura puramente logica come questa possiamo aspettarci una fluttuazione delle connessioni molto maggiore di quella che si osserva su una rete fisica e una modifica sulla rete logica è indolare rispetto ad una modifica su una rete fisica. Essendo poi delle reti logiche i peers possono entrare ed uscire continuamente. La presenza o l'assenza di un peer magari dipende dalla volontà di un utente di interagire con la rete (ma non è detto che un peer corrisponda ad un utente).

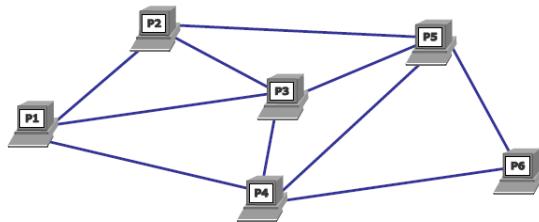


Figura 35 - Rete di overlay

### Primitive

Ci sono prima di tutto delle primitive amministrative, quelle che modificano l'infrastruttura logica:

- connessione alla overlay network (*joining*);
- disconnessione dalla overlay network (*departure*).

Ci sono poi le primitive di servizio, che tipicamente sono queste:

- richiesta di una risorsa;
- condivisione di una risorsa.

Finché c'è una rete di overlay non si può dire nulla sul piano applicativo; le primitive di servizio indicano che ci si sta focalizzando su una specifica esigenza applicativa come lo scambio di risorse. Questo è ragionevole, anche se la parola "risorsa" è talmente astratta che può essere usata in contesti di tipo più infrastrutturale (scalando il termine verso il basso). Assumiamo che ogni peer condivide un insieme finito  $O(1)$  di risorse, cioè una frazione costante delle risorse a cui siamo interessati. Questo significa che il quantitativo delle risorse a cui siamo interessati è lineare,  $O(n)$ .

### Efficienza delle operazioni

Si valuta in termini di:

- *Overhead di comunicazione.* Per noi è il numero di connessioni nella rete di overlay che l'informazione deve attraversare per portare a termine un'operazione, e questa valutazione del numero di connessioni la facciamo come notazione asintotica (in funzione di  $n$ , del numero di oggetti che sono messi a disposizione e/o del numero di macchine che fanno parte dell'infrastruttura, visto che come ordine di grandezza sono la stessa cosa). Uno potrebbe dire che dietro una connessione logica c'è una connessione fisica, un attraversamento della rete fisica, e non tutti gli attraversamenti della rete fisica sono equivalenti: questa però è una rete di overlay, e la guardiamo al giusto livello di astrazione, quindi pensare un termine "connessioni" a questo livello può avere un senso.
- *Risorse dei peers.* Stiamo parlando di quanta memoria è necessaria (eventualmente anche di più) per portare a termine l'operazione, e facciamo riferimento al peer più carico. Si possono sicuramente fare analisi più dettagliate; possiamo ad esempio far riferimento alla banda necessaria, tenendo presente che la banda complessiva è legata sicuramente al numero di connessione. Possiamo anche far riferimento alla banda del peer più carico.

### Architetture

Le reti di overlay possono essere strutturate oppure non strutturate (la rete, essendo costruita sopra quella fisica, ce la possiamo fare un po' come vogliamo). Se la rete non è strutturata la topologia della rete e la distribuzione delle risorse non sono tra loro correlate, quindi le risorse possono essere allocate ovunque. Di cosa è funzione quindi la topologia? La struttura cresce in maniera assolutamente caotica, in funzione della volontà dei peer di fare join o departure. La rete in alternativa può essere strutturata, per cui c'è una correlazione tra la topologia della rete e la distribuzione delle risorse. Se pensiamo che la risorsa sia una coppia  $\langle nome, indirizzo \rangle$  la struttura di overlay, che noi già conosciamo, e la struttura dei name server in Internet, è quella una struttura di overlay gerarchica. In una struttura di overlay gerarchica sappiamo a quale livello di gerarchia cercare l'informazione richiesta. La posizione delle risorse in una rete strutturata è calcolabile con un algoritmo. Gli algoritmi possono essere molto sofisticati, e sfruttare la topologia specifica che viene costruita proprio per rispondere a questa esigenza applicativa in maniera

molto furba. Che tipo di struttura prende una rete se vogliamo strutturarla? Ci possiamo aspettare, nelle reti strutturate, di ritrovare delle vecchie conoscenze, come le reti ad anello.

#### Approccio centralizzato - Napster

Un server centrale dispone di un grande indice di tutte le risorse reperibili nella overlay network che indica dove sono. Si richiede al server centrale chi possiede la risorsa, e la risposta da parte del server centrale da luogo ad una connessione, da parte mia, al peer specificato. Qual è l'efficienza della soluzione? I parametri che avevamo scelto di considerare sono overhead di comunicazione memoria.

Overhead di comunicazione: devo fare una connessione sul server centrale e una connessione a chi possiede la risorsa, quindi ci metto una "fatica" costante. Memoria: se c'è bisogno di un indice centralizzato, e le risorse sono dell'ordine di  $n$ , è evidente che il server centralizzato e quello che ha bisogno di più memoria, di una memoria che è dell'ordine di  $n$ . I peer eseguono search, join e leave sui server.

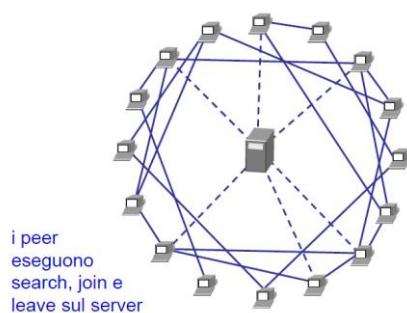


Figura 36 - Approccio centralizzato

#### Approccio distribuito: Gnutella

Se mi serve una risorsa la richiesta viene inviata a tutti i peers della rete, e la connessione viene fatta al peer che risponde positivamente. L'overhead di comunicazione è dell'ordine di  $n$ ; questa richiesta si sposta nella rete, e nel caso peggiore la attraversa tutta. Memoria:  $O(1)$ : qui non c'è bisogno di mantenere memoria oltre una quantità costante. Per essere più concreti: per implementare questa rete possiamo fare flooding (magari selettivo, con un identificatore della richiesta per evitare cicli) sulla rete di overlay. Per aumentare l'efficienza posso mettere dei TTL sulla richiesta, per limitare gli hop. Se però si dà una limitazione superiore al numero degli hop, se la rete ha una strutturazione che è più profonda di quanto si era pensato, si rischia di avere dei falsi negativi. Memoria  $O(1)$  significa che non si ha praticamente nulla in memoria. Io devo tenere in memoria il numero dei miei vicini, e possiamo assumere che il numero di peer vicini sia costante. È un approccio opposto a quello di Napster. I tratteggi nella figura rappresentano le richieste, gli archi a tratto pieno le connessioni, che hanno luogo solo quando una risorsa è stata identificata. Qui c'è una cosa tipica di tutte le infrastrutture: una fase di ricerca dell'informazione

(che segue in maniera più o meno rozza la topologia di overlay) e l'acquisizione di una risorsa, che avviene con un collegamento diretto con chi possiede la risorsa. Queste due fasi si trovano anche in contesti completamente diversi, come il VoIP.

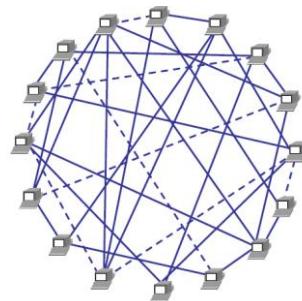


Figura 37 - Approccio distribuito (Gnutella)

#### Approccio misto

New-gnutella. Utilizza simultaneamente entrambi gli approcci. C'è un primo livello di peers, che si chiamano *supernodi*, che costituisce una rete distribuita, e un peer di secondo livello fa riferimento ad un peer di primo livello per le sue ricerche, tipicamente come nelle reti centralizzate. Assomiglia ancora all'approccio distribuito, ma scala un po' meglio. Ogni peer di primo livello gestisce un numero massimo di peer di secondo livello e c'è un nucleo di supernodi (che fanno da infrastruttura centrale della rete) e un insieme di nodi periferici che si connettono al proprio supernodo preferito. Come al solito, una volta identificato il possessore della risorsa ci si accede direttamente. Questa soluzione non è poi così intermedia, e solo una soluzione declinata in maniera più scalabile.

#### Utilizzo delle risorse

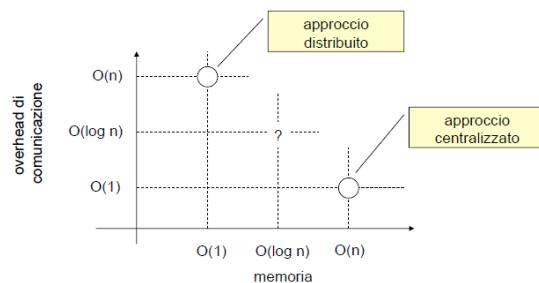


Figura 38 - Utilizzo di risorse

Come sono messi gli approcci dal punto di vista dell'uso di risorse? Ci piacerebbe trovare un'architettura che fosse, dal punto di vista della memoria e dell'overhead di comunicazione, con richieste di tipo logaritmico. La prospettiva logaritmica è quella delle *distributed hash tables*, DHT, che sono utilizzate nelle overlay networks strutturate (quindi si ha bisogno di struttura, per fare questo DHT).

## DHT

Sono strutture di dati distribuite per la memorizzazione di coppie *<identificatore, risorsa>*. Queste strutture fanno eseguire in modo efficiente operazioni di inserimento della risorsa, associata ad un certo identificatore, e ricerca della risorsa a partire dall'identificatore. Sono funzioni analoghe a quelle di una tabella hash, ma in ambiente distribuito. Per definire una DHT abbiamo bisogno di definire uno spazio di identificatori, e capire che rapporto c'è tra lo spazio degli identificatori di risorse e peers. Alla base di ogni DHT c'è uno spazio degli identificatori comune a risorse e peers, cioè ogni risorsa ha una un identificatore, ogni peer ha un identificatore, e gli spazi degli identificatori coincidono, sono fatti esattamente allo stesso modo.

Gli identificatori vengono spesso calcolati tramite funzioni hash del nome della risorsa e/o dell'indirizzo IP del peer. Essendo identificatori calcolati con funzioni hash non è nella struttura dell'identificatore che c'è la soluzione del problema che stiamo guardando. Le funzioni hash poi non sono invertibili, quindi se le informazioni originali servono, vanno salvaguardate. Ogni peer è autorità su una porzione dello spazio degli identificatori. Esattamente come nei DNS, solo che nei DNS l'informazione è strutturata in modo gerarchico, qui no.

### Memorizzazione diretta e indiretta

- Memorizzazione diretta. Al momento dell'inserimento di una risorsa, questa viene memorizzata effettivamente nella DHT, viene cioè copiata nel peer che è autorità sul suo ID anche se è diverso dal peer che ha condiviso la risorsa. Io entro sulla rete con una risorsa e un certo ID, e c'è un id di quella risorsa. Capita che io non sia, per come è strutturata la rete, autorità sull'ID di quella risorsa. Che faccio? Esistono due possibilità: prendo questa risorsa e me ne privo, allocando fisicamente la risorsa a quel peer che ha autorità sull'id corrispondente.
- Memorizzazione indiretta: ciò che inserisco, l'informazione che metto sul peer che è autorità di quella risorsa, non è la risorsa, ma l'informazione di chi possiede la risorsa, e la risorsa la lascio fisicamente nel peer che l'ha messa a disposizione. In altri termini, la DHT memorizza solo il riferimento alla risorsa cercata, per esempio l'indirizzo IP del peer che la detiene. I peers sono responsabili della memorizzazione delle risorse che hanno inserito nella DHT. Naturalmente questo implica un passo in più per accedere alla risorsa, perché si deve acquisire l'informazione di chi la possiede, poi si accede direttamente al peer. Si può aumentare il livello di indirezione: si puoi mettere a disposizione di chi cerca il peer che possiede la copia che identifica chi possiede la risorsa. Si può scalare un numero arbitrario di volte.

## Qualità delle DHT

La qualità delle DHT viene misurata in termini di:

- *efficacia*: reperimento della risorsa richiesta ogni volta sia presente nella struttura distribuita.
- *scalabilità*:
  - overhead di comunicazione  $O(\log n)$ ;
  - memoria dei peers  $O(\log n)$ ;
- *robustezza*: gestione dei nuovi peers (*joining*) e gestione delle disconnessioni (*departure*).

## Algoritmi di routing basati su DHT

### Chord

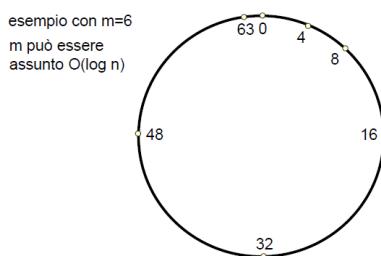


Figura 39 - Chord

Definisce le operazioni

- *insert(id, item)*: aggiunge una nuova risorsa con l'ID specificato;
- *lookup(id)* : ritorna il peer autorità sull'ID specificato ed equivale ad una memorizzazione doppiamente indiretta: ti ritorna il peer autorità sull'ID specificato. Il peer autorità a questo punto ha la coppia  $\langle id, \text{dov'e la risorsa} \rangle$  e va a cercare la risorsa. In Chord lo spazio di indirizzamento degli ID è un intervallo unidimensionale da 0 a  $2m-1$ , con  $m$  che va scelto a priori. Prendiamo ad esempio  $m=6$  (Chord usa  $m=160$ ).

Questo spazio dei nomi è lo spazio dei peer che partecipano a questo gioco. Strutturalmente queste macchine sono pensate come distribuite su un anello, e la posizione sull'anello è specificata dall'identificatore attribuito alla macchina. Se c'è un numero abbastanza alto di macchine per coprire questo spazio possiamo assumere che  $m$  sia pari a  $O(\log n)$ . Non si passa da un  $m$  all'altro in maniera indolore, perché le macchine devono ricostruire un po' di informazioni. Questa infrastruttura è dinamica rispetto alla possibilità di entrare e uscire. Abbiamo detto che ci sono dei nodi che sono autorità per un certo numero di informazioni che ci dicono dove sono le risorse. Ogni peer qui è autorità sul proprio ID, e su tutti gli ID minori di esso fino al peer predecessore. Il nodo che ha autorità sull'ID  $x$  è denominato *successor(x)*.

La composizione dell'anello può cambiare continuamente; l'inserimento di un nuovo peer cambia i nodi autorità. Perché  $\text{successor}(x)$ ? Perché se sei 18, chi ha l'autorità su 18? In questo schema di numerazione è 32. Un peer conosce i possessori delle risorse che hanno un ID che cade nell'intervallo sul quale è autorità. Un peer conserva un puntatore al successore e un puntatore al predecessore; questi puntatori servono sia per il routing della ricerca sia per verifiche di consistenza, eseguite nel momento in cui si fa join-departure di qualche peer.

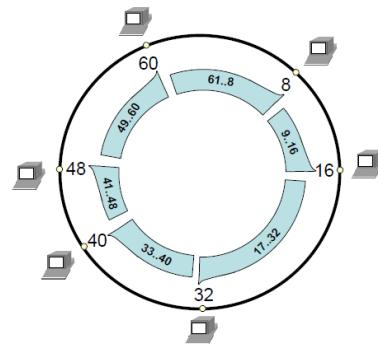


Figura 40 - Esempio di algoritmo di Chord

#### Algoritmo di routing

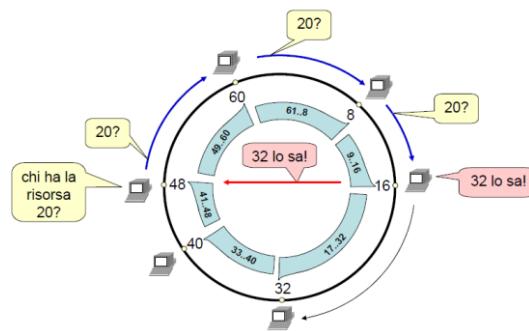


Figura 41 - Algoritmo di routing

Si inoltra la richiesta  $\text{lookup}(\text{id})$  al successore finche il successore è minore o uguale ad id altrimenti si ritorna il successore.

Questo è l'algoritmo banale. Sicuramente è facile da implementare, e la memoria necessaria ad ogni peer è costante. È però poco robusto (un solo peer può disconnettere l'anello) e poco efficiente (in media  $n/2$  hop, che diventano  $n-1$  nel caso peggiore, con un overhead di comunicazione  $O(n)$ ). L'algoritmo che si usa su Chord è basato su finger tables. Ogni peer ha una tabella chiamata finger table di dimensione  $\log n$ . Supponiamo che un peer abbia  $\text{id} = k$ ; per  $i = 0, \dots, \log n$ , l'entry  $i$ -esima della finger table è il peer che ha autorità sull' $\text{id}$   $\text{start} = (k + 2^i) \bmod n$ . La prima riga contiene il successore di  $k$ .

i	start	successor
0	$13 + 2^0 = 13+1=14$	autorità su 14
1	$13 + 2^1 = 13+2=15$	autorità su 15
2	$13 + 2^2 = 13+4=17$	autorità su 17
3	$13 + 2^3 = 13+8=21$	autorità su 21
4	$13 + 2^4 = 13+16=29$	autorità su 29

Figura 42 - Finger Table

In pratica parto da me, e stimo esponenzialmente la distanza dalla risorsa (dimezzo, se necessario dimezzo ancora, e ancora) dando luogo ad un andamento logaritmico.

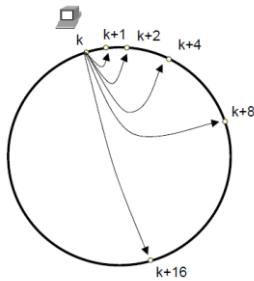


Figura 43 - Andamento logaritmico della richiesta

Il peer che ha autorità su un ID ha almeno il valore di quell'ID, quindi il peer che ha autorità su  $k+x$  è almeno a distanza  $x$  da  $k$ . Ogni peer inoltra la richiesta riguardo alla risorsa  $h$  al peer più lontano della sua finger table che precede  $h$  (in pratica salta l'intervallo utile più lungo). L'algoritmo termina quando l'entry della prima riga della finger table (cioè il successore del peer corrente) è maggiore di  $h$ . In questo caso il successore del peer corrente è anche autorità su  $h$ .

Esempio

### Primo passaggio

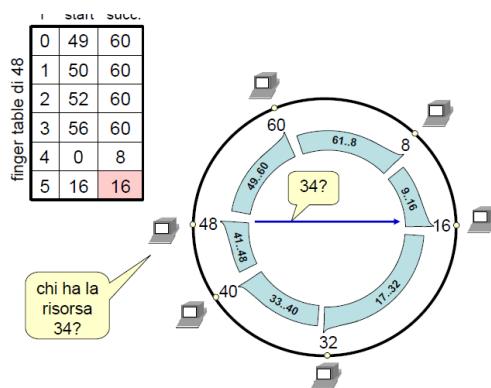


Figura 44 - Primo passaggio

48 inoltra la richiesta lookup(34) a 16. 16 è l'ultimo valore che precede 34 nella colonna successore della finger table.

### Secondo passaggio

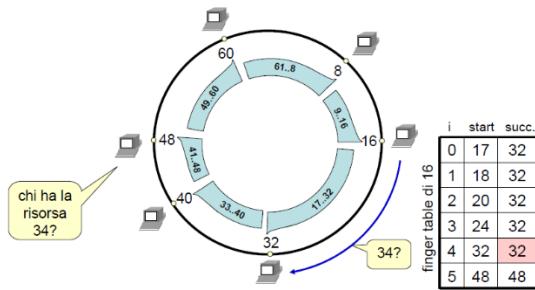


Figura 45 - Secondo passaggio

16 inoltra la richiesta lookup(34) a 32. 32 è l'ultimo valore che precede 34 nella colonna successor della finger table.

### Terzo passaggio

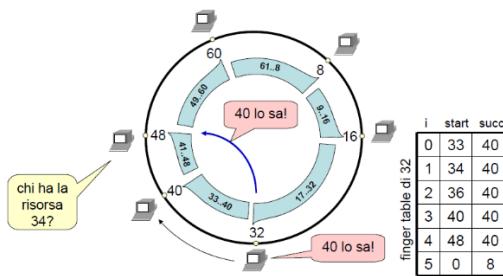


Figura 46 - Terzo passaggio

32 non ha nella finger table valori minori di 34. Il successore di 32 è il peer che ha autorità su 34. 32 risponde a 48.

Analisi:

- Overhead di comunicazione.
  - Osservazione: non può essere saltato due volte un intervallo della stessa dimensione (cioè corrispondente allo stesso numero di riga della finger table);
  - Caso peggiore: la risorsa h precede immediatamente il nodo k che la sta cercando;
  - Il numero di passaggi è al massimo  $\log n$ .
- Memoria utilizzata dai peers: la finger table è lunga  $\log n$ .

### Verifica di consistenza

Chord non è “perfettamente consistente”: ci possono essere degli stati in cui la dinamica non funziona perfettamente. Su questa rete abbiamo una situazione del tipo: l’anello, e per ciascuna macchina successore e predecessore. Un puntatore in avanti e un puntatore all’indietro così è facile, ma questo è un mondo altamente dinamico, in cui macchine arrivano ed escono continuamente, quindi si potrebbero avere situazioni di inconsistenza, anche soltanto per il fatto che dal punto di vista delle transazioni di

ingresso e di uscita non c'è una verifica particolarmente puntuale. Quello che succede però è che ogni peer verifica la consistenza con il suo successore inviandogli periodicamente un messaggio di *stabilizzazione* (*stabilize()*).

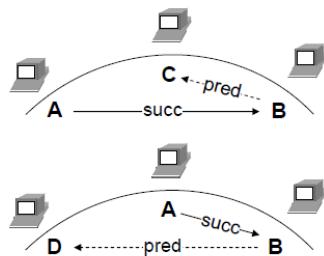


Figura 47 - Verifica di consistenza

Nella figura, A invia il messaggio si stabilizzazione a B. Adesso ci sono vari casi possibili.

Se  $\text{pred}(B) = A$  : B sa qual è il suo predecessore, tramite un puntatore. Il predecessore si B e A, il successore di A e B, quindi B invia un messaggio di notify ad A dicendo "I puntatori sono consistenti, predecessore e successore sono definiti in modo consistente". La situazione però potrebbe essere diversa: il predecessore di B potrebbe essere C, che è maggiore di A, cioè c'è una macchina in mezzo di cui A non ha tenuto conto. A questo punto B invia ad A un notify di C dicendo "I puntatori sono inconsistenti", e A aggiorna il suo successore a C, cioè prende coscienza di questo fatto e a questo punto A procede di nuovo alla verifica di consistenza con C. Se invece il predecessore di B è D, che è minore di A (cioè siamo saltati all'indietro) B aggiorna il predecessore di B ad A, e B invia ad A un messaggio di notify. A questo punto i puntatori di A e B sono consistenti, ma il puntatore di D potrebbe essere inconsistente con B, e quello che succede è che si spera che D scopra questa cosa alla prossima verifica di consistenza con B, in cui i conti non tornano e a questo punto sarà la macchina B stessa a dire alla macchina D che c'è un successore diverso da considerare. Questi sono i giochi di puntatori che fanno queste macchine per tenersi dentro al cerchio, in modo consistente. A questo punto come avviene l'aggiunta di un nuovo peer? Il nuovo peer può entrare nella rete, che magari suppone sia consistente. Il nuovo peer deve conoscere almeno un peer della rete, e questo è tipico di tutte queste reti di overlay: si suppone di avere un bandolo della matassa, una tra tutte le macchine e nota, e si entra su quella.

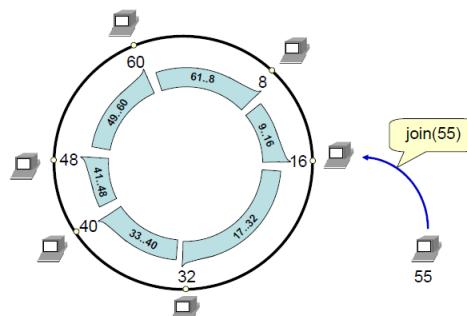


Figura 48 - Inserimento di un nuovo peer

Qui c'è 55 che vuole fare un join sulla rete rivolgendosi a 16, quindi 55 invia a 16 un  $\text{join}(55)$ . A questo punto bisogna andare a vedere chi è che ha autorità su questo numero, e infilarsi in mezzo; quanto è difficile questa operazione? Dal punto di vista della rete è equivalente a fare un lookup su 55. In questo caso non puoi sapere l'informazione 55, ma puoi sapere chi e l'autorità su 55. Quindi dal punto di vista della complessità questa operazione ha una complessità logaritmica, perché il lookup ha una complessità logaritmica, e in più c'è l'aggiornamento dei puntatori. A questo punto c'è l'aggiornamento del puntatore al successore e la verifica di consistenza, come abbiamo visto prima: aggiorni il puntatore al successore e hai che il predecessore lancia la sua solita verifica di consistenza, le cose non vanno a buon fine e vieni rialacciato al nuovo puntatore. Ricapitolando l'aggiunta di un nuovo peer:

- Il nuovo peer deve conoscere almeno un peer della rete – supponiamo che il nuovo peer 55 conosca 16.
- 55 invia a 16  $\text{join}(55)$ .
- 48 risponde a 55 che 60 ha attualmente autorità su 55 (stesso meccanismo del lookup).

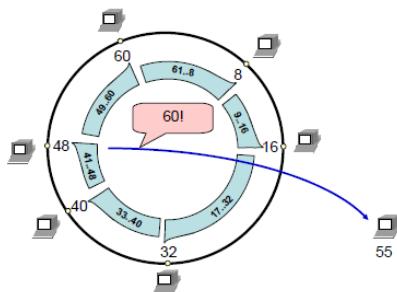


Figura 49 - Aggiunta di un nuovo peer – aggiornamento successivo di 55

- 55 aggiorna  $\text{succ}(55) = 60$ .
- 55 lancia una verifica di consistenza con 60.

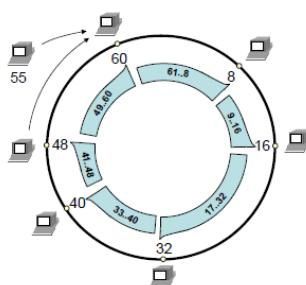


Figura 50 - Aggiunta di un nuovo peer - verifica della consistenza

- L'aggiornamento sarà completo quando:
  - o la verifica di consistenza con 60 avrà aggiornato  $\text{pred}(60) = 55$ ;
  - o 48 lancerà una verifica di consistenza con 60.

Una volta entrati si deve calcolare la finger table. Quanto è difficile calcolare la finger table? Quando sei la k-esima stazione della rete, cosa devi fare? Devi fare  $k+1, k+2, k+4, \dots$  sono addizioni, non ci vuole niente a farle. Il problema è capire poi chi sia autorità sul “punto d'atterraggio”, ma quello è un lookup, quindi quelle sono tutte attività che richiedono uno sforzo logaritmico. Le cose quindi riescono ancora a funzionare decentemente. Ricapitolando:

- Calcolo della colonna start.
- Ricerca di  $\text{lookup}(\text{start}[i])$  per ogni i.

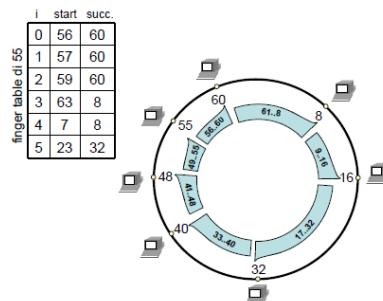


Figura 51 - Creazione della finger table

Si possono fare DHT in maniera ancora più “stravagante”, in particolare si possono usare gli approcci che si usano in CAN e Kademlia.

#### Can

CAN sta per *Content Addressable Network*. Qui l'idea è abbastanza semplice: lo spazio degli identificatori è un ipercubo, e ad ogni valore dell'identificatore è associato un punto dello spazio d-dimensionale, dove d è scelto opportunamente:

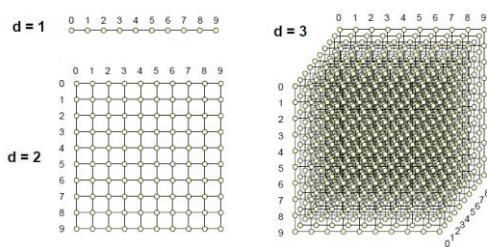


Figura 52 - Can - Spazio degli identificatori

Normalmente hai un identificatore, con l'approccio di Chord, che è un numero in uno spazio unidimensionale; qui invece diventa una coppia, una tripla, una n-upla a d valori in uno spazio d-dimensionale. A questo punto lo spazio degli identificatori viene partizionato in zone, e due zone sono adiacenti se condividono più di un punto, e qui per esempio:

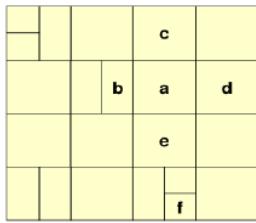


Figura 53 - Can - Partizionamento dello spazio

a e adiacente a b, d, e, c. Ciascuna zona naturalmente ha un'autorità. A che serve l'adiacenza fatta in questo modo? Perché è interessante sapere chi è adiacente a chi? Perché si può utilizzare questa adiacenza per esplorare questo spazio, per andare a cercare chi è autorità per una certa informazione. Attenzione: si dice che gli identificatori sono di natura diversa, ma detto così potrebbe essere un po' fuorviante; non è che sono di natura diversa, hanno funzioni un po' diverse gli identificatori delle risorse e gli identificatori dei peer. Lasciamo questa cosa in sospeso, e vediamo a cosa corrispondono gli identificatori attraverso l'evoluzione del partizionamento.

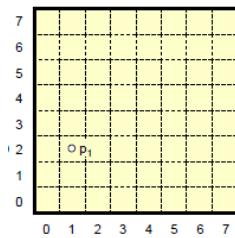


Figura 54 - Evoluzione dello spazio - 1

Questa è la situazione iniziale, l'ipercubo gigante quando non c'è nulla, ma c'è un solo peer, e tutto lo spazio è affidato a lui. Come è fatto questo spazio? Questo è lo spazio degli identificatori, quindi ci sarà l'identificatore 2, 4, 5, 7; sono tutti identificatori possibili, e a ciascuno di questi può corrispondere una risorsa. Come al solito non è detto che le risorse ci siano tutte, c'è un peer che corrisponde ad una casella di questo spazio, e tutto lo spazio è affidato a lui. Questa è la stessa cosa che si fa su Chord, ma invece di pensare ad un unico segmento qui si ha a che fare con una cosa che ha una dimensione diversa.

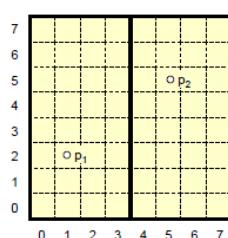


Figura 55 - Evoluzione dello spazio - 2

Adesso arriva una nuova macchina, p<sub>2</sub>, si estrae una zona a caso, si prende la zona e la si divide in 2, e si assegna a p<sub>2</sub> un identificatore random nella sua zona. Qui il processo di costruzione dell'identificatore da

associare alla macchina e un processo che è funzione del punto in cui andrà a finire la macchina, e questo è un pochino diverso da Chord.

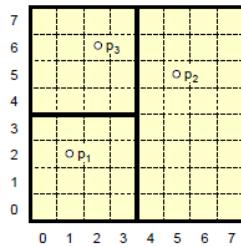


Figura 56 - Evoluzione dello spazio - 3

Di nuovo: arriva p<sub>3</sub>, si estrae una zona a caso e la si divide in 2, e si associa a p<sub>3</sub> un identificatore random nella zona. Questo vuol dire che si taglia questo ipercubo a pezzi, ogni volta selezionando una macchina che è autorità per quella porzione. Dopo molti join lo spazio degli indirizzi potrebbe apparire così:

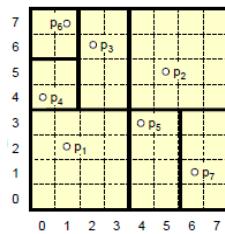


Figura 57- Evoluzione dello spazio - 4

Uno può sperare che se il processo di randomizzazione è uniforme il taglio di questo spazio sia un taglio ragionevolmente uniforme. Come funziona il meccanismo? Il peer che fa join deve conoscere almeno un peer della overlay network, però questo è tipico delle overlay network, e ne abbiamo appena parlato sul piano (diverso) di Chord. Questo si chiama normalmente *problema del bootstrap*. Cosa fa il nuovo peer? Estrarre un id temporaneo random e cerca il peer che è autorità su quell'id. Questo è un ID temporaneo che serve soltanto ad accedere alla specifica porzione dello spazio degli indirizzi, e per cercare la zona giusta si utilizzano le funzionalità delle DHT. A questo punto è stata trovata una zona, con dentro un peer; si prende la zona e la si divide in sotto-zone, e si assegna al peer l'ID definitivo della sotto-zona. Se ci pensiamo un attimo si poteva fare una cosa simile anche per Chord: in Chord quando uno arriva ha già un ID, si tiene quell>ID e si va a collocare nel posto giusto. Però si potrebbe fare una cosa: quando è stato collocato nel posto giusto si potrebbe prendere l'insieme degli ID che sono di autorità per quella specifica porzione dello spazio e si potrebbe tagliare in maniera specifica, magari un po' uniforme.

*Suddivisione di una zona*

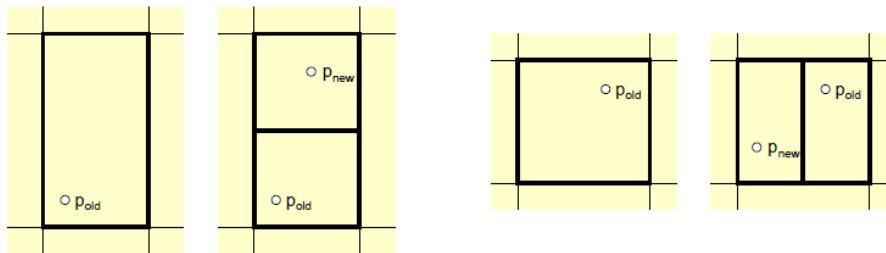


Figura 58 - Suddivisione di una zona

Come avviene la suddivisione di una zona? La suddivisione di una zona è fatta in modo da rendere il recovery più facile dopo una departure, e in particolare si fa una prima suddivisione orizzontale, poi verticale, poi orizzontale e così via, in modo da suddividere in modo omogeneo lo spazio. Naturalmente le zone più grandi sono interessate da un join con maggiore probabilità, e quindi a questo punto il partizionamento dovrebbe tendere a essere omogeneo. Supponiamo che lo spazio d-dimensionale sia suddiviso in maniera omogenea. A questo punto ogni peer ha  $2d$  vicini, perché se  $d$  sono le dimensioni quello ha adiacenza verso l'alto e verso il basso. Se scegliamo:

$$d > \frac{\log_2(n)}{2}$$

---

Si potrebbe anche avere semplicemente  $\log(n)$ , tanto quanto che si cerca è un asintoto, le informazioni di adiacenza occupano una memoria che è  $O(\log_2 n)$ , e quindi a questo punto ho raggiunto l'obiettivo di una memoria logaritmica da occupare. Per praticità si usa una dimensione  $d$  fissa, e indipendente dalla dimensione della rete di overlay, però se ci pensiamo un attimo, anche per Chord, quando abbiamo scelto  $m$ , abbiamo detto che si può scegliere  $m$  in modo che torni nei punti logaritmici, ma che  $m$  tipicamente è una costante (ad esempio 160), un numero che si sceglie di dimensionare in modo corretto rispetto alla struttura della overlay network. Adesso, quando si fa routing su questa rete lo si fa in maniera greedy: per raggiungere il peer autorità su un id si va su un peer che si conosce e ci si muove nello spazio delle adiacenze finché non si trova il peer giusto. Adesso, al crescere di  $n$ , la lunghezza media del cammino cresce come  $O(n^{1/d})$ .

### Kademlia

Lo spazio degli identificatori è costituito da chiavi di 160 bit. Questi identificatori vengono messi in corrispondenza delle foglie di un albero binario in base al loro prefisso unico più corto. Quello che potremmo fare è prendere queste stringhe di 160 bit e metterle su un albero binario (completo); questo è un po' uno spreco, perché è vero che questo è uno spazio parecchio popolato di oggetti, però 160 bit sono tanti. A questo punto che cosa si può fare? Invece di rappresentarlo tutto ci si limita a rappresentarne la porzione effettivamente necessaria.

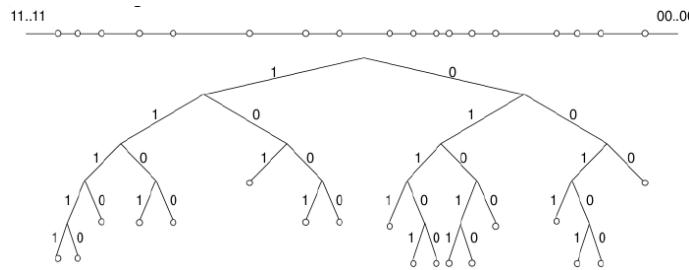


Figura 59 - Kademlia: spazio degli identificatori

Nell'albero in figura c'è un oggetto (per prefisso si intende qualunque cosa, anche peer) il cui prefisso è 101. Se non c'è nessun altro oggetto che inizia con 101 chi me lo fa fare a rappresentare tutto il sotto-albero? Quello è l'unico oggetto che ha prefisso 101. Se ce ne fosse un altro con quel prefisso si allungherebbe di un pezzettino quel sotto-albero tirando fuori il prefisso unico più corto relativo a quegli oggetti. Che ci faccio con questa cosa? Definisco un concetto di distanza; qui la distanza tra due identificatori  $x$  e  $y$  è misurata con un'operazione di XOR bit a bit:

$$d(x, y) = x \oplus y$$

Ad esempio, la distanza tra 010101 e 110001 è  $d(010101, 110001) = 100100$ . Se definisco la distanza tra due identificatori in questo modo i nodi nello stesso sotto-albero sono più vicini tra loro rispetto a quelli di altri sotto-alberi in quanto hanno lo stesso prefisso. Questa è l'idea di vicinanza dei sotto-alberi a cui si appartiene. Un peer è autorità su tutti gli id che sono più vicini al peer stesso rispetto ad altri peer (sei autorità per un certo "vicinato"). Guardiamo la cosa qui:

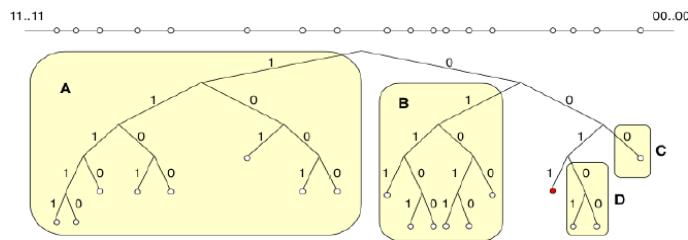


Figura 60 - Kademlia: sottoalberi

pensiamo ad un peer con prefisso (in questo caso unico) 0011. Questo peer non ha certamente autorità sui sotto-alberi A, B, C, D. Quello che succede e che qui faccio si che quel peer comunica ad almeno un altro peer per ognuno dei sotto-alberi che contengono risorse di cui questo peer non è autorità. Basta che quel peer ne conosca un altro. A questo punto se qualcuno deve fare una ricerca, e si rivolge a me, se io sono autorità per quella risorsa allora rispondo direttamente. Altrimenti, quello che succede e che io so qual è il sotto-albero in cui è allocata quella risorsa, perché né conosco l'identificatore. Supponiamo allora che io scopra che la risorsa è allocata in un dato sottoalbero, siccome io so che c'è almeno un peer lì dentro che io conosco allora invio la richiesta a quel peer.

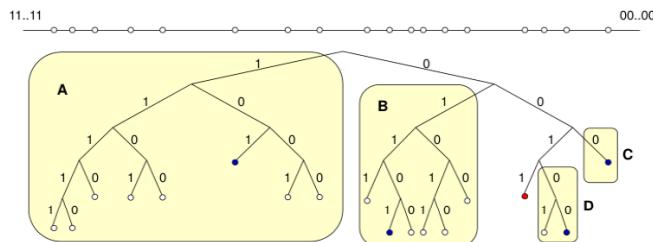


Figura 61 - Kademlia: adiacenze

Questo vuol dire che o io rispondo subito, perché sono autorità per quella risorsa, oppure se non ho la risposta posso dare come risultato della richiesta un signore che "ne sa di più", contribuisce ad essere autorità di una zona che è la metà dello spazio in cui stai cercando. Quest'altro peer potrà anche lui rispondere immediatamente oppure dare un riferimento a chi si occupa di un altro sottoalbero. Questo implica che ogni volta che fai una re-direzione di una ricerca dimezzi lo spazio di ricerca, e questo sa tanto di una ricerca che richiede un numero logaritmico di connessioni. Naturalmente si deve fare un management per fare in modo che un peer conosca almeno un peer in ciascun sotto-albero. Se si riesce a fare questo si riesce a rispondere in modo logaritmico, e anche la memoria necessaria per ciascuno dei peer è  $O(\log n)$ . Tutto questo naturalmente ha un senso se questi alberi sono ragionevolmente bilanciati; se sono completamente sbilanciati c'è poco da fare, ma qui c'è una funzione hash che fa distribuire questo spazio in maniera abbastanza uniforme. Kademlia è un meccanismo per la gestione dello spazio monodimensionale che assomiglia moltissimo a Chord: in Chord lo spazio degli identificatori viene rappresentato come un cerchio, qui come un albero, ma non è che ci sia tanta differenza.

## Architettura di un internet data center

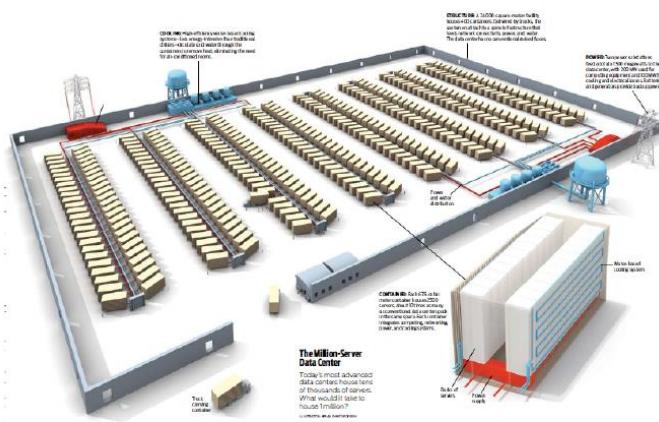


Figura 62 - Architettura di un internet data center

I principali servizi erogati in un internet data center sono hosting e housing; quando si parla di hosting si parla di:

- fornitura e manutenzione dell'hardware e del software necessari per gestire un server Web alimentato con pagine prodotte dal cliente;
- configurazione iniziale dei server: hardware, software di base e web server;
- user administration (a livello di sistema operativo);
- mantenimento del controllo degli accessi e fornitura delle autorizzazioni per accessi;
- il ruolo del cliente e produrre delle pagine e di metterle a disposizione.

Nell'housing invece il cliente porta dentro l'internet data center le proprie apparecchiature, che ovviamente devono avere delle caratteristiche standardizzate. Chi offre housing mette anche a disposizione l'interconnessione all'infrastruttura di connettività. L'housing ultimamente sta cambiando pelle: al posto delle macchine fisiche si stanno fornendo macchine virtuali, e questo consente di scalare di più. Le caratteristiche fisiche di un internet data center sono:

- alimentazione elettrica:
  - diverse linee di alimentazione elettrica, tra loro indipendenti, per ogni armadio rack;
  - disponibilità di UPS per proteggere da cali o mancanza di tensione, ecc;
  - gruppi elettrogeni di emergenza.
- sistemi di condizionamento ridondanti;
- sistemi antincendio:
  - sistemi di rilevazione dei fumi basati rilevatori;
  - sistemi di spegnimento.
- sistemi antiallagamento (spesso, ma non sempre);
- sorveglianza h24.

Quanto all'housing, perché portare le proprie macchine in un internet data center? Si può offrire il servizio anche da casa, ma in presenza di requisiti di prestazioni, scalabilità e via dicendo si deve ricorrere per forza ad un internet data center. Quanto alla connettività esterna si dispone di linee ad alta velocità ridondate, di un *autonomus system* e di un ampio insieme di prefissi, per poter scalare rispetto all'utenza; tipicamente, se l'utente è importante non si accontenta di un indirizzo, ma vuole un prefisso. L'affidabilità si ottiene spesso con dei *peering multi homed*: abbiamo un data center in un certo posto e vogliamo avere connettività. La cosa migliore da fare è interagire con almeno due provider, cosicché, in caso di fault della connessione verso un provider i servizi vengono offerti attraverso l'altro provider.

### Sicurezza logica

La sicurezza la si ottiene in questo modo: al primo nodo si ha un firewall d'accesso. I firewall d'accesso sono tipicamente firewall ridondati, e sono ridondati con le stesse tecniche viste per i servizi web based: ci sono dei bilanciatori di carico che mandano le richieste ad un firewall oppure ad un altro. Questo consente di avere alti livelli di disponibilità: se si vuole ad non ci si possono permettere sui firewall d'accesso neppure interventi di manutenzione programmata; se si ha un solo firewall non posso pensare di sottoporlo a manutenzione.

Ci sono poi degli *intrusion detection system*: dei sistemi che cercano di capire se c'è un intervento malevolo in corso; gli *intrusion prevention system* cercano invece di prevenire il comportamento malevolo. Gli *intrusion detection system* possono essere dislocati nell'architettura in vari posti, ad esempio subito dopo il firewall d'accesso o nella rete di un cliente.

Se ci sono vari clienti, le loro reti devono rimanere separate; si ha bisogno di una LAN di ogni cliente, poi una LAN per gestire tutte queste reti. In realtà i clienti hanno bisogno di più LAN, almeno 3-4. Come si fa a gestire una cosa del genere? Non si può pensare di stendere e togliere cavi continuamente, perciò le LAN che vengono offerte ai clienti sono LAN virtuali. Naturalmente ci sono moltissimi apparati di networking, come switch e router, configurati per essere acceduti solo via SSH.

La sicurezza logica è affidata al SOC (Security Operations Center), tipicamente un gruppo di persone esperte di sicurezza che dovrebbero presidiare l'internet data center 24x7.

### Sicurezza fisica

Accanto alla sicurezza logica si deve affiancare anche la sicurezza fisica, con controlli perimetrali e con controlli sempre più specifici mano a mano che ci si avvicina ai server. Un esempio di sala di massima sicurezza è quella delle PKI (*public key infrastructure*); in queste infrastrutture c'è bisogno di un posto fisico in cui sia contenuta una certa "chiave", che non deve essere vista da nessuno. Normalmente questi sono luoghi presidiati con particolare cura. Accanto al SOC c'è tipicamente anche un NOC (Network Operations Center), un gruppo di persone che si occupa della rete. Il NOC fa provisioning, configurazione

e monitoraggio della rete e dei sistemi. Che vuol dire provisioning? La prima cosa da fare per mettere insieme un internet data center è comprare delle apparecchiature, che vanno configurate e monitorate. C'è bisogno anche di identificare e risolvere i fault nonché di curare i rapporti con l'utenza. Il rapporto con l'utenza viene curato dal NOC attraverso un call center chef a customer care. L'help desk che risponde alle chiamate è tipicamente gerarchico: si parla con delle persone che non sono molto preparate, ma sono in grado di capire a chi girare la chiamata. Questo permette di scalare la chiamata al supporto applicativo, al NOC o al SOC. La gestione dei guasti o dei problemi avviene tramite dei trouble ticket, in cui vengono scritti tutti i passaggi di lavorazione del guasto o del problema. Questo è un buon modo per tracciare il singolo guasto e l'effettivo funzionamento dell'internet data center, ma anche per verificare se la QoS prevista negli SLA contrattuali è stata raggiunta oppure no. I trouble ticket sono spesso usati per determinare se certi SLA sono stati raggiunti oppure no, e, se non sono stati raggiunti, di applicare delle penali. Anche per gli internet data center c'è uno standard, ed è lo standard TIA-942 (*Telecommunication industrial association*). Questo standard definisce una serie di linee guida per la progettazione e la costruzione dei data center per quanto riguarda i sistemi di cablaggio e la progettazione delle reti. Dice come deve essere fatto il cablaggio, e specifica i principali punti di riferimento per l'affidabilità da usare in chiave contrattuali. Specifica anche caratteristiche ambientali, come la refrigerazione degli ambienti. Di recente gli aspetti relativi al consumo energetico sono diventati preponderanti, e vengono costruiti edifici che consentano di abbattere i consumi. Una tendenza attuale è una cosa che somiglia alla corsa agli armamenti della guerra fredda. Amazon, Google ed altri big player stanno costruendo enormi data center, dove c'è spazio, acqua per la refrigerazione, connettività via fibra ottica, ma soprattutto energia a basso costo. Sono in grado di spostare l'elaborazione, in una data fascia oraria, nei posti in cui in quella fascia oraria l'elettricità costa meno. Il data center è fatto di container, e ogni container è una macchina autonoma. Quando serve altra capacità di calcolo viene posto un altro container. Si può spostare la refrigerazione in uno spazio molto ristretto.

## Socket

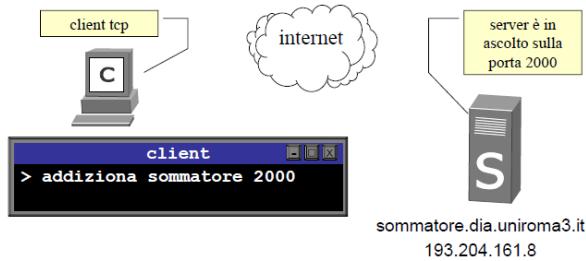


Figura 63 - Esempio Socket

Stiamo percorrendo il confine tra l'applicazione e la rete, adesso guardiamo proprio il dettaglio di questo confine. Il punto di vista diventa il punto di vista del programmatore che scrive codice e deve funzionare in rete.

Le *socket* sono delle API (*Application Programmer Interface*) che consentono ai programmatori di gestire comunicazioni tra processi (*inter-process communication*), nascondendo all'utente il livello di trasporto. A differenza degli altri costrutti di comunicazione (pipe, code di messaggi e memoria condivisa) le socket consentono il colloquio tra processi che risiedono su macchine diverse, e dunque costituiscono lo strumento di base per realizzare un servizio di rete. Si possono usare sia per TCP che per UDP, e nascondono al programmatore tutti i dettagli del livello di trasporto (three-way handshake, finestre, segments, sequence numbers, ecc). Come è fatta una comunicazione TCP?

- dichiarazione al sistema operativo che si intende instaurare nel seguito una connessione (specifica delle caratteristiche). Stiamo chiedendo al sistema operativo delle porzioni di memoria in cui andremo ad immagazzinare le code di pacchetti da inviare.
- apertura della connessione: diversa nel server e nel client: il server specifica la porta della macchina su cui risiede e rimane in attesa che un client si connetta, il client esegue un tentativo di connessione specificando indirizzo e porta del server.
- scambio di dati bidirezionale (trasmissione e ricezione).
- chiusura della connessione.

## Client

Supponiamo che sia disponibile un server TCP sulla macchina *sommatore.dia.uniroma3.it*, con indirizzo IP 193.204.161.8 , in ascolto sulla porta 2000. Il server riceve una sequenza di numeri interi; per ogni numero ricevuto restituisce la somma del numero con tutti i precedenti. Vogliamo realizzare un client che sfrutta il servizio offerto dal server; vogliamo specificare sulla linea di comando l'indirizzo del server e la porta su cui è in ascolto, in modo da poter riusare il client anche per altri server analoghi. Il client si chiama “addiziona”, e viene lanciato così:

---

C:\>addiziona 193.204.161.8 2000

---

oppure così:

---

C:\>addiziona sommatore.dia.uniroma3.it 2000

---

### Schema del client TCP



Figura 64 - Schema del client TCP

In particolare, la dichiarazione e l'inizializzazione è strutturata:

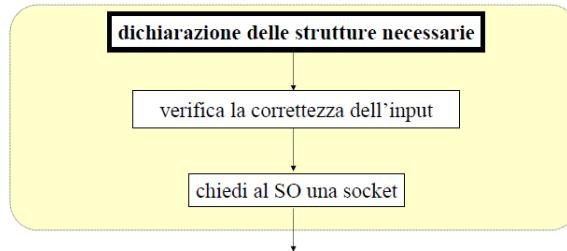


Figura 65 - Struttura dichiarazione e inizializzazione

Client in C

```
#include <stdio.h> /* per usare input-output */
#include <sys/socket.h> /* per usare socket */
#include <stdlib.h> /* per usare exit() */
#include <errno.h> /* gestione degli errori */
#include <netdb.h> /* per gethostbyname() */
#include <string.h> /* per memcpy() */
int main(int argc, char** argv)
{
/* DICHIAZIONI ED INIZIALIZZAZIONE */
    /* dichiarazione delle strutture necessarie */
    int sock; /* descrittore della socket */
    struct sockaddr_in server;
    struct hostent *hp;
    char input[256];
    /* verifica della correttezza dell'input */
    if(argc!=3) {
        printf("uso: %s <host> <numero-della-porta>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    /* chiedo al resolver di tradurre il nome in indirizzo ip */
    /* se argv[1] e' gia' un indirizzo, allora hp viene
       semplicemente aggiornato con l'indirizzo fornito */
    hp = gethostbyname(argv[1]);
    if( hp == NULL ) {
        printf("%s: l'host %s e' sconosciuto.\n",
               argv[0], argv[1]);
        exit(EXIT_FAILURE);
    }
    /*chiedi al S.O. una socket */
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if( sock < 0 ) {
        printf("client: errore %s nella creazione della socket\n",
               strerror(errno));
        exit(EXIT_FAILURE);
    }
/*INSTAURAZIONE DELLA CONNESSIONE */
/* inizializzazione della struttura server */
server.sin_family = AF_INET;
memcpy(&server.sin_addr, hp->h_addr, hp->h_length);
server.sin_port = htons(atoi(argv[2]));

```

Tutte le librerie necessarie

**Argc:** è un valore intero (in questo caso 3).

**Char\*\* argv:** p un puntatore a puntatori e punta ad un array in cui ogni elemento punta alla stringa contenente la richiesta

**Int socket:** Maniglia attraverso la quale accedo al file; scrittura e lettura di file.

**Struct sokaddr\_in server:** record per poter interagire con le

Scrivo all'utente come deve scrivere la richiesta.

Argv[0] rappresenta la richiesta  
(in questo caso di addizionare)

Si chiede al sistema operativo una socket.

Funzione: int socket(int domain, int type, int protocol)

Dove:

- Domain: è il dominio in cui ci troviamo (AD\_INET è il dominio di internet).
- Type: modalità di comunicazione (a pacchetto o altro).
- Protocol: protocollo usato (o per quello di default).

**Memcpy:** copia in server.sin\_addr l'indirizzo del server. H atre parametri: puntatore all'inizio del blocco di memoria destinazione della copia, puntatore all'inizio del blocco di memoria da copiare e dimensione del blocco da copiare.

**Hton:** "host-to-network-short" converte l'intero in un formato indipendente dalla piattaforma. Traduce l'intero in un linguaggio condiviso con altri host.

Atoi: converte la stringa in un intero.

```

/* connessione */
if(connect(sock, (struct sockaddr *)&server, sizeof(server))<0) {
    printf("client: errore %s durante la connect\n",
    strerror(errno));
    exit(EXIT_FAILURE);
}
printf("client: connesso a %s, porta %d\n", argv[1],
    ntohs(server.sin_port));
/* SCAMBIO DATI */
printf("client: num. o \"quit\"? ");
scanf("%s",input);
while( strcmp(input,"quit") != 0 ) {
    char result[256];
    if( write(sock, input, strlen(input) ) <0 ) {
        printf("errore %s durante la write\n", strerror(errno));
        exit(EXIT_FAILURE);
    }
    if( read(sock, result, sizeof(result)) < 0 ) {
        printf("errore %s durante la read\n", strerror(errno));
        exit(EXIT_FAILURE);
    }
    printf("client: ricevo dal server %s\n", result);
    printf("client: num. o \"quit\"? ");
    scanf("%s",input);
}
/*CHIUSURA */
close(sock);
printf("client: ho chiuso la socket\n");
return 0;
}

```

Connect: struttura => int  
connect(int *sock\_fd*, struct sockaddr \**serv\_addr*, int *addrlen*);

Si usa solamente con  
SOCK\_STREAM (protocolli  
connessi). Il server deve essere già  
in attesa di una connessione e la  
struttura struct sockaddr è un input  
(contiene l'indirizzo IP e la porta  
TCP del server). La funzione ritorna  
-1 in caso di errore. Timeout, errore  
nei parametri, connessione già  
aperta

**Sock:** identifica la connessione.

La struct effettua il casting della  
struttura server di tipo sockaddr\_in  
nel tipo sockaddr.

Questa connect può andare in  
errore o può funzionare: in questo  
caso con la printf: con htons

Con strcmp avviene una comparazione tra stringhe e  
restituisce 0 se sono uguali.

Char *result* contiene il risultato dell'addizione.

Con write invio al server dei caratteri contenuti nella  
variabile input. La funzione è: int write(int *sock\_fd*, const  
void \**buf*, int *len*). Invia il contenuto della variabile *buf*  
(buffer) alla socket specificata. Si usa esclusivamente con  
SOCK\_STREAM (cioè con protocolli connessi). La funzione  
ritorna il numero di byte inviati oppure -1 in caso di errore ed  
è la stessa funzione che consente la scrittura su un file.

Con read ricevo dal server il risultato e lo memorizzo in  
result. La funzione è: int read(int *sock\_fd*, void \**buf*, int *len*).

Solo per socket connesse (SOCK\_STREAM). Legge un  
messaggio di lunghezza massima len dalla socket specificata;  
la chiamata è bloccante (se non c'è alcun messaggio da  
leggere, il programma rimane sospeso). La funzione ritorna il  
numero di byte letti oppure -1 in caso di errore ed è la stessa  
funzione che consente la lettura da un file

Con close(sock) avviene la chiusura dal lato client. Viene  
eseguita sia dal client che dal server per dichiarare al sistema  
operativo che non si ha più interesse a conservare aperta la  
connessione (necessariamente è di tipo SOCK\_STREAM). Più  
processi dello stesso host possono condividere la stessa  
socket: solo quando tutti avranno eseguito una close il  
sistema operativo provvederà a chiudere la connessione  
(trasmissione di un fin). la chiusura è simmetrica: la  
connessione sarà effettivamente chiusa quando sarà stata  
chiusa sia sul server che sul client.

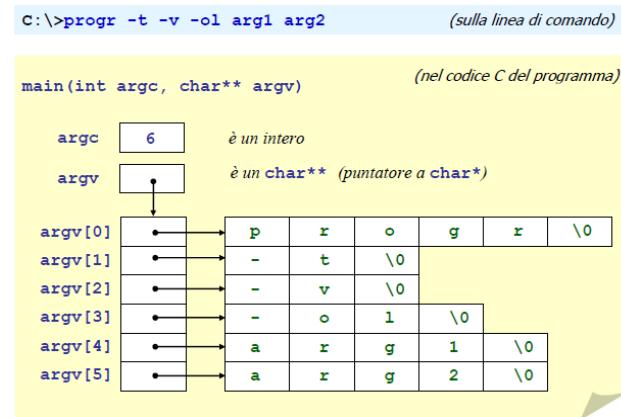


Figura 66 - Uso generale di argc e argv

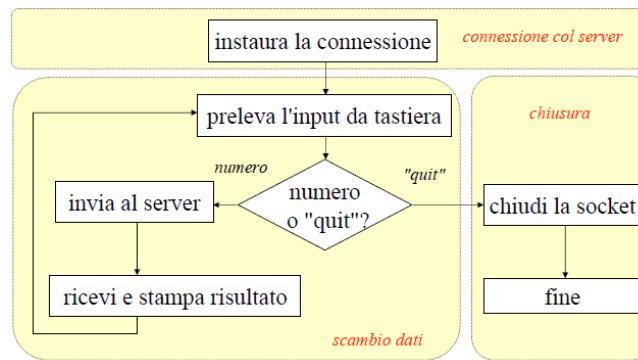


Figura 67 - Instaurazione della connessione

## Server

Il processo server viene lanciato da linea di comando specificando la porta TCP di ascolto come parametro.



## Server in C

```

/*DICHIARAZIONE DELLE STRUTTURE NECESSARIE E INTESTAZIONE DEL
PROGRAMMA*/
#include <stdio.h> /* per usare input-output */
#include <stdlib.h> /* per usare exit() */
#include <sys/socket.h> /* per usare socket */
#include <netinet/in.h> /* per sockaddr_in */
#include <errno.h> /* gestione degli errori */
main(int argc, char** argv)
{
int sock; /* socket di ascolto */
int sockmsg; /* socket di dialogo */
struct sockaddr_in server;
/*VERIFICA DELLA CORRETTEZZA DELL'INPUT */
if ( argc != 2 ) {
    printf("uso: %s <numero-della-porta>\n",
    argv[0]);
    exit(EXIT_FAILURE);
}
sock = socket(AF_INET, SOCK_STREAM, 0);
if( sock < 0 ) {
    printf("server %d: errore %s nel creare la socket\n",
    getpid(), strerror(errno));
    exit(EXIT_FAILURE);
}
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons(atoi(argv[1]));
if(bind(sock, (struct sockaddr *)&server, sizeof(server))) {
    printf("server %d: bind fallita\n", getpid());
    exit(EXIT_FAILURE);
}
printf("server %d: rispondo sulla porta %d\n",
getpid(), ntohs(server.sin_port));
if( listen(sock, 4) < 0 ) {
    printf("server %d: errore %s nella listen\n",
    getpid(), strerror(errno));
    exit(EXIT_FAILURE);
}
/*ATTESA DELLA CONNESSIONE*/
  
```

ci aspettiamo che il server sia lanciato dall'amministratore del servizio in questo modo:  
C:\>addizionatore 2000

Con **Sock** conservo il descrittore, con **AF\_INET** indico l'utilizzo di internet e con **SOCK\_STREAM** uso TCP. È identica alla richiesta fatta dal client. L'unica differenza è che la socket creata (**sock**) non verrà usata direttamente, ma come *modello* per altre socket (che chiameremo "di dialogo").

**INADDR\_ANY**: uso l'indirizzo della macchina.

**Server.sin\_port**: preparo la struttura specificando il numero di porta.

**Bind**: chiedo al server di assegnare (bind = legare) alla mia socket il numero di porta specificato. La funzione generale è: int bind(int *sock\_fd*, struct sockaddr \* *my\_addr*, int *addrlen*);

Assegna una porta alla socket appena creata; è eseguita sempre dal server (la può eseguire anche il client); se si richiede la porta zero, il SO assegna alla socket la prima porta dinamica disponibile (e aggiorna la struttura *struct sockaddr*). La struttura *struct sockaddr* funge sia da input che da output (può essere consultata per sapere il numero della porta assegnata). Ritorna -1 in caso di fallimento (porta richiesta non disponibile, parametri non validi, socket già assegnata ad un indirizzo).

**Listen**: dimensiono (almeno 4 slot) la coda di backlog e marco la socket come "listening". La funzione di base è: int listen(int *sock\_fd*, int *backlog*);

La funzione eseguita solo dal server; etichetta la socket come una socket "listening": il sistema operativo accetta connessioni sulla porta individuata implicitamente dalla socket e le mette in una coda, lunga almeno quanto specificato nel parametro *backlog*. Il server potrà estrarre una connessione dalla coda di backlog tramite la funzione *accept* ritorna -1 in caso di insuccesso (memoria insufficiente, parametri non validi, bind non eseguita, la socket è già connessa. La listen si applica solamente alle connessioni di tipo **SOCK\_STREAM** (che sono connesse))

```

while(1) {
    int totale=0, len;
    char input[256];
    if( (sockmsg = accept(sock, o, o)) < 0 ) {
        printf("server %d: errore %s nella accept\n", getpid(), strerror(errno));
        exit(EXIT_FAILURE);
    }
    printf("server %d: accettata una nuova connessione\n",
    getpid());
/*SCAMBIO DATI*/
while(1) {
    char message[256];
    int len = read( sockmsg, input, sizeof(input) );
    if ( len == 0 ) break;
    input[len]='\0';
    printf("server %d: arrivato il numero: %s\n",
    getpid(), input);
    totale = totale + atoi(input); /* esegui la somma */
    printf("server %d: invio il totale %d\n",
    getpid(), totale);
    sprintf(message, "%d", totale); /* prepara messaggio */
    write(sockmsg, message, sizeof(message));
}
/*CHIUSURA CONNESSIONE*/
close(sockmsg); //Chiusura della socket di dialogo
printf("server %d: ho chiuso la socket di dialogo\n",
getpid());
} /* fine del ciclo infinito del server e della funzione main*/
close(sock); /* inutile: non arrivero' mai qui! */
printf("server %d: ho chiuso la socket di ascolto\n",
getpid());

```

Con il `while` entro in un ciclo infinito (il server è sempre in funzione). In `sockmsg` memorizzo il descrittore della nuova socket di dialogo creata sul modello di `sock`.

La funzione `accept` è descritta in questo modo: `int accept( int sock_fd, struct sockaddr * addr, int addrlen);`  
 La funzione eseguita solo dal server quando è pronto a ricevere una connessione. Il programma viene bloccato su questa istruzione fino a quando un client non abbia fatto una richiesta di connessione sulla stessa porta e con lo stesso tipo di socket (stesso input alla funzione `socket`). La struttura `struct sockaddr` è un output: verrà aggiornata con l'indirizzo ip e la porta del client connesso. La funzione ritorna una nuova socket ("di dialogo") creata sul modello di quella passata come primo parametro. La `accept` si applica solamente alle connessioni di tipo `SOCK_STREAM`.

#### LEGGI MESSAGGIO DAL CLIENT

Con il primo `while` si entra in un ciclo infinito.

Con `len` la lunghezza del messaggio letto, mentre con `read` si attende di ricevere il messaggio dal client (chiamata "bloccante"), `sizeof` indica la dimensione massima del messaggio da leggere.

#### RILEVAZIONE DEL MESSAGGIO

*If (len==0)* indica la rilevazione indiretta della chiusura della connessione da parte del client. Con `input[len]='\0'` indichiamo come termina la stringa.

#### ESEGUI LA SOMMA

Aggiorno il `totale` con la stringa trasformata in un numero intero on `atoi`.

#### INVIA IL RISULTATO AL CLIENT

Preparo la stringa da inviare al client con `sprintf`(`sprint print formatted`); in `%d` inserisco il totale intero.

Con `write` invio il messaggio

Questo è uno schema dello scambio di dati:



Figura 68 - Socket lato server: Scambio dati

Riassunto delle primitive TCP:

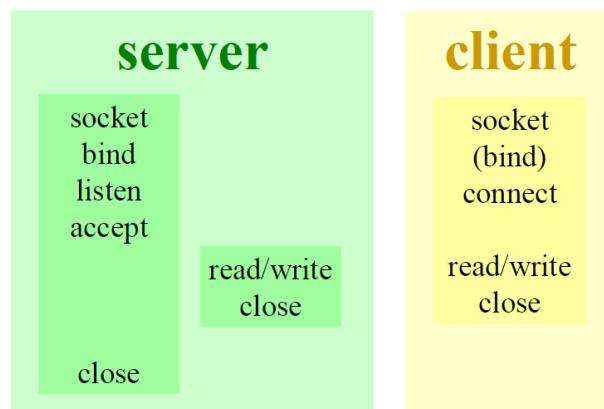


Figura 69 - Riassunto delle primitive TCP

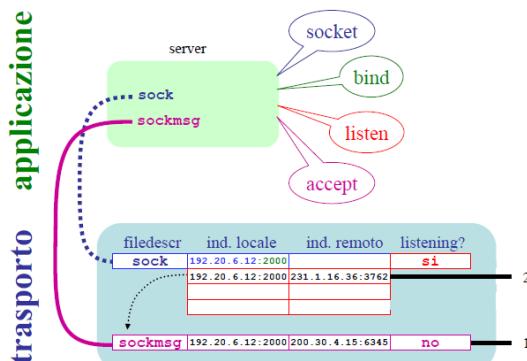


Figura 70 - Socket a livello di trasporto

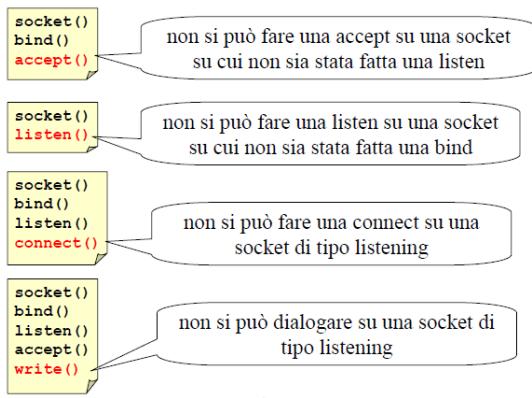


Figura 71 - Successioni errate delle primitive

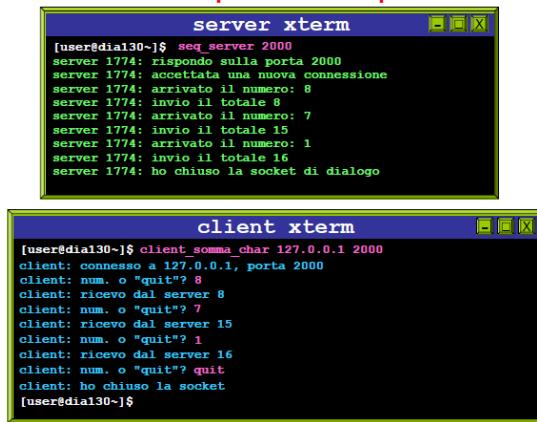


Figura 72 - Esempio di output

### Server TCP che fa uso di fork

Il server è in attesa di una connessione TCP su una porta specificata sulla linea di comando: i client vengono serviti in parallelo facendo uso della funzione “*fork*” dove ogni client invia una sequenza di numeri interi, trasferiti come stringhe di caratteri. Per ogni numero ricevuto il server restituisce al client una stringa di caratteri rappresentante la somma di tutti i numeri inviati al server dallo stesso client.

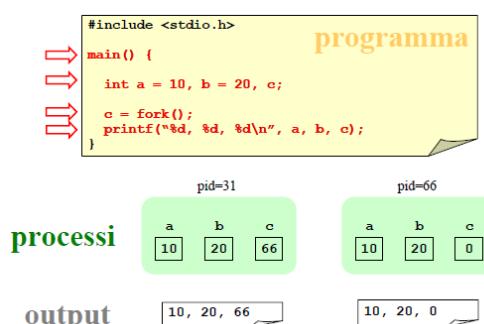


Figura 73 - Semantica della fork

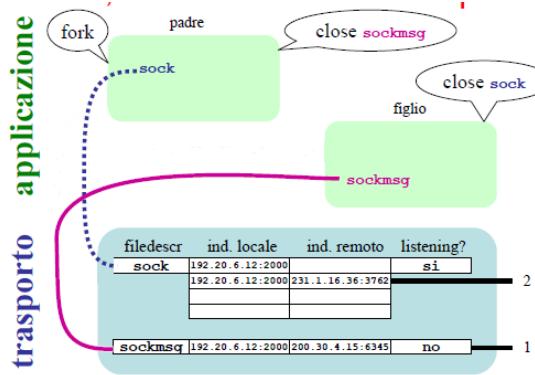


Figura 74 - Socket, fork a livello di trasporto

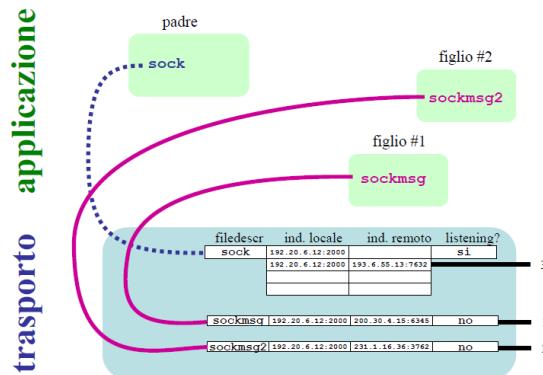


Figura 75 - Socket, fork a livello di trasporto

Dopo una fork, il processo padre e il processo figlio hanno i medesimi privilegi nei confronti della socket di ascolto e della socket di dialogo: le operazioni eseguite dai due processi sulla socket di ascolto (*accept*) e sulla socket di dialogo (*read/write*) potrebbero interferire. Il livello di trasporto non invia il pacchetto fin, se tutti i processi che condividono la stessa socket non hanno eseguito una *close*.

Se il processo padre non chiude la socket di dialogo, allora la chiusura da parte del processo figlio della socket di dialogo non provocherà il rilascio della connessione:

- il client rimane appeso;
- il processo padre può bloccarsi per aver superato il numero massimo di “file” che un processo può tenere aperti contemporaneamente.

Se il processo figlio non chiude la socket di ascolto, allora la chiusura da parte del processo padre della socket di ascolto non avrà l’effetto di rendere la porta di nuovo disponibile: il server non potrà essere rilanciato con lo stesso numero di porta.

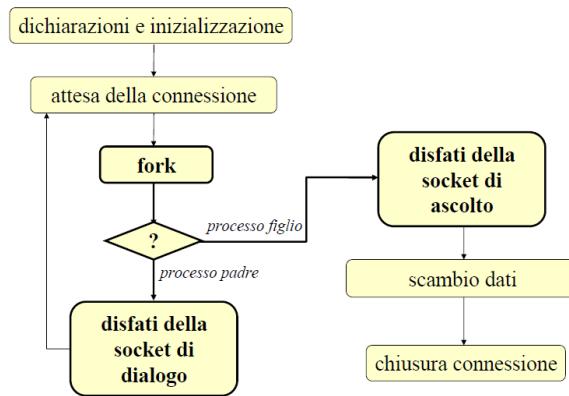


Figura 76 - Schema di un server TCP con fork

```

if ( fork() == 0 ) {
    printf("figlio %d: servo io il client\n", getpid() );
    close(sock); /* il figlio chiude la socket di ascolto */
    ...
    exit(0);
} else {
    ...
    close(sockmsg); /* il padre chiude la socket di dialogo */
}
  
```

qui ci va scambio dati

torna a attesa della connessione

Figura 77 - Nuova parte di codice



Figura 78 - Esempio di output

## Client e server UDP

### Client

Il colloquio TCP è un colloquio unidirezionale contemporaneo. Si prende un pacchetto e si tira al di là della rete. Dobbiamo fare la solita dichiarazione al sistema operativo sul fatto che intendiamo usare una socket e spediamo il pacchetto al server.



Figura 79 - Schema di un semplice client UDP

*Client in C*

```
/* DICHIARAZIONE DELLE STRUTTURE NECESSARIE E INTESTAZIONE DEL
PROGRAMMA */

#include <stdio.h> /* per usare input-output */
#include <stdlib.h> /* per usare exit() */
#include <string.h> /* per usare memset() */
#include <sys/types.h>
#include <sys/socket.h> /* per usare socket */
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h> /* gestione degli errori */
#define MY_PORT_ID 6089 /* la mia porta */
#define SERVER_PORT_ID 6090 /* la porta del server */
#define SERV_HOST_ADDR "127.0.0.1"
main() {
    int sock, retcode;
    struct sockaddr_in my_addr;
    struct sockaddr_in server_addr;
    char msg[12]; /* conterrà il messaggio da inviare */
```

È l'invocazione della funzione socket. La primitiva e esattamente la stessa, ma invece di specificare che lavoriamo su un flusso si specifica d lavorare su un datagramma. La presenza dello o implica il fatto di lavorare con il protocollo di default, quindi UDP.

**Sock:** conservo il descrittore.  
**AF\_INET:** usata in internet.  
**SOCK\_DGRAM:** uso UDP.

```
/* CHIEDI AL SISTEMA OPERATIVO UNA SOCKET */
```

```
sock = socket(AF_INET, SOCK_DGRAM, 0);
if( sock < 0 ) {
    printf("client: errore %s nel creare la socket\n", strerror(errno));
    exit(1);
}
```

**Bzero:** inizializzo la struttura riempendola di zeri.  
**Hton:** specifico la porta desiderata.  
**Bind:** lego la mia socket alla porta

```
/* INIZIALIZZA LA SOCKET */
```

```
printf("client: eseguo la bind\n");
memset((char *) &my_addr, '\0', sizeof(my_addr));
my_addr.sin_family = AF_INET;
my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
my_addr.sin_port = htons(MY_PORT_ID);
if ( (bind(sock, (struct sockaddr *) &my_addr, sizeof(my_addr)) < 0) ) {
    printf("client: errore %s nella bind\n", strerror(errno));
    exit(1);
}
```

Si invia il messaggio al server specificando indirizzo dell'host e della porta.

Dopo aver spedito questo messaggio si chiude la socket.

La *sendto* invia un singolo pacchetto contenente il messaggio nel parametro msg all'indirizzo specificato. Si usa solamente con SOCK\_DGRAM (protocolli non connessi). La funzione ritorna -1 in caso di errore e setta la variabile errno al valore di EMSGSIZE qualora l'errore sia dovuto alla taglia eccessiva del messaggio (che non entra in un solo pacchetto UDP). Il parametro flags sarà sempre posto a zero per i nostri scopi.

```
/* INVIA IL MESSAGGIO AL SERVER E CHIUDI */
```

```
bzero((char *) &server_addr, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
server_addr.sin_port = htons(SERVER_PORT_ID);
printf("client: invio il messaggio al server\n");
sprintf(msg, "hello world");
retcode = sendto(sock,msg,12,0,(struct sockaddr *)
&server_addr,sizeof(server_addr));
```

---

```

if (retcode <= -1) {
    printf("client: errore %s nella sendto\n",
           strerror(errno));
    exit(1);
}
close(sock); /* close the socket */
}

```

---

### Server

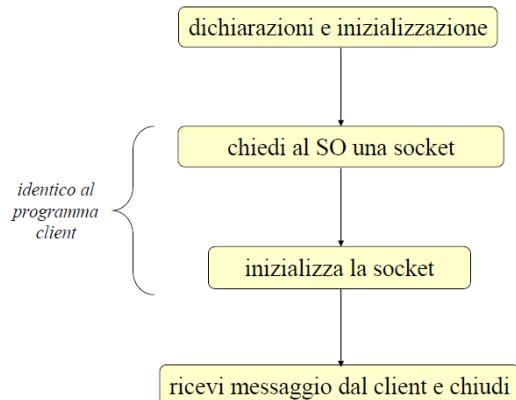


Figura 8o - Schema di un semplice server UDP

### Server in C

---

```

/* DICHIARAZIONE DELLE STRUTTURE NECESSARIE E INTESTAZIONE DEL
   PROGRAMMA */

#include <stdio.h> /* per usare input-output */
#include <sys/socket.h> /* per usare socket */
#include <stdlib.h> /* per usare exit() */
#include <errno.h> /* gestione degli errori */
#include <string.h> /* per bzero() */
#include <arpa/inet.h>
#define MY_PORT_ID 6090

main() {
    int sock, nread, addrlen;
    struct sockaddr_in my_addr;
    struct sockaddr_in client_addr;
    char msg[50];

    /* CHIEDI AL SISTEMA OPERATIVO UNA SOCKET */
    sock = socket(AF_INET, SOCK_DGRAM, 0);
    if( sock < 0 ) {
        printf("client: errore %s nel creare la socket\n",
               strerror(errno));
        exit(1);
    }
}
```

```

}

/* INIZIALIZZA LA SOCKET */
printf("client: eseguo la bind\n");
bzero((char *) &my_addr, sizeof(my_addr));
my_addr.sin_family = AF_INET;
my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
my_addr.sin_port = htons(MY_PORT_ID);
if ( (bind(sock, (struct sockaddr *) &my_addr, sizeof(my_addr)) < 0) ) {
    printf("client: errore %s nella bind\n",
    strerror(errno));
    exit(1);
}

/* RICEVI IL MESSAGGIO DAL CLIENT E CHIUDI */
printf("server: attendo il messaggio dal client\n");
nread = recvfrom(sock, msg, 11, 0,
(struct sockaddr *) &client_addr, &addrlen);
printf("Server: sono arrivati %d bytes\n", nread);
if (nread > 0) printf("server: messaggio=%.11s\n", msg);
close(sock);
}

```

**Nread:** attendo un messaggio sulla porta specificata implicitamente dalla socket con `recvfrom`: questo contiene i messaggi grandi fino a 11 caratteri e nell'output avremo l'indirizzo del mittente.  
Inoltre `recvfrom` legge un messaggio della lunghezza massima specificata . La struttura `struct sockaddr` e un output: verrà aggiornata dal livello di trasporto e conterrà l'indirizzo del mittente; se non arriva alcun messaggio, il programma rimane sospeso (la chiamata è "bloccante"). Si usa solamente con `SOCK_DGRAM` (protocolli non connessi). La funzione ritorna il numero di byte letti oppure -1 in caso di errore. Il parametro `flags` è generalmente impostato a zero.

server	client
socket	socket
bind	(bind)
recvfrom	sendto
close	close

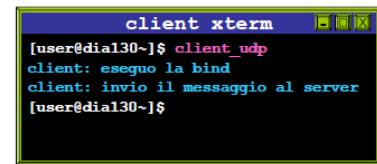
Figura 82 - Riassunto delle primitive UDP



```

server xterm
[user@dia130~]$ server_udp
server: eseguo la bind
server: attendo il messaggio dal client
server: sono arrivati 12 bytes
server: messaggio=hello world
[user@dia130~]$

```



```

client xterm
[user@dia130~]$ client_udp
client: eseguo la bind
client: invio il messaggio al server
[user@dia130~]$

```

Figura 81 - Esempio di output

## Tecnologie e metodologie per lo strato di trasporto: affidabilità ed efficienza di TCP

Parlare di TCP in realtà è parlare di tantissime cose, perché TCP è stato implementato e reimplementato su diverse piattaforme, e le implementazioni continuano a modificarsi, poiché è un protocollo end to end, quindi possiamo fare tutte le modifiche che vogliamo ( purché siano consistenti con lo standard).

### Instaurazioni di connessioni

Un problema banale: dobbiamo instaurare una connessione con TCP e fare il 3 way handshake. Per farlo dobbiamo scegliere un numero iniziale di sequenza, che va scelto con una certa cura. Per quale motivo? Malfunzionamenti della rete possono far nascere dei duplicati di pacchetti, che possono essere pericolosi per l'applicativo. Supponiamo di lavorare su una connessione per un prelievo bancario. Cosa succede?

- alcuni pacchetti usati nella connessione vengono ritardati;
- vengono spediti dei duplicati;
- viene effettuato il prelievo;
- i pacchetti in ritardo arrivano dopo un po' nell'ambito di un'altra connessione;
- il prelievo viene duplicato.

Il numero iniziale di sequenza utilizzato da un estremo viene chiamato *initial sequence number* (ISN), e per far sì che non ci siano sovrapposizioni nello spazio di numerazione ogni connessione usa un ISN diverso. Se esce fuori un pacchetto numerato 4 in una connessione arrivata al numero di sequenza 100, il primo pacchetto non viene riscontrato. Per capire qual è il contesto della connessione si aggiunge alla quartina  $\langle ip_1, porta_1, ip_2, porta_2 \rangle$  la coppia di ISN utilizzata dai due interlocutori. Possiamo pensare di utilizzare il valore dell'orologio di sistema. Quando una connessione inizia posso prendere come ISN la stringa dei  $k$  bit meno significativi dell'orologio. In questo modo ogni connessione numera le proprie PDU partendo da un numero diverso; in questo modo ho una relazione lineare tra tempo e numero di partenza di una connessione. Questo  $k$  deve essere abbastanza grande in modo che le vecchie PDU devono già essere state eliminate quando la stringa di  $k$  bit assume nuovamente gli stessi valori. Questa però è una soluzione inefficace: c'è il problema del crash. Supponiamo che un calcolatore si fermi, per qualche motivo. Quando riparte, il calcolatore non sa quali sono i numeri assegnati ai pacchetti spediti prima del crash, che vengono ricavati dall'orologio. C'è il rischio a questo punto di riusare ISN già usati in precedenza.

In questo piano metto un punto in ragione del numero di pacchetto inviato in un certo istante. Il collegamento tra i punti sta ad indicare che i pacchetti fanno parte di una stessa connessione. Se tutti i pacchetti fossero piazzati lungo la bisettrice (*linea della numerazione iniziale*) significherebbe che c'è un perfetto sincronismo tra TCP e l'orologio, ma a questo punto tutti i numeri di pacchetto uscirebbero fuori in punti che non sono proprio allineati e coperti con la bisettrice. Se uscissero più lentamente starebbero sotto la bisettrice.

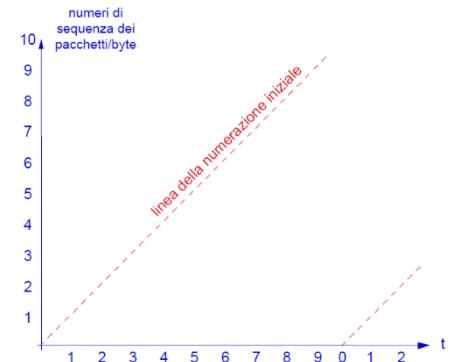


Figura 83 - Uso del clock

Dopo il crash, la connessione ha stesso server, stesso client, stesse porte. La regola dice che il numero di sequenza deve essere scelto sulla bisettrice. Se i due pacchetti "numero 5" sopravvivono, e sono contemporaneamente presenti nella rete, sono pericolosissimi. Di quanto si devono distanziare i tempi di questi pacchetti? Devono essere distanti almeno quanto si presume sia il tempo di vita del pacchetto nella rete. Il problema del crash rende problematico questo semplice utilizzo del numero iniziale di sequenza sulla bisettrice. Il problema potrebbe essere mitigato dal fatto che il tempo di boot di una macchina è enorme, ma potremmo avere macchine dedicate a fare lavori specifici, con tempi di boot limitatissimi, e a questo punto il problema si pone. Una soluzione inefficiente a questo problema è la seguente: quando riparte, il calcolatore aspetta sempre prima di mandare il primo pacchetto, almeno un tempo pari al massimo tempo  $T$  di sopravvivenza in rete di un pacchetto.  $T$  può essere troppo grande per molti motivi (ci sono macchine senza disco, che hanno bisogno immediatamente di una connessione TCP).

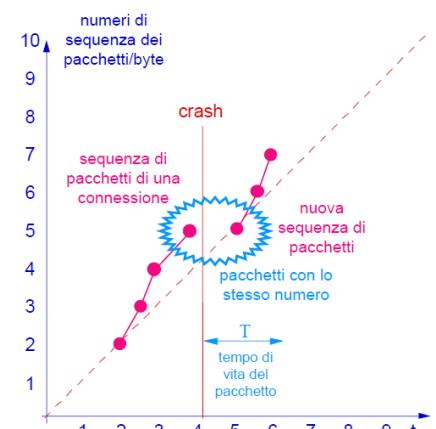


Figura 84 - Uso del clock - Problema del crash

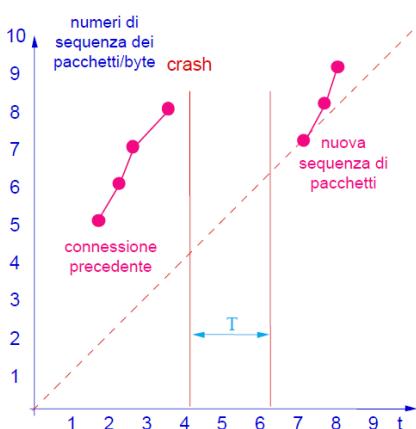


Figura 85 - Uso del boot + attesa del timeout

Una soluzione migliore è il metodo di Tomlinson, che implica una restrizione sui numeri di sequenza un po' meno drastica. Un numero di sequenza non deve essere usato per l'intervallo di tempo  $T$  che precede l'istante in cui è potenzialmente utilizzabile come numero iniziale di sequenza. Quindi il numero di pacchetto  $\rho$  è inutilizzabile per il tempo compreso tra  $(\rho - T)$  e  $\rho$ . Come interpretare questa restrizione? Se un certo numero di sequenza non è utilizzabile per un certo periodo, si deve inviare un pacchetto e si verifica se in quel momento il numero di sequenza che sto usando è utilizzabile. Non si deve cambiare, se non è utilizzabile si aspetta che lo diventi.

Questo può introdurre del ritardo nella spedizione dei pacchetti "sfortunati".

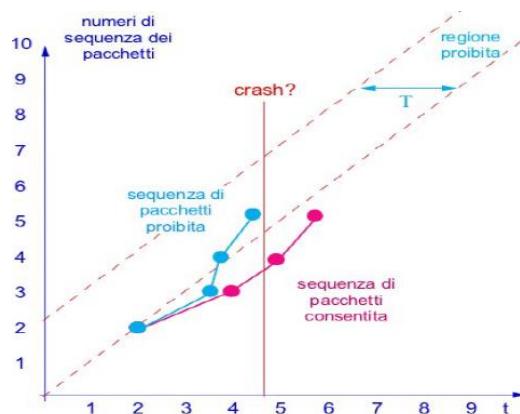


Figura 86 - Uso del clock - Attesa del timeout con regione proibita

Possiamo costruire attorno alla bisettrice quella che viene chiamata "regione proibita", nella quale non ci possono essere pacchetti. Questa regione proibita è larga  $T$ , orizzontalmente. Quando si fa partire una connessione si sceglie il numero di sequenza sulla bisettrice, e si inizia a numerare, però non si può spedire un pacchetto se è nella regione proibita, quindi bisogna aspettare che questo pacchetto esca dalla regione proibita. Se siamo fuori dalla regione proibita non corriamo il rischio dei pacchetti contemporaneamente presenti. E se saliamo sopra la regione proibita?

Le cose vanno sicuramente bene, perché dopo il crash è passato sicuramente il tempo  $T$ . Dopo un avvio lento, i pacchetti possono essere spediti rapidamente, quindi si rischia di entrare nella regione proibita dal basso; serve un orologio molto veloce. Dobbiamo tenere presente che l'orologio ogni tanto riparte da 0, quindi in realtà avremo una sequenza di regioni proibite.

Il metodo di Tomlinson però non viene usato. L'RFC 793 dice di utilizzare un contatore a 32 bit, con un incremento da parte del software che implementa TCP ogni 4 microsecondi. Se la memoria è persa (per esempio, in seguito ad un crash) si osserva un "quiet time" di un MSL (*maximum segment lifetime* (per questi fini si considera MSL = 2 minuti)). Generalmente però si usa un contatore a 32 bit, incrementato ogni 8 microsecondi e ad ogni nuova connessione.

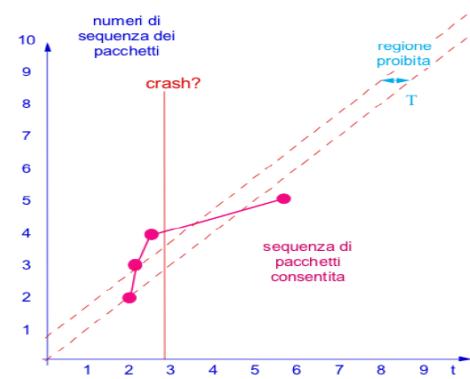


Figura 87 - Uso del clock - Attesa del timeout con regione proibita

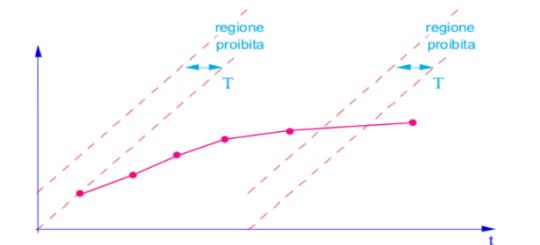


Figura 88 - Confronto tra le regioni

## Efficienza

Quando si considera l'efficienza si deve considerare il rapporto tra i dati che le applicazioni devono trasferire e i dati complessivamente inviati (che contengono anche informazioni di intestazione, per esempio). TCP ha a disposizione le informazioni per intervenire sull'efficienza della rete, e vede il trasferimento come un flusso di pacchetti; gli strati superiori non vedono i pacchetti, e gli strati inferiori non sono in grado di cogliere la relazione tra pacchetti diversi. Le strategie di TCP per aumentare l'efficienza sono ridurre il numero di pacchetti, inviati e persi. Per ridurre il numero di pacchetti inviati possiamo cercare di cumulare più segmenti TCP in un segmento più grande, ma anche ridurre il numero delle notifiche (*acknowledgment*, variazioni delle finestre). Per ridurre il numero di pacchetti persi dobbiamo ridurre le congestioni. Il TCP mittente può aspettare che i dati si accumulino prima di inviarli, il TCP destinatario può aspettare che i dati si accumulino prima di riscontrarli. Questo ci deve però permettere di agire con eccezioni, ci sono dati urgenti che vanno riscontrati immediatamente. La politica di trasmissione adottata può influenzare drammaticamente le prestazioni di TCP.

CODA

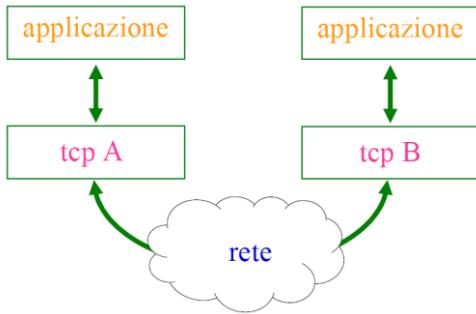


Figura 89 - Gradi di libertà di TCP

Il campo *window* si trova nella header di un segmento TCP e specifica il numero di byte, a partire dal byte indicato nell'ACK, che il ricevitore è disponibile ad accettare. Il campo *window* nei pacchetti TCP da sempre è 16 bit, quindi la finestra massima è di 65635 byte. Ad esempio, consideriamo il trasferimento di 8192 byte tra sx.dia.uniroma3.it e dx.dia.uniroma3.it, e supponiamo che la fase di 3-wayhandshake sia già terminata (omettiamo i pacchetti 1,2 e 3 della conversazione).

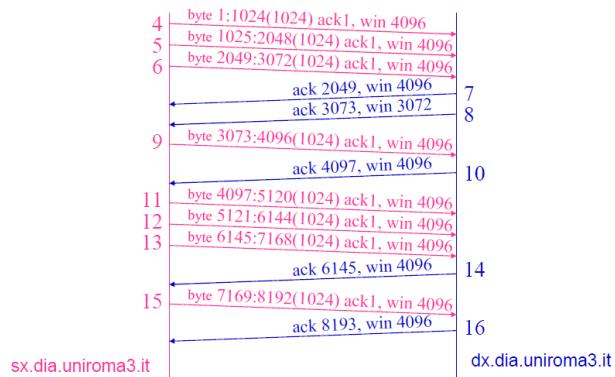


Figura 90 - Esempio di finestra di controllo di flusso

Il 3 way handshake è finito, osserviamo i pacchetti 4-5-6. La freccia relativa al pacchetto 4 dice "porto i byte da 1 a 1025, faccio un ACK del pacchetto 1 e specifico una finestra di 4096 byte. Vediamo i riscontri che arrivano in direzione opposta; il primo pacchetto (7) ha un riscontro cumulativo sui pacchetti 4 e 5, il secondo indica di aver ricevuto tutti i byte del pacchetto 6. Quel 3072 ci dice che il buffer dell'applicazione non si è pulito completamente, quindi non è in grado di accettare i 4096 byte. Col pacchetto 11 sx riparte appieno; dx col pacchetto 14 riscontra i pacchetti 11 e 12. [pacchetto 15: invece di 8183 è 8193, c'è un errore].

Vediamo lo stesso esempio in maniera diversa:

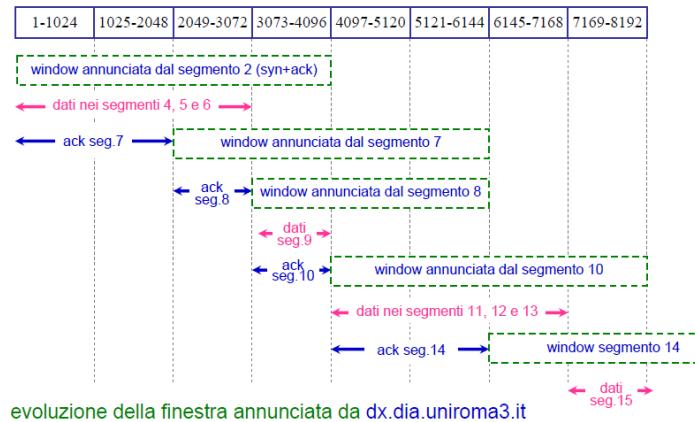


Figura 91 - Esempio della finestra di controllo di flusso

Vediamo come TCP fa il suo lavoro di ottimizzazione per flussi interattivi. In un flusso interattivo ci si scambiano frequentemente piccole quantità di dati. Se mandiamo pochi byte per ciascun pacchetto il rischio è quello di un'efficienza bassissima. Un tipico esempio di servizio interattivo è telnet: ogni volta che un utente preme un tasto un carattere viene inviato al server telnet (talvolta un eco del carattere viene inviato in verso opposto per mostrarlo all'utente). Mostriamo come l'invio di un singolo carattere comporti la spedizione di 3 segmenti TCP, corrispondenti a 121 byte IP.

- L'utente preme un tasto.
- TCP prepara un segmento, lo incapsula in un datagram e lo invia: pacchetto 1: 41 byte (20 IP header + 20 TCP header + 1 telnet).
- Lo strato TCP ricevente riscontra il segmento: pacchetto 2: 40 byte (20 IP header + 20 TCP header).
- Quando il server telnet legge il carattere allora TCP notifica una variazione della finestra di un byte: pacchetto 3: 40 byte (20 IP header + 20 TCP header).

#### TCP per servizi interattivi

Il server telnet processa il carattere e può inviare un eco: altri 121 byte scambiati per consentire il transito di un byte in senso opposto. La prima cosa che possiamo fare è questa: possiamo spedire riscontri ritardati. Posso ritardare le variazioni di finestra di controllo di flusso e di ACK di un po' di tempo. In questo modo riesco a evitare di mandare un pacchetto per la variazione e uno per l'ACK, molti messaggi possono essere cumulati nello stesso pacchetto. Nell'esempio del telnet se il carattere viene processato abbastanza in fretta il secondo ed il terzo pacchetto potrebbero essere cumulati, ma in ogni caso il mittente spedisce 41 byte per ogni carattere.

### *Algoritmo di Nagle*

Per servizi interattivi si può fare qualcosa di meglio. L'algoritmo di Nagle ha lo scopo di ridurre il numero di pacchetti che viaggiano dal produttore al consumatore del flusso nel caso in cui il produttore produca pochi byte alla volta. L'algoritmo funziona in questo modo:

- Si trasmette il primo troncone di dati (anche un solo byte) e si lasciano cumulare gli altri byte nel buffer in uscita.
- Il segment successivo è inviato solo quando una di queste tre condizioni è verificata:
  1. il precedente è stato riscontrato;
  2. la finestra è satura;
  3. è stata raggiunta la dimensione di un segment.

Ad esempio, se l'utente di un servizio telnet digita velocemente e la rete è molto lenta ogni segment può trasportare tutti i byte scritti durante il round-trip delay. Questo algoritmo ha minore effetto sulle brevi distanze: gli ACK tornano velocemente e le spedizioni sono più frequenti. In rete locale un carattere viene spedito, riscontrato e mostrato in eco in circa 15ms, quindi per vedere in modo significativo gli effetti dell'algoritmo bisognerebbe digitare più di 60 caratteri al secondo (un carattere ogni 16 millisecondi). In rete geografica i ritardi sono consistenti, quindi gli effetti dell'algoritmo sono più visibili.

### *Algoritmo di Clark*

Il duale è l'algoritmo di Clark, che risolve un problema opposto. Con l'algoritmo di Nagle avevamo un mittente che produce pochi byte alla volta. Qui succede l'esatto opposto: c'è un mittente che genera una montagna di byte, e un ricevente che li elabora poco alla volta (*sindrome silly window*). Il buffer di ingresso del ricevente si satura immediatamente, e la finestra di controllo di flusso diventa subito 0. Quando l'applicazione ricevente legge un byte il TCP ricevente notifica la variazione della finestra di controllo di flusso. Il TCP trasmittente va a saturare di nuovo la finestra di controllo di flusso. Questo significa che il tempo di comunicazione lo detta l'applicazione ricevente. La soluzione di Clark è questa: le variazioni della finestra vanno inviate solo quando la finestra raggiunge metà dell'ampiezza massima o la massima dimensione del segmento negoziata all'inizio (MSS). I problemi affrontati da Nagle e Clark sono complementari:

- l'applicazione invia un byte alla volta (TCP ha il buffer di uscita vuoto);
- l'applicazione legge un byte alla volta (TCP ha il buffer di ingresso vuoto).

## Controllo di congestione in TCP

TCP contribuisce attivamente al controllo di congestione con accorte politiche di trasmissione. Esistono tecniche di controllo di congestione open loop e closed loop. Le tecniche closed loop sono basate su questo: la congestione viene notificata e sulla base di questa notifica si intraprendono delle azioni. TCP può inferire che la rete è congestionata osservando la perdita dei pacchetti. Lui sa che ciascun pacchetto che spedisce dovrà essere riscontrato, e può osservare che ci sono dei pacchetti che si perdono. Con le tecnologie attuali possiamo pensare che la probabilità di perdita di un pacchetto a causa di errori di trasmissione sia bassa, quindi se un pacchetto si perde probabilmente c'è una congestione. Questa ipotesi è ragionevole sulle reti locali wired, o anche su una dorsale di un provider. Ma se siamo su una wi-fi? La probabilità di perdita per un errore in trasmissione non è così trascurabile, anche se nelle reti wi-fi la rilevazione degli errori e la ritrasmissione può essere affidata al livello 2 (quindi non si vedono le perdite, ma solo dei ritardi nella trasmissione). Si assume perso un pacchetto che non viene riscontrato prima del suo timeout, o per il quale vengono ricevuti ACK duplicati (generalmente 3). Perché anche a presenza di ACK duplicati può essere segno di una congestione? TCP deve generare un ACK (duplicato) quando arriva un pacchetto fuori sequenza; questo tipo di ACK non può essere ritardato. Quando arriva un pacchetto fuori sequenza possiamo solo dire "ho ricevuto correttamente i pacchetti fino al numero x", e lo si dice anche se questo lo si era già detto in passato (magari il pacchetto x era stato già ricevuto). L'ACK duplicato è stato ideato per i pacchetti fuori sequenza, ma può essere usato anche per pacchetti persi. Se arrivano un certo numero (generalmente 3) di ACK duplicati è una forte evidenza della perdita del pacchetto.



Figura 92 - Controllo di congestione

Questo è un modo più rapido per capire se c'è una congestione, invece di aspettare il timeout (che è un tempo generalmente alto).

## Finestra di controllo di congestione

TCP usa una seconda finestra scorrevole per il controllo di congestione ( $cwnd$ ). Quando TCP capisce che la rete è congestionata modifica l'ampiezza della finestra  $cwnd$ . Dal punto di vista di un'entità TCP, mentre la finestra per il controllo di flusso rappresenta una limitazione del mio interlocutore, la finestra di controllo di congestione costituisce una mia auto-limitazione.

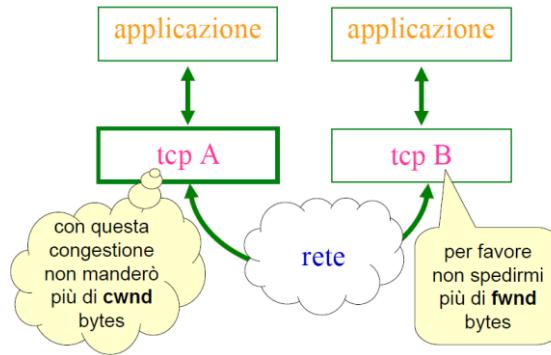


Figura 93 - fwnd e cwnd

A ha una finestra che gli viene notificata da B, e una finestra che gestisce in proprio, che modifica in funzione delle proprie considerazioni sulla rete. La finestra che viene usata per spedire è la più piccola tra le due.

Esempio:

- il ricevente ha 8Kbyte di buffer libero e quindi mostra al mittente una finestra di 8Kbyte;
- il mittente però ha una finestra di controllo di congestione di 4Kbyte;
- il mittente invia al massimo 4Kbyte.

Esempio:

- il ricevente ha 8Kbyte di buffer libero e quindi mostra al mittente una finestra di 8Kbyte;
- il mittente ha una finestra di controllo di congestione di 32Kbyte
- il mittente invia al massimo 8Kbyte.

#### Algoritmo di slow start

Come si sceglie il valore giusto per cwnd? Questa è la cosa più importante, perché se ci si sbaglia in alto si genera una montagna di pacchetti che vengono rigettati, se ci si sbaglia in basso non si sfrutta appieno il bus, quindi ci vuole una buona idea. La buona idea (che è più un punto di partenza) è rappresentata dall'algoritmo slow start.

Quando si instaura una connessione la finestra di congestione cwnd è inizializzata alla dimensione del massimo segment utilizzabile (mss). All'inizio quindi la scelta è abbastanza prudente: un pacchetto. Alcuni inizializzano a 2 mss per partire un po' più velocemente sulla rete. Questa soluzione viene sfruttata da chi fa content delivery, perché ha tutto l'interesse a guadagnare più banda possibile nel più breve tempo possibile. Ogni segmento riscontrato prima del timeout fa aumentare la finestra di congestione di mss, anche se il segment riscontrato non conteneva tutti i byte di un mss. Se un pacchetto viene perso (timeout o ack duplicati) allora cwnd viene riportata al valore iniziale. Se la trasmissione procede senza perdita di pacchetti la finestra cwnd cresce esponenzialmente. Perché esponenzialmente? Bisogna stimare una dimensione, un numero, e lo si deve stimare nel più breve tempo possibile. Con una crescita esponenziale

(1, poi 2, 4, 8...) si arriva a scegliere la dimensione corretta con un numero di tentativi che è logaritmico, con il minimo numero di tentativi si arriva alla stima corretta.

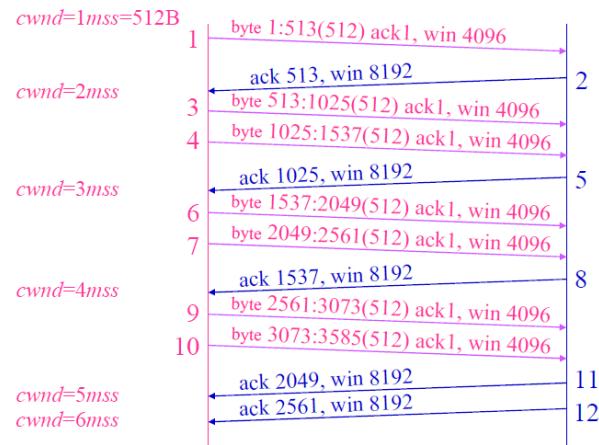


Figura 94 - Esempio di slow start

- Supponiamo che all'inizio cwnd valga 1 segment;
- viene spedito 1 segment:
  - supponiamo sia riscontrato;
  - cwnd sale a 2 segment;
- vengono spediti 2 segment:
  - supponiamo siano entrambi riscontrati;
  - cwnd sale a 4 segment;
- vengono spediti 4 segment:
  - supponiamo siano tutti e 4 riscontrati
- ...

## Congestion avoidance

Oltre a cwnd la finestra di controllo di congestione è gestita usando un ulteriore parametro: ssthresh (soglia). Questo permette di usare un algoritmo slow start unitamente ad una politica di congestion avoidance.

1. inizializzazione di cwnd a mss;
2. quando un pacchetto è riscontrato cwnd cresce;
  - se cwnd <= ssthresh allora cwnd aumenta di un mss (slow start);
  - se cwnd > ssthresh allora cwnd aumenta di mss<sup>2</sup>/cwnd (congestion avoidance);
3. se si verifica una congestione (timeout o ack duplicati) allora:
  - ssthresh è portata alla metà del minimo tra la finestra di controllo di flusso e la cwnd corrente (ma deve valere almeno 2mss);
  - cwnd è portata a mss;

L'effetto dell'algoritmo è quello di far crescere esponenzialmente la finestra di controllo di congestione fino alla soglia, e linearmente dopo la soglia. Cosa succede in presenza di congestioni? Non soltanto si porta cwnd in basso, ma si abbassa anche la soglia; si fa crescere l'ambito nel quale la crescita della finestra di controllo di congestione è lineare. Supponiamo di essere nello stato di congestion avoidance e che la finestra valga cwnd. Possiamo spedire  $\frac{cwnd}{mss}$  pacchetti; supponiamo siano tutti riscontrati. Ognuno fa crescere la finestra di  $\frac{mss^2}{cwnd}$ , quindi la finestra cresce globalmente di  $\left(\frac{cwnd}{mss}\right)\left(\frac{mss^2}{cwnd}\right) = mss$ .

Attenzione: in realtà il comportamento approssima la formula precedente, perché ad ogni pacchetto riscontrato si effettua  $cwnd = cwnd + \frac{mss^2}{cwnd}$ .

Mettendo tutto insieme:

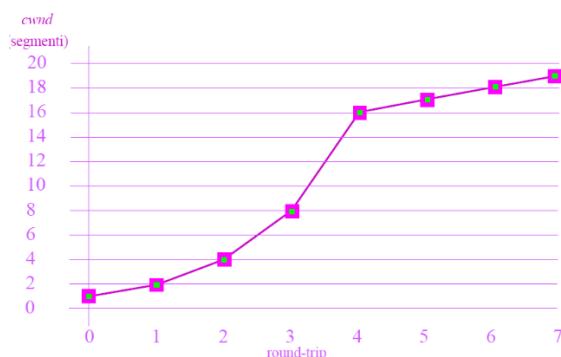


Figura 95 - Esempio di congestion avoidance

Tutto questo si chiama *TCP Tahoe*:

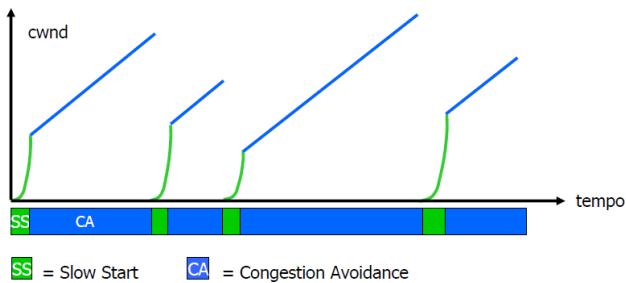


Figura 96 - TCP Tahoe

### Timeout e ritrasmissione

Il timer di ritrasmissione viene lanciato quando viene spedito un segment. Se il segment non è riscontrato entro il tempo in cui il timer di ritrasmissione raggiunge il timeout allora il segment viene ritrasmesso. La scelta del timeout è cruciale:

- se è troppo breve si rischia di congestionare la rete;
- se è troppo lungo si perde di efficienza

È molto più difficile scegliere un timeout di trasporto che di data link. Su un singolo link:

- il tempo di arrivo di un ack è facilmente predicibile;
- il valore di timeout ideale è poco più del valore atteso del tempo di ack;

sull'intera rete:

- la varianza del tempo di ack è molto maggiore;
- varianza e valore atteso del tempo di ack cambiano nel tempo;
- conseguenza: il timeout di TCP viene cambiato continuamente.

Per misurare il ritardo roundtrip si usa l'algoritmo di Jacobson. Per ogni connessione viene gestita la variabile rtt (roundtrip time):

- rtt è la stima corrente del tempo per ricevere l'ack dal destinatario;
- quando un segment viene riscontrato in tempo si dispone del suo tempo M di roundtrip;
- rtt viene aggiornato con  $rtt = \alpha rtt + (1-\alpha) M$ :
  - $\alpha$  misura quanto rapidamente tenere conto delle variazioni;
  - normalmente  $\alpha = 7/8$ .

rtt viene aggiornato con una specie di filtro passa basso. Quanto basso? Ce lo dice  $\alpha$ . Una volta calcolato rtt si ha timeout =  $\beta$  rtt. Abbattere una connessione TCP non è soltanto fare il 3wh una volta di più, ma bisogna anche riportare gli stimatori di TCP nello stato giusto. Nelle prime implementazioni di TCP si sceglieva  $\beta=2$ , poi si è osservato che un valore costante non risultava efficace.  $\beta$  dovrebbe dipendere dalla

deviazione standard della distribuzione del tempo di ack; varie implementazioni TCP usano  $timeout = rtt + 4D$ . D, detta deviazione media, è una stima velocemente calcolabile della deviazione standard:

$$D = \alpha D + (1 - \alpha)|rtt - M|$$

- il parametro  $\alpha$  può essere diverso da quello usato per rtt;
- D può essere calcolato usando solo addizioni e sottrazioni tra interi e shift.

### Algoritmo di Karn

Quando un segment è ritrasmesso non è chiaro a quale istanza di trasmissione si riferisca un ack; ciò può influenzare il calcolo di rtt. Con l'algoritmo di Karn i segmenti ritrasmessi non concorrono al calcolo di rtt. Il timeout è raddoppiato ogni volta che un segment non riesce a passare, fino a che non arriva correttamente (*exponential backoff*).

### Fast recovery e fast retransmit

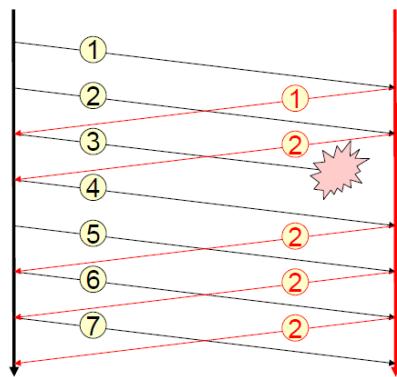


Figura 97 - ACK duplicati

Quando arriva un pacchetto fuori sequenza (cioè contenente byte successivi a quelli attesi), viene generato immediatamente un ack (ack duplicato); questo tipo di ack non viene mai ritardato. La ricezione di un ack duplicato potrebbe dipendere da un'inversione di ordine dei pacchetti nel viaggio verso la destinazione: in questo caso arriverà presto un ack che riscontra i byte successivi. La ricezione di vari (generalmente 3) ack duplicati con lo stesso numero è una forte evidenza del fatto che un pacchetto si è perso.

- Algoritmo fast retransmit: il segment che sembra essersi perso viene ritrasmesso immediatamente, senza attendere che arrivi il timeout (il timeout rischia di essere troppo lento).
- Algoritmo fast recovery: non si porta al minimo cwnd, ma si esegue congestion avoidance. Perché? Si suppone che si sia perso un solo segment e che quindi, visto che i dati continuano a fluire, sia una misura troppo drastica far ripartire slow start.

Normalmente si usano questi due algoritmi assieme. Quando si riceve il terzo ack duplicato:

- si modifica ssthresh come per congestion avoidance;

- ssthresh è portata alla metà del minimo tra la finestra di controllo di flusso e la cwnd corrente (ma deve valere almeno 2mss);
- si ritrasmette il segment mancante;
- si fa crescere cwnd in modo lineare.

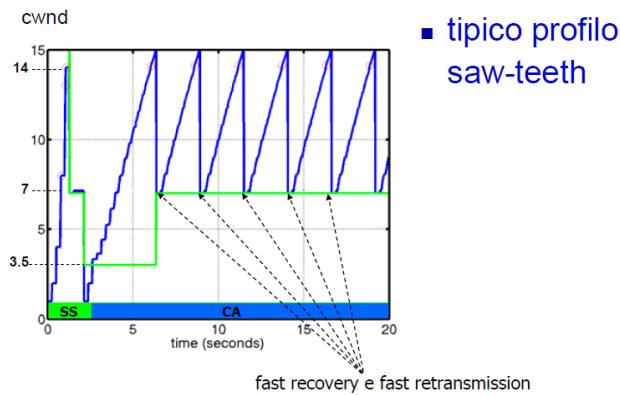


Figura 98 - TCP reno: saw-teeth

## Comprendere il TCP

### TCP onnisciente

Qual è la scelta migliore che possiamo fare? Siamo di fronte ad un TCP che deve spedire una grande quantità di dati, come possiamo fare a massimizzare il throughput? Qui c'è un TCP "finto", e vediamo come un TCP può sfruttare una conoscenza completa della rete, con delle ipotesi estremamente semplificative:

- routing stabile;
- TCP sa tutto delle caratteristiche della rete (o meglio, del cammino fino alla destinazione);
- non c'è traffico.

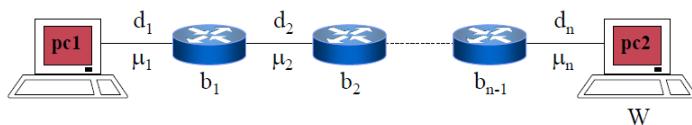


Figura 99 - Collegamento tra due PC

Visto che il routing non cambia la rete è un cammino, semplicemente. Ci interessa il numero dei router, la banda dei collegamenti attraversati (di  $\mu_i$  sono i ritardi e le velocità dell'i-esimo hop). Siamo su scala geografica, i ritardi possono non essere trascurabili. I  $b_i$  sono rappresentano il numero di bit che possono essere messi nella coda dell'i-esimo router.  $W$  è la dimensione della finestra di controllo di flusso di PC2, e supponiamo che questa dimensione non cambi nel tempo (non cambia la sua disponibilità nel tempo). Supponiamo che PC1 debba spedire a PC2 un quantitativo molto significativo di pacchetti, e facciamo l'ulteriore ipotesi semplificativa che ogni pacchetto abbia esattamente 1 bit (nella pratica i pacchetti non sono tutti uguali, ma con un grande flusso di pacchetti si tende a farli tutti della massima dimensione); supponiamo inoltre che i riscontri abbiano 0 bit. Dobbiamo calcolare un upper bound per il throughput. A regime, quanti bit al secondo può spedire PC1 a PC2? Tra queste bande disponibili ce n'è una che è la più piccola di tutte, non possiamo spedire più di:

$$\underline{\mu} = \min_i(\mu_i) \text{ bit/sec}$$

altrimenti il buffer crescerebbe all'infinito e butteremmo molti pacchetti. Il TCP di PC1 ha la mappa completa della rete, quindi sa pure qual è il link più lento. A questo punto TCP dovrebbe spedire pacchetti a questa velocità, ma purtroppo il TCP di PC1 non può modificare la velocità della sua interfaccia di rete; per adattarsi bene a questa rete PC1 deve spedire un massimo di  $\mu/l$  pacchetti al secondo, cioè un pacchetto ogni  $l/\mu$  secondi. Se devo minimizzare l'occupazione dei buffer dei router devo mandare i pacchetti temporalmente equi spaziati, in modo tale da mettere poco alla prova il buffer dei router. Questo è quello che vorrebbe fare TCP. Un possibile piano di spedizione che potrebbe elaborare TCP è questo:

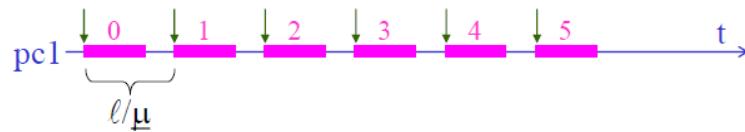


Figura 100 - Piano di spedizione per elaborare TCP

Prima spedisce il primo pacchetto, poi i successivi, scegliendo arbitrariamente l'istante in cui spedire ciascun pacchetto. Come fa a fare questo piano di spedizione ideale? Per gestire questo piano per bene il TCP di PC1 dovrebbe scegliere proprio l'istante in cui spedire ciascun pacchetto tramite un timer. Se i pacchetti sono spediti equi spaziati ogni router ha al più un pacchetto in coda; qual è il ritardo roundtrip di un pacchetto in questa storia?

$$\text{latenza minima} = \frac{l}{\mu_1} + \dots + \frac{l}{\mu_n} + 2d_1 + 2d_n$$

Sfortunatamente, per TCP è improponibile spedire pacchetti a istanti di tempo prefissati. Questo vorrebbe dire, da parte del sistema operativo, gestione di interruzioni di clock sincrone specifiche per ciascuna spedizione. Per quanto si elabori un piano di spedizione preciso ci potrebbero essere dei conflitti tra le connessioni. Ci siamo messi nelle condizioni migliori possibili, come fa TCP a implementare quel piano di spedizione? La risposta è una caratteristica di TCP: riesce ad auto temporizzarsi (self-clocking), in modo tale che i pacchetti si allineino tutti quanti alla velocità di spedizione.

#### Self-clocking

Come riesce TCP a spedire, approssimativamente, un pacchetto ogni  $l/\mu$  secondi? Per capire questo comportamento consideriamo un secondo scenario:



- speed of the first hop (pc1-r1):  $\mu_1 = 10 \text{ Mbit/sec} = 10^7 \text{ bit/sec}$
- speed of the second hop (pc2-r1):  $\mu_2 = 5 \text{ Mbit/sec} = 5 \cdot 10^6 \text{ bit/sec}$
- delay from pc1 to r1:  $d_1 = t_p = 1 \text{ ms} = 10^{-3} \text{ sec}$
- delay from r1 to pc2:  $d_2 = t_p = 1 \text{ ms}$
- size of the packets:  $l = 1,000 \text{ bits} = 10^3 \text{ bits}$
- size of the acks: “0 bits”

Figura 101 - Scenario di due computer e un router

PC1 a regime deve mandare fuori 5 Mb/s, ma ha una NIC da 10 Mb/s. Supponiamo che il 3wh sia stato già fatto, e che slow start e congestion avoidance non ci siano. Supponiamo che la coda di r1 abbia una dimensione infinita, e che il TCP di PC1 debba spedire una grande quantità di dati. PC2 mostra una finestra di controllo di flusso di 10 pacchetti (10.000 bit), che non cambia.

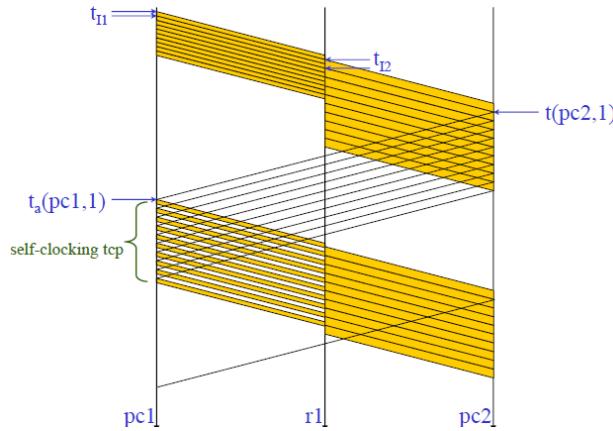


Figura 102 - Notazioni

Un po' di notazione:

- $t(r_1, n)$  è l'istante in cui  $r_1$  riceve l'intero pacchetto  $n$ ;
- $t(pc_2, n)$  è l'istante in cui  $pc_2$  riceve l'intero pacchetto  $n$ ;
- $t_a(pc_1, n)$  è l'istante in cui  $r_1$  riceve il riscontro per il pacchetto  $n$ .

$pc_1$  spedisce 10 pacchetti consecutivi, poi si ferma. Potrà rimettersi a spedire quando si avrà il primo ack sui pacchetti spediti. Quanto ci vuole a spedire un pacchetto?

$$t_{I1} = 10^3 / 10^7 = 10^{-4} \text{ sec}$$

Abbiamo 10 pacchetti, quindi per spedirli tutti servono

$$10t_{I1} = 10^{-3} \text{ sec}$$

Nel transitorio  $pc_1$  ha spedito tutti i pacchetti uno appresso all'altro.  $r_1$  li invia alla velocità del secondo link. I riscontri vengono inviati quando il pacchetto viene ricevuto, e  $pc_1$  allinea la spedizione dei propri pacchetti con i riscontri che arrivano. Con gli ack più grandi di 0 bit il comportamento non cambia troppo. Questo è lo stesso meccanismo del go-back-n. Lì c'è una pausa della trasmissione di  $pc_1$ , che dipende da molti fattori. Se ci sono  $n$  linee il discorso non cambia, il delay che domina tutti gli altri è sempre quello del collegamento più lento. Un po' di conti:

- $t_{I2} = 2 t_{I1}$
- Il primo pacchetto arriva all'istante:
  - $t(pc_2, 1) = t(r_1, 1) + t_{I2} + t_p = t_{I1} + t_p + t_{I2} + t_p = 10^{-4} + 10^{-3} + 10^3 / (5 \cdot 10^6) + 10^{-3} = 10^{-4} + 10^{-3} + 0.2 \cdot 10^{-3} + 10^{-3} = 2.3 \cdot 10^{-3} \text{ sec}$
- the first ack arrives:
- $t_a(pc_1, 1) = t(pc_2, 1) + 2 \cdot t_p = 2.3 \cdot 10^{-3} + 2 \cdot 10^{-3} = 4.3 \cdot 10^{-3} \text{ sec}$
- $pc_1$  può inviare l'undicesimo pacchetto all'istante  $t_a(pc_1, 1)$ 
  - $pc_2$  riceve il secondo pacchetto all'istante:
    - $t(pc_2, 2) = t(r_1, 1) + 2 \cdot t_{I2} + t_p = t(pc_2, 1) + t_{I2} = 2.3 \cdot 10^{-3} + 0.2 \cdot 10^{-3} \text{ sec} = 2.5 \cdot 10^{-3} \text{ sec}$
  - $pc_1$  riceve il secondo ack:
    - $t_a(pc_1, 2) = t(pc_2, 2) + 2 \cdot t_p = 2.5 \cdot 10^{-3} + 2 \cdot 10^{-3} = 4.5 \cdot 10^{-3} \text{ sec}$
    - $t_a(pc_1, 2) - t_a(pc_1, 1) = 0.2 \cdot 10^{-3} \text{ sec} = t_{I2}$
  - d'ora in poi  $pc_1$  invia, durante il burst, un pacchetto ogni  $0.2 \cdot 10^{-3} \text{ sec}$ , quindi  $pc_1$ , durante i burst, non userà la sua nic a 10 Mb/sec ma solo a  $1,000 \text{ bit} / 0.2 \cdot 10^{-3} \text{ sec} = 5 \cdot 10^6 \text{ bit/sec} = 5 \text{ Mb/sec}$

Quando il transitorio termina pc1 trasmette i suoi burst a una velocità media che è quella dell'hop più lento tra quelli attraversati. Quando pc1 ha a disposizione la finestra di controllo di flusso libera spedisce i pacchetti alla velocità della linea 2.

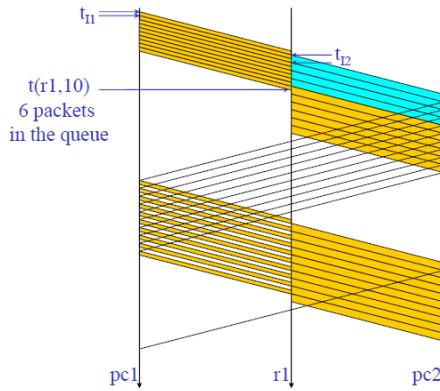


Figura 103 - Coda per r1

Cosa succede alla coda di r1? La coda di r1 ha la sua dimensione massima all'istante

$$t_{(r1,10)} = 10 \cdot t_{l1} + t_p = 10 \cdot 10^{-4} + 10^{-3} \text{ sec} = 2 \cdot 10^{-3} \text{ sec}$$

All'istante  $t_{(r1,10)}$  r1 ha già trasmesso a pc2 per un tempo di  $9 \cdot t_{l1} = 9 \cdot 10^{-4}$  sec. Quindi, all'istante  $t_{(r1,10)}$  r1 ha già trasmesso a pc2 4 pacchetti. Ci sono 6 pacchetti in coda. C'è qualcosa che ancora non torna, la lunga pausa. Fa sì che si abbiano i pacchetti equispaziati, ma non si sfrutta appieno la banda disponibile. Dovremmo avere la possibilità di spedire abbastanza pacchetti fino all'arrivo del primo riscontro. Cosa si può fare? Dobbiamo "discutere" con pc2, ci deve far vedere una finestra di controllo di flusso più ampia. Mettiamoci in una situazione in cui la finestra di controllo di flusso di pc2 è molto abbondante, ma comunque non posso andare alla velocità maggiore di ciò che consente  $\mu$ . Posso scegliere una dimensione della finestra tale da permettere di spedire pacchetti fino all'istante in cui pc1 riceve l'ultimo riscontro. Qual'è il valore ottimale per W dal punto di vista del TCP di pc1?

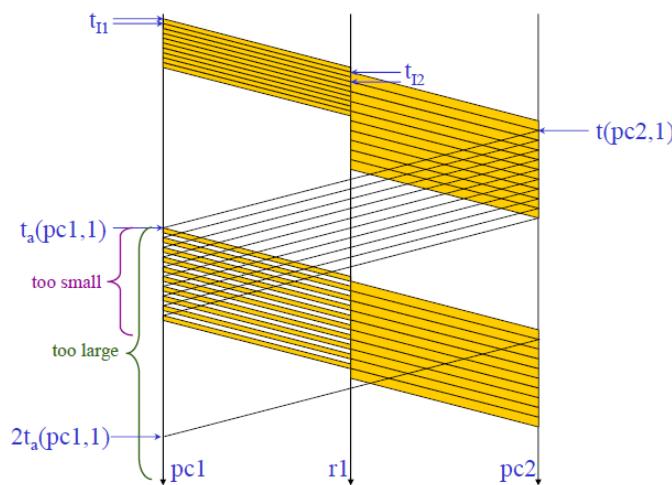


Figura 104 - Valori troppo stretti o grandi per W

Se la finestra è troppo grande continuo a spedire pacchetti senza motivo, se è troppo piccola ho una pausa che vorrei evitare.

Qual è il valore ottimale per W dal punto di vista del TCP di pc1? Ci sono due casi:

- se  $W/l \cdot t_{l2} < t_a(\text{pc1},1)$ 
  - dobbiamo aspettare un po' quando ciascun burst è terminato
- if  $W/l \cdot t_{l2} > t_a(\text{pc1},1)$

si rischia (nel transitorio) di sovraccaricare le code dei router attraversati. Nello stato di attesa dovremmo comunque avere un comportamento self-clocking.

Faccio in modo che  $\frac{W}{l} \cdot t_{l2} = t_a(\text{pc1},1)$

ricordando che  $t_a(\text{pc1},1) = t(\text{pc2},1) + 2t_p = t_{l1} + t_p + t_{l2} + t_p + 2t_p$  abbiamo che

$$\frac{W}{l} \cdot t_{l2} = t_{l1} + t_{l2} + D$$

dove D è il ritardo roundtrip. Trascurando il termine  $t_{l1} + t_{l2}$  si ha:

$$\frac{W}{l} = \frac{D}{t_{l2}} = D \cdot \mu$$

Questo  $D\mu$  è un numero magico delle reti, il prodotto banda-latenza, ed è il valore ottimale della finestra di controllo di flusso, quello che dà luogo al miglior piano di spedizione possibile.

## TCP in una rete stabile

TCP non è onnisciente, vediamo come si comporta in una rete stabile. Come selezionare per TCP il valore della finestra per il self-clocking? TCP può fare affidamento su due informazioni: i pacchetti si perdono oppure i pacchetti non si perdono. A partire da un valore iniziale, TCP può incrementare W fino a che un pacchetto si perde. Questo incremento dovrebbe essere abbastanza veloce da approssimare il W ottimale nel minor numero di round. Possiamo calcolare la cwnd come segue. All'istante iniziale  $cwnd_0 = 1 \text{ mss}$ . In generale:

$$cwnd_i = cwnd_{i-1} + \frac{mss^2}{cwnd_{i-1}}$$

Ponendo  $cwnd_i = x_i \text{ mss}$  si ha

$$x_i \text{ mss} = x_{i-1} \text{ mss} + \frac{mss^2}{cwnd_{i-1}}$$

$$x_i \text{ mss} = x_{i-1} \text{ mss} + \frac{mss^2}{x_{i-1} \text{ mss}}$$

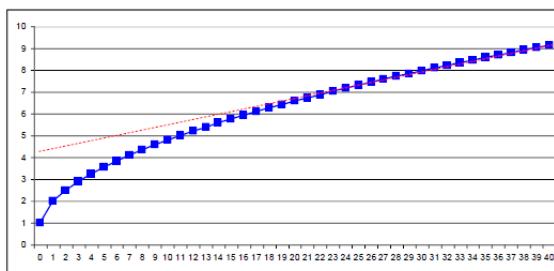
$$x_i = x_{i-1} + \frac{1}{x_{i-1}}$$

con  $x_0 = 1$

i	$x_i$	cwnd	i	$x_i$	cwnd
0	1	1000	18	6,261351	6261,351
1	2	2000	19	6,421061	6421,061
2	2.5	2500	20	6,576799	6576,799
3	2.9	2900	21	6,728848	6728,848
4	3,244828	3244,828	22	6,877462	6877,462
5	3,55301	3553,01	23	7,022865	7022,865
6	3,834462	3834,462	24	7,165257	7165,257
7	4,095255	4095,255	25	7,304819	7304,819
8	4,33944	4339,44	26	7,441715	7441,715
9	4,569884	4569,884	27	7,576093	7576,093
10	4,788708	4788,708	28	7,708087	7708,087
11	4,997533	4997,533	29	7,837821	7837,821
12	5,197631	5197,631	30	7,965407	7965,407
13	5,390027	5390,027	31	8,090950	8090,950
14	5,575555	5575,555	32	8,214545	8214,545
15	5,754909	5754,909	33	8,336280	8336,280
16	5,928674	5928,674	34	8,456238	8456,238
17	6,097345	6097,345	35	8,574494	8574,494

32

Figura 105 - Tabella che descrive l'evoluzione di cwnd



$x_i$  as a function of  $i$  (cwnd<sub>i</sub> is mss times bigger)

Figura 106 - Grafico dell'evoluzione di cwnd

L'andamento degli  $x_i$  e della cwnd assomiglia ad un andamento logaritmico.

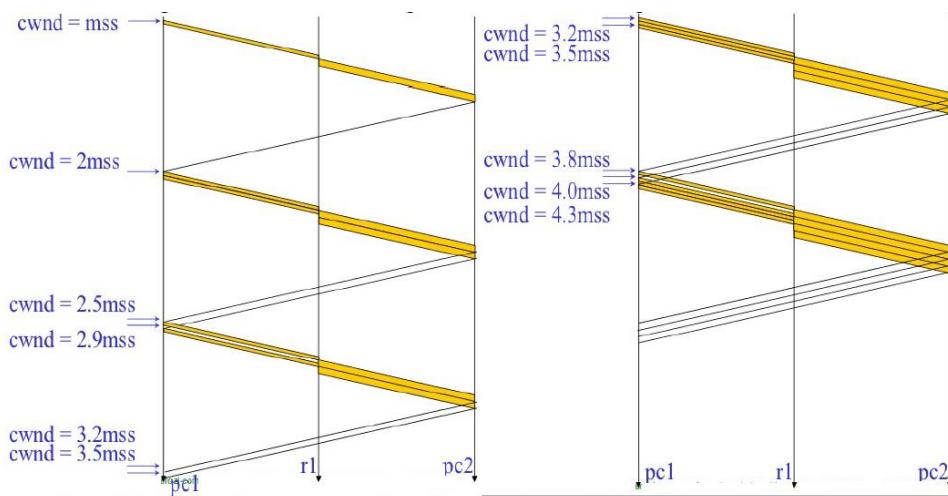


Figura 107 - Andamento cwnd

Non è una buona idea incrementare W in questo modo. Pc1 invia al più 3 pacchetti consecutivi per volta, quindi lo spazio occupato nella coda dei router è di al più 3 pacchetti. Ad un certo punto cwnd raggiunge W e smette di crescere. Se W è grande (diciamo infinita) ad un certo punto cwnd cresce troppo e alcuni

pacchetti vengono persi. Questo è il modo in cui TCP capisce che cwnd è troppo grande. TCP raggiunge questo momento in un numero di turni lineare con la dimensione ottimale di cwnd, e questo non è ammissibile. Si può fare di meglio? Siamo in una situazione simile a questa: ci sono due persone, A e B. A pensa un numero, e B deve indovinarlo. A risponde "OK" se B indovina, "<" se indica un numero minore di quello pensato e maggiore se indica un numero maggiore. Qual è la strategia migliore per indovinare nel minor numero di tentativi? Una strategia esponenziale (slow start) permette di trovare il valore ottimale di cwnd in un numero logaritmico di tentativi. TCP combina l'approccio esponenziale con quello lineare: slow start per un'approssimazione veloce della dimensione ottimale e congestion avoidance per una stima più accurata.

Durante una connessione TCP la rete cambia varie volte: in particolare cambia la banda disponibile, lo spazio disponibile nella coda che attraversa il router.

### TCP reno

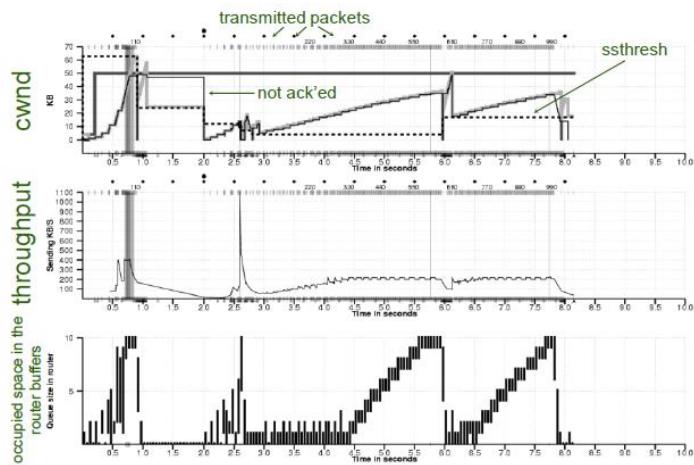


Figura 108 - TCP reno

Il comportamento tipico di TCP Reno è il seguente: all'inizio della connessione si fa slow start, per identificare velocemente la larghezza di banda disponibile. A regime permanente si fa fast-recovery e fast retransmit. È molto raro che un pacchetto perso venga ritrasmesso dopo un timeout. Il sintomo di perdita di un pacchetto sono i soliti 3 ack duplicati, e quello che si ottiene è un andamento a dente di sega. Il pattern sawtooth si chiama AIMD (additive increase multiplicative decrease); se non ci sono pacchetti perduti si fa crescita lineare, quando un pacchetto viene perduto ssthresh diventa la metà di cwnd.

### AIMD

Il comportamento tipico di TCP Reno è il seguente: all'inizio della connessione si fa slow start, per identificare velocemente la larghezza di banda disponibile. A regime permanente si fa fast-recovery fast retransmit. È molto raro che un pacchetto perso venga ritrasmesso dopo un timeout. Il sintomo di perdita di un pacchetto sono i soliti 3 ack duplicati, e quello che si ottiene è un andamento a dente di sega. Il

pattern sawtooth si chiama AIMD (additive increase multiplicative decrease); se non ci sono pacchetti perduti si fa crescita lineare, quando un pacchetto viene perduto  $ssthresh$  diventa la metà di  $cwnd$ .

Mettiamo il tutto su un piano:

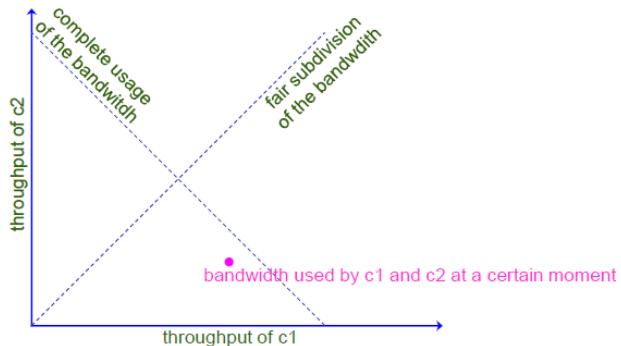


Figura 109 - Piano di AIMD

Il throughput è la banda che  $c1$  e  $c2$  stanno sfruttando in un dato istante. Nella pratica la banda disponibile su una linea può oscillare, quindi la linea "complete usage..." potrebbe non essere fissa, come stiamo ipotizzando. Che succede quando due connessioni fanno AIMD contemporaneamente? Succede che se stanno nella crescita AIMD (stanno facendo congestion avoidance) fanno tutte e due crescita lineare, quindi aumentano linearmente la finestra di controllo di congestione, avvicinandosi alla linea di utilizzo completo della banda.

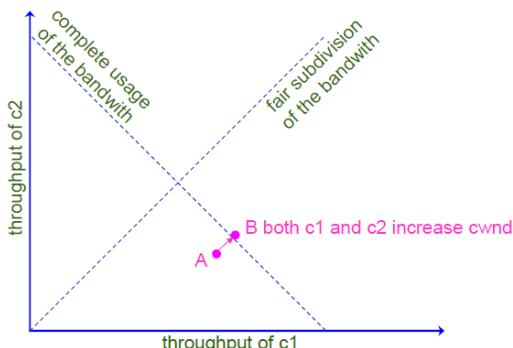


Figura 110 - Due connessioni che formano AIMD

A questo punto tutte e due le connessioni sono arrivate al limite. Se superiamo la linea di massimo utilizzo magari nel primo istante i router ce lo permettono, perché c'è spazio nel buffer, ma un attimo dopo i pacchetti vengono buttati (se si fa random early drop vengono buttati un attimo prima che si raggiunga la banda massima).

Quando la perdita viene percepita c1 e c2 dimezzano la finestra di controllo di congestione:

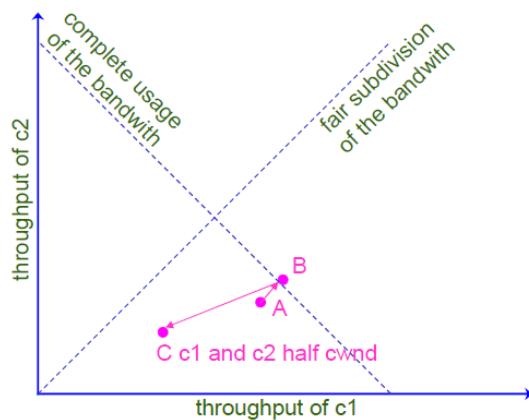


Figura 111 - Due connessioni che formano AIMD

per poi ricominciare a farla crescere in modo additivo, riportandosi verso la linea di completo utilizzo della banda.

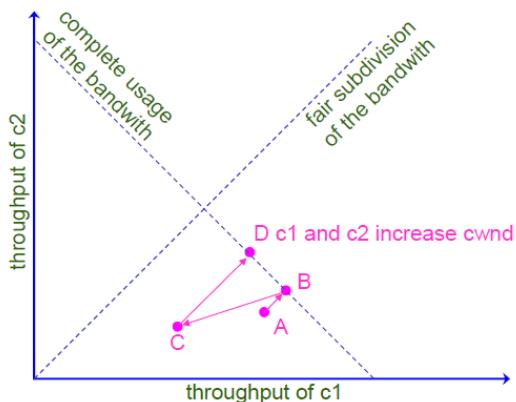


Figura 112 - Due connessioni che formano AIMD

Questa decrescita permette di approssimare pian piano una divisione equa della banda. È anche vero che si continua a rimbalzare avanti e indietro, chi ce lo fa fare? Una volta che abbiamo capito qual è l'utilizzo ottimale potremmo rimanere lì. La linea di massimo utilizzo però non è fissa: se la banda aumenta dobbiamo essere in grado di prendercene di più, e allo stesso modo se si abbassa dobbiamo ridurre le nostre pretese, o si perderebbero pacchetti. Nell'additive increase la linea di incremento è una linea a 45°, perché tutte le connessioni fanno  $cwnd=cwnd+1$  ad ogni roundtrip; nel multiplicative decrease succede questo:

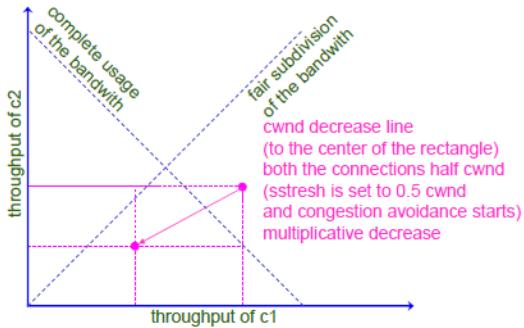


Figura 113 - Decremento moltiplicativo

Al roundtrip successivo ci si mette al centro di quel rettangolo. Questo meccanismo è corretto perché le due connessioni si muovono progressivamente verso la linea di condivisione corretta della connessione.

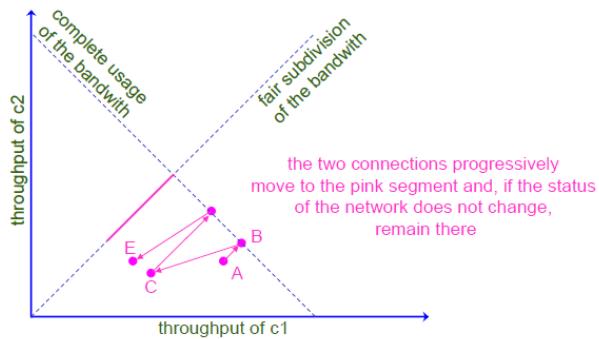


Figura 114 - Equità di AIMD

Se lo stato della rete non cambia le due connessioni rimangono su quel segmento. Ci sono delle alternative, che dovremo scartare. Prima di scartarle apprezziemo con un calcolo facilissimo il throughput che abbiamo nella fase AIMD. Se la rete è stabile cwnd si muove tra due valori:  $cwnd_{max}/2$  e  $cwnd_{max}$ . Quindi, se rtt è abbastanza grande, il throughput:

$$\frac{cwnd_{max}}{2rtt} < \text{throughput} < \frac{cwnd_{max}}{rtt}$$

Dal momento che cwnd cambia linearmente, possiamo stimare che il throughput sia, mediante:

$$\frac{3}{4} \cdot \frac{cwnd_{max}}{rtt}$$

## AIAD

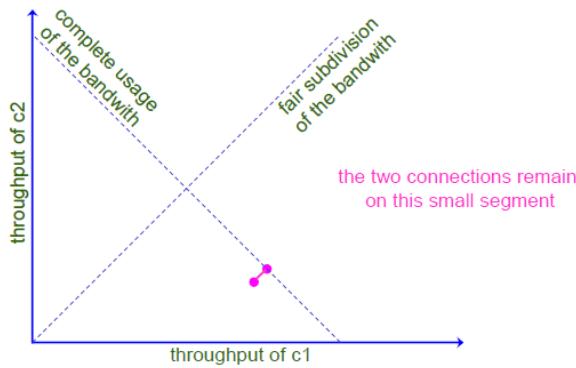


Figura 115 - AIAD

Se si fa AIAD non ci si sposta mai da quel segmento, quindi l'idea di fairness ce la perdiamo.

## MIMD

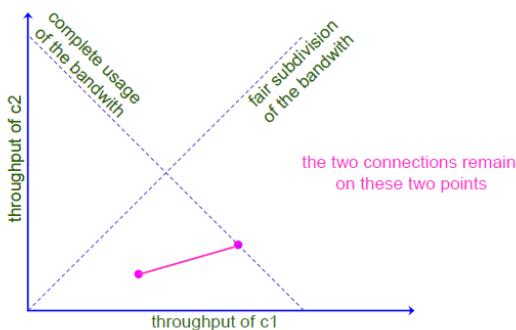


Figura 116 - MIMD

Le due connessioni rimangono comunque su quel segmento.

## MIAD

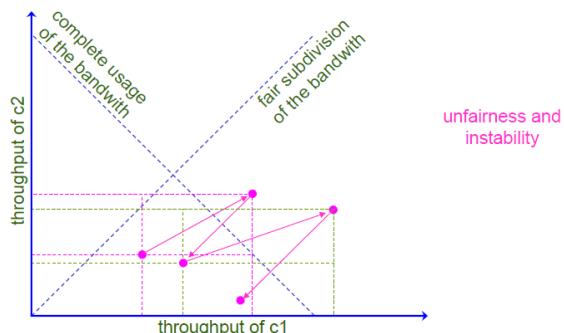


Figura 117 - MIAD

Si va verso l'unfairness, verso una situazione in cui una delle due macchine assorbe tutta la banda. Supponiamo che le due connessioni abbiano un tempo diverso di percezione della perdita di un pacchetto.

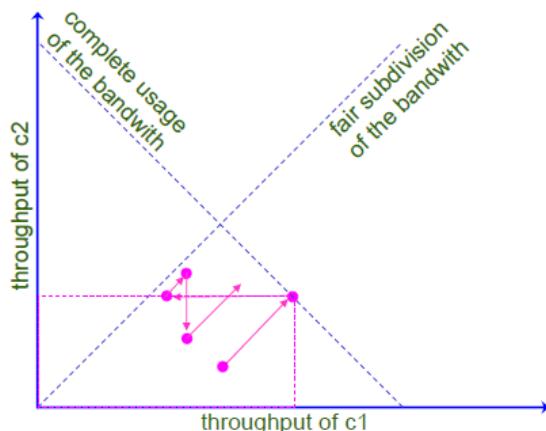


Figura 118 - MIAD

## Algoritmi per l'instradamento per l'infrastruttura di una rete fissa

### Qualità degli algoritmi di instradamento

I router devono fare il forwarding dei pacchetti per la maggior parte del tempo, e vorremmo quindi che gli algoritmi siano efficienti, cioè che questa cosa sia evitata: per la maggior parte del tempo dobbiamo calcolare le tabelle di instradamento. Le cpu e le memorie attualmente disponibili sui router sono talvolta insufficienti se confrontati con la complessità delle reti. Vorremmo in qualche modo misurare l'ottimalità degli algoritmi:

- nella scelta dei cammini, rispetto a qualche criterio;
- i criteri di ottimalità dei cammini devono essere misurabili;
- normalmente si usa il numero di hop o il costo delle linee, talvolta assunto inversamente proporzionale alla velocità;
- criteri che tengano in considerazione il carico corrente della rete sono difficili da considerare.

La qualità di un algoritmo si misura rispetto a:

- robustezza e adattabilità (rispetto alle variazioni di topologia): in una rete di grandi dimensioni possono esserci frequentemente guasti alle linee e/o ai router;
- stabilità: se non cambia la topologia non devono cambiare le tabelle ed a fronte di una variazione di topologia occorre convergere rapidamente ad un nuovo instradamento stabile;
- equità: nessun nodo deve essere privilegiato o danneggiato.

Scegliere un algoritmo però non è facile: spesso i criteri di ottimalità sono in contrasto, e gli algoritmi complessi possono comportare configurazioni difficili.

### Tassonomia degli algoritmi

Gli algoritmi possono essere statici oppure dinamici. Negli algoritmi statici i criteri di instradamento sono fissi e indipendenti dalla topologia:

- algoritmi statici: criteri fissi di instradamento, indipendenti dallo stato della topologia (static routing e flooding);
- algoritmi dinamici: instradamento in funzione dello stato della topologia e/o del carico.

Uno potrebbe dire: il routing o è sempre statico oppure è sempre dinamico, ma non è così: in una parte della rete il routing statico, in un'altra è dinamico:

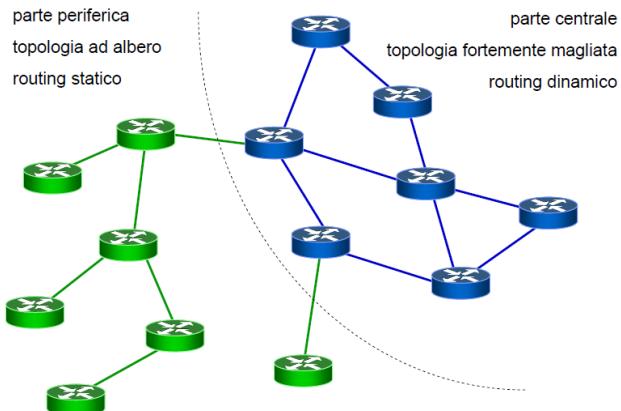


Figura 119 - Rapporto tra routing statico e dinamico

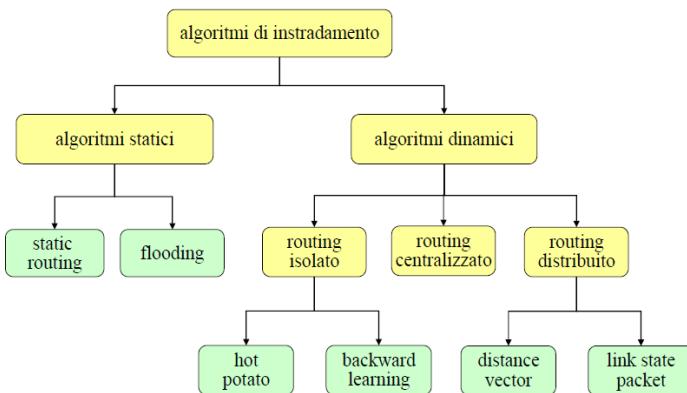


Figura 120 - Tassonomia degli algoritmi di instradamento

Questo è logico: nella parte ad albero non abbiamo grandi gradi di libertà, se cade una linea c'è poco da fare. In alcuni casi comunque si usano algoritmi dinamici anche per topologie ad albero. Nelle parti "magliate" della rete abbiamo diverse alternative, ed è giusto che l'algoritmo sia dinamico. Nonostante questa distinzione l'affermazione "il livello 3 deve essere lo stesso su tutta la rete" rimane valida. Pensiamo al livello 3 di IP, che cosa fa il livello 3? Un intermediate system, quando riceve un pacchetto sceglie una linea e fa l'inoltro. Un altro modo di vedere il livello 3 è: c'è una tabella di instradamento che deve essere compilata. Ciò che è legacy è il formato del pacchetto IP: il formato è quello e le operazioni sono le stesse, ed è vero per tutta la rete vista. Come ogni router si calcola le tabelle di instradamento, e come queste siano mantenute consistenti, non ha importanza. Il calcolo delle tabelle di instradamento nelle diverse tabelle della rete può essere fatto in modo autonomo.

### Algoritmi statici

Static routing vuol dire semplicemente che su ogni nodo la tabella di instradamento è compilata dall'amministratore di sistema. L'amministratore di rete in caso di guasti deve intervenire cambiando fisicamente la tabella. Ci sono varianti quasi statiche: si forniscono più alternative in ordine di priorità, che vengono scelte in funzione dello stato della rete.

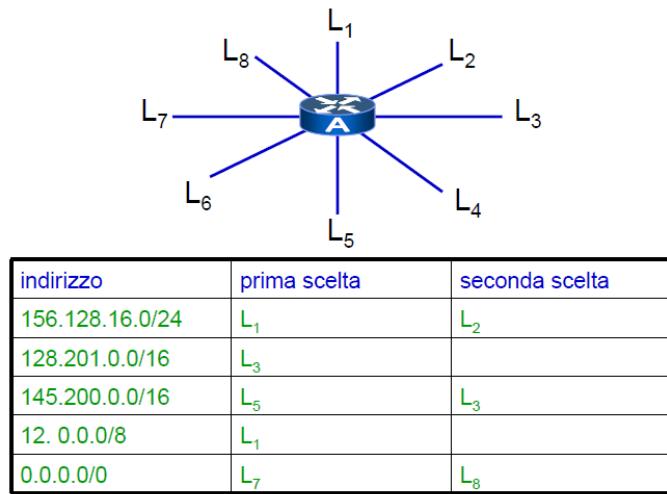


Figura 121 - Static routing

È interessante la prospettiva del flooding. Nel flooding ogni pacchetto viene ritrasmesso su tutte le linee, salvo su quella da dove è arrivato. La rete è così piena di pacchetti. Ci sono poi tutta una serie di varianti: selective flooding: si ritrasmette solo su un insieme di linee selezionato (IS-IS (iso 10598));

- scarto dei pacchetti troppo vecchi (pacchetti con “age counter” a bordo);
- scarto di un pacchetto al suo secondo passaggio su un nodo.

Le varianti necessitano di memorie estese e identificatori di pacchetto.

Algoritmi dinamici

#### *Routing isolato*

Ogni intermediate system (IS) calcola in modo indipendente le proprie tabelle, senza consultare gli altri IS. Un possibile algoritmo è quello che viene chiamato “hot potato”: posso mandare il pacchetto sulla linea con coda più breve. Questo tipo di algoritmo è stato studiato per molto tempo: se la rete è essenzialmente ad albero il pacchetto non sopravvive. Se la rete è sufficientemente connessa la probabilità che un pacchetto arrivi a destinazione è molto alta. Un'alternativa è il backward learning. Il backward learning viene fatto dagli switch, che stabiliscono “questo mac sta sulla porta x, questo sulla porta y” in base agli indirizzi mittente dei pacchetti in ingresso. Un possibile miglioramento è l'aggiunta di un campo nei pacchetti che specifica il costo, incrementato ad ogni attraversamento di IS. In questo modo si possono mantenere negli IS più alternative ordinate per costo, ma così si imparano solo i miglioramenti e non i peggioramenti. Si può risolvere aggiungendo un tempo di scadenza sulle entry. Questa scelta può generare cicli, perciò si accoppia usualmente con il calcolo di spanning tree.

#### *Routing centralizzato*

C'è un routing control center (RCC) a cui affluiscono tutte le informazioni sulla topologia non la rete. L'RCC calcola tutte le tabelle di instradamento e le distribuisce a tutti i router sulla rete. Questa cosa ha grossi problemi di affidabilità: se si ha un RCC si ha un singolo point of failure. Intorno al RCC c'è un

traffico intenso proprio nel momento in cui questo traffico non ci dovrebbe essere, cioè nelle situazioni di fault, che vanno gestite velocemente. Per certe tecnologie però i RCC stanno ricominciando a diventare di qualche interesse.

### *Routing distribuito*

Le funzionalità di RCC vengono svolte da tutti gli intermediate system. Ci sono due principali paradigmi:

- *distance vector*: dico ai miei vicini tutto ciò che so del mondo;
- *link state packet*: dico a tutto il mondo ciò che so dei miei vicini.

### *Algoritmi distance vector*

Ogni *is* invia una tabella (distance vector) a ciascun *is* adiacente. Questa tabella contiene non tutta la tabella di instradamento di chi l'ha inviata, ma la parte essenziale (vengono omessi dettagli locali, come i nomi delle interfacce). gli *is* ricalcolano le tabelle di instradamento fondendo tutte le indicazioni presenti nei vari distance vector ricevuti. Più concretamente: un *is* appena acceso deve costruire la sua tabella di instradamento; si prende quella dei vicini e si costruisce la propria; nella tabella dei vicini c'è:

- l'elenco delle destinazioni che fanno parte della rete;
- l'elenco di tutte le destinazioni;
- per ciascun prefisso la linea che bisogna attraversare per raggiungerlo;
- per ciascuna linea quanto costa raggiungerla.

Immaginiamo di conoscere le tabelle dei vicini. Sappiamo che per raggiungere la destinazione 1 il primo vicino paga 50, il secondo 20, il terzo 100 e si decide di scegliere la destinazione scelta dal secondo vicino. Non paghiamo però 20, ma qualcosa di più; per capire qual è il vicino che conviene al costo di ciascuna linea dobbiamo aggiungere il costo per raggiungere il vicino. Problema: i vicini come hanno costruito la propria tabella di instradamento? Quando un router parte ha le informazioni di raggiungibilità dei propri vicini (allo stato iniziale solo le informazioni delle destinazioni direttamente connesse sono note), e può dirle ad altri vicini; con una specie di passaparola si arriva ad una situazione di equilibrio in cui tutti sanno come raggiungere tutti. A parità di costo possiamo fare una scelta basata sul numero di hop, e in caso di ulteriore parità potremmo fare una scelta random. Le scelte random però non piacciono troppo ai chi progetta reti, perché se si deve fare il debug della rete non siamo in grado di capire esattamente cosa sia accaduto. Si può quindi decidere di fare una scelta deterministica (ad esempio la linea con identificativo più basso). L'invio dei distance vector avviene periodicamente e ad ogni modifica della tabella di instradamento. Non sappiamo chi sono i nostri vicini, quindi si prende il distance vector e si manda a tutti (multicast).

Questi algoritmi convergono lentamente, e sono difficili da usare per reti con più di 1000 nodi. Per capire questo bisogna analizzare un po' meglio questo tipo di algoritmo. Vediamo prima un esempio: siamo nei panni di un router che ha diverse linee, e il router su ogni linea riceve dei distance vector, <rete, costo per raggiungere le reti>.

		distance vector pervenuti sulle linee			
		linea 1	linea 2	linea 3	linea 4
net	cost				
1	0		1 3	1 5	3 5
2	3	2 5	2 2	2 0	4 7
4	2	3 4	3 6	3 2	5 2

Figura 122 - Esempio di distance vector

Queste sono tabelle di instradamento depurate dalle informazioni che non interessano, ci interessa solo quanto paga quel vicino per raggiungere determinate destinazioni. Arrivati i distance vector, ho i costi delle linee attraversate (la linea 1 costa 2, la linea 2 costa 2, la 3 costa 3 e la 4 costa 1). Si fa poi la somma di queste informazioni, ottenendo una tabella che elenca, per ogni rete (prefisso), la linea da usare e il suo costo.

				tabella risultante
sommati i costi delle linee				
costo 2	costo 2	costo 3	costo 1	
net 1 2	net 1 5	net 1 8	net 3 6	net 1 2
2 5	2 4	2 3	4 8	2 3
3 8	3 5	5 3		3 5

Figura 123 - Esempio di distance vector

A questo punto prendo la tabella, tolgo la linea di inoltro perché ai miei vicini non interessa e gliela invio. Cerchiamo ora di capire se l'algoritmo converge velocemente, lentamente, oppure semplicemente se converge. Gli algoritmi distance vector reagiscono rapidamente alle buone notizie (si apre un nuovo collegamento) e lentamente alle cattive notizie (un link funzionante si interrompe). Per capire questa cosa guardiamo un esempio:

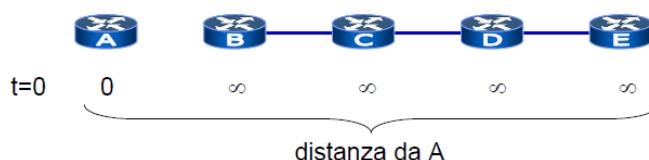


Figura 124 - Problema del count-to-infinity

Ogni router sa che per raggiungere A si paga un costo infinito (quella riga di  $\infty$  è una specie di riassunto delle tabelle di instradamento con riferimento ad A).



	A	B	C	D	E
t=0	0	$\infty$	$\infty$	$\infty$	$\infty$
t=1	0	1	$\infty$	$\infty$	$\infty$
t=2	0	1	2	$\infty$	$\infty$
t=3	0	1	2	3	$\infty$
t=4	0	1	2	3	4

Figura 125 - Il problema count-to-infinity (buono notizia)

Qui ci sono tutte le decisioni che prendono i router ad un certo istante. Stiamo ipotizzando che tutti i router prendano decisioni sincrone, ma quanto questa cosa sia consistente con la realtà dipende dal protocollo.



	A	B	C	D	E	
t=4	0	1	???	2	3	4
t=5	0	3	2	3	3	4
t=6	0	3	4	3	3	4
t=7	0	5	4	5	4	4
t=8	0	5	6	5	5	6
t=9	0	7	6	7	6	6
t=?	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

Figura 126 - Il problema count-to-infinity (brutta notizia)

Supponiamo che la linea tra A e B vada nuovamente fuori servizio. Succede che all'istante  $t=5$  B vede A a costo 3, e C vede A a costo 2. Ci sembra tutto abbastanza ragionevole, ma perché D vede A a costo 3? Lui ha ricevuto il distance vector di C, che dice a B: "Se passi attraverso di me per raggiungere A paghi 2", quindi passando attraverso C per raggiungere A pago 3. Questo problema è intrinseco negli algoritmi distance vector, si può alleviare ma non rimuovere (problema count-to-infinity). Si può vedere che col passare del tempo il costo calcolato va verso infinito. Quanto tempo ci metto? Non un tempo infinito, perché si sceglie un valore convenzionale che significa infinito (1-2-3-4-5-6, è rotto, infinito). A capire che c'è un guasto ci si mette un tempo proporzionale al numero scelto per rappresentare l'infinito (per le buone notizie il tempo è proporzionale al numero di router, si arriva a distanza k in k passi). Si può scegliere un numero che è  $n+1$ , dove  $n$  è la massima distanza massima tra due router. A questo punto però si deve fare una valutazione statica a priori, di questa distanza. Questo significa che quando si collega un altro router quel valore non va più bene. È bene sceglierlo sufficientemente robusto rispetto a variazioni di topologia. Un altro esempio di cattiva notizia:

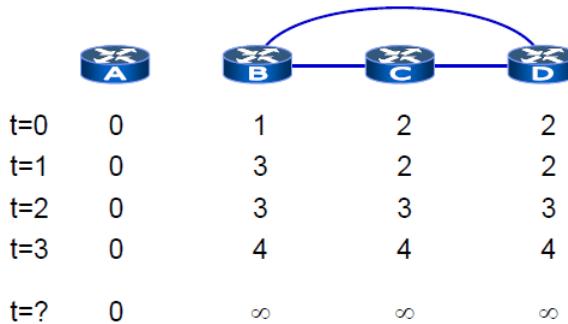


Figura 127 - Un altro esempio di cattiva notizia

Quando si attribuisce un certo valore di infinito, ad esempio 5, si deve aspettare che la distanza raggiunga 5 su tutte le macchine.

#### Efficacia in condizioni di raggiungibilità

L'algoritmo converge veramente? Non è che ad un certo punto si possono creare delle situazioni limite in cui si cicla all'infinito, senza raggiungere la stabilità? Questo non succede, ma per rendercene conto usiamo la formalizzazione di Bellman:

- vertici numerati 1, ..., n ed archi orientati;
- vertice 1 = destinazione;
- $A(i)$  insieme dei vertici j per cui c'è un arco orientato  $(i,j)$  uscente da i;
- $A(1)$  insieme vuoto (la destinazione non ha archi uscenti);
- $a_{ij} (>= 0)$  = metrica dell'arco  $(i,j)$ ;
- lunghezza di un cammino = somma delle metriche degli archi traversati.

Questo algoritmo tratta ciascuna destinazione indipendentemente da tutte le altre. Cosa dà un router quando gli arrivano i distance vector? Si fa il calcolo con riferimento ad una sola destinazione, non ci si chiede cosa succede sulle altre. Il discorso fatto su una sola destinazione quindi è di completa generalità.

Facciamo due ipotesi di lavoro:

- diciamo che i cicli sono tutti positivi. L'ipotesi è assolutamente realistica: è difficile pensare a cicli in una rete con costi nulli o negativi.
- Connettività: c'è un cammino orientato, diretto, dalla destinazione i a 1, per ogni nodo  $i=2,\dots,n$ .

È un'ipotesi un po' forte: che succede quando un collegamento si interrompe, o non c'è? Se facciamo analisi su un algoritmo di questo tipo facciamo analisi anche con riferimento al problema del count to infinity. Questa ipotesi non ci aiuta nella risoluzione di questo problema. Come si rimedia? Possiamo collegare ciascun vertice al vertice 1 con un arco dal peso idealmente infinito, o comunque molto grande (ad esempio 1000, 10000....). L'algoritmo funziona in questo modo:

$$x_j^k = \min_{j \in A(i)} \{a_{ij} + x_j^{k-1}\}, i = 2, \dots, n$$

$$x_i^k = 0$$

All'iterazione  $k$  prendo le informazioni che mi arrivano dai miei vicini all'iterazione  $k-1$ . Come mi conviene inizializzare le variabili? Col costo iniziale che pensa di dover sostenere un vertice per raggiungere la destinazione, oppure col massimo valore possibile. È ammissibile anche scegliere valori positivi qualunque, perché dobbiamo saper rispondere a variazioni di topologia, e verificare se il tutto continua a convergere anche quando tutti partono da una qualunque conoscenza della rete, anche sbagliata. Secondo Bellman e Ford, se dopo  $k$  iterazioni si verifica che  $x_i^k = x_i^{k-1}$ .

L'algoritmo termina, e le  $x_i^k$  coincidono con le soluzioni del sistema:

$$x_i^* = \min_{j \in A(i)} \{a_{ij} + x_j^*\}, i = 2, \dots, n$$

$$x_1^* = 0$$

Ci si arriva a queste condizioni di equilibrio? Bellman dice che le condizioni di equilibrio:

- vengono sempre raggiunte;
- vengono raggiunte in un numero lineare di passi;
- producono le distanze più brevi verso 1.

I cammini della soluzione possono essere più di uno. Abbiamo ancora bisogno di due definizioni preliminari. Dati due vertici  $i \neq 1$  e  $j \neq 1$  definiamo  $e_{ij}^k$  la lunghezza del cammino minimo da  $i$  a  $j$  con esattamente  $k$  archi. Questo cammino può contenere anche cicli. Inoltre definiamo  $w_{ii}^k$  la lunghezza del cammino minimo da  $i$  a  $i$  con al più  $k$  archi. Anche questo cammino potrebbe non esserci, e in quel caso il valore è  $\infty$ .

#### Lemma delle distanze

Dobbiamo dimostrarlo se vogliamo essere certi che l'algoritmo di Bellman e Ford funzioni.

$$x_i^k = \min \left[ \min_{j=2, \dots, n} (e_{ij}^k + x_j^0) \right]$$

$$\forall i = 2, \dots, n \wedge k \geq 1$$

$k \geq 1$  significa "a partire dalla prima iterazione". Questo ci dice che i valori delle distanze sono il minimo tra due componenti:

- la prima componente ("count-to-infinity") è determinata dalle condizioni iniziali potenzialmente errate (questa componente cresce al crescere di  $k$ );
- la seconda componente ("good-news") è la distanza minima tra i cammini con al più  $k$  passi (per  $k$  sufficientemente grande è proprio il cammino minimo).

Come vanno nel tempo gli argomenti di quel min? Il primo argomento cresce nel tempo, mentre il secondo decresce. Nella situazione in cui quest'ultima prevale siamo nella situazione di stabilità. Vediamo gli aspetti fondamentali della dimostrazione del lemma delle distanze.

### Dimostrazione

- ipotesi induttiva (vera per  $k$ )
 
$$\forall i = 2, \dots, n \wedge k \geq 1$$

$$x_i^k = \min \left[ \min_{j=2, \dots, n} (e_{ij}^k + x_j^0), w_{i1}^k \right]$$
- caso base, per  $k = 1$ 

$$\forall i = 2, \dots, n \quad x_i^1 = \min \left[ \min_{j=2, \dots, n} (e_{ij}^1 + x_j^0), w_{i1}^1 \right]$$

– è un passo dell'algoritmo di Bellman e Ford

$$\forall i = 2, \dots, n \quad x_i^1 = \min_{j \in A(i)} (a_{ij} + x_j^0)$$
- vogliamo dimostrare l'asserto per  $k+1$ 

$$\forall i = 2, \dots, n \quad x_i^{k+1} = \min \left[ \min_{j=2, \dots, n} (e_{ij}^{k+1} + x_j^0), w_{i1}^{k+1} \right]$$
- consideriamo separatamente le due componenti
 

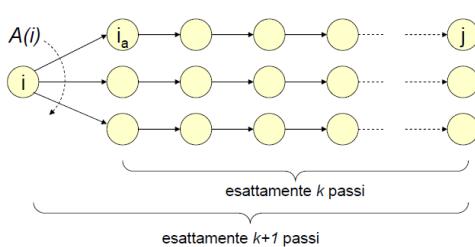
$\min_{j=2, \dots, n} (e_{ij}^{k+1} + x_j^0)$   
componente "count-to-infinity"

$w_{i1}^{k+1}$   
componente "good-news"

Figura 128 - Dimostrazione per induzione

Figura 129 - Caso induttivo

$$\min_{j=2, \dots, n} (e_{ij}^{k+1} + x_j^0) = \min_{i_a \in A(i)} \left( a_{ii_a} + \min_{j=2, \dots, n} (e_{i_a j}^k + x_j^0) \right)$$



$$w_{i1}^{k+1} = \min_{i_a \in A(i)} (a_{ii_a} + w_{i_a 1}^k)$$

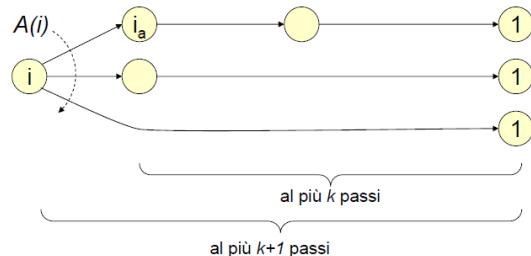


Figura 130 - Componente good news

$$\begin{aligned} x_i^{k+1} &= \min \left[ \min_{j=2, \dots, n} (e_{ij}^{k+1} + x_j^0), w_{i1}^{k+1} \right] \\ &= \min \left[ \min_{i_a \in A(i)} \left( a_{ii_a} + \min_{j=2, \dots, n} (e_{i_a j}^k + x_j^0) \right), \min_{i_a \in A(i)} (a_{ii_a} + w_{i_a 1}^k) \right] \\ &= \min_{i_a \in A(i)} \left[ \min \left( a_{ii_a} + \min_{j=2, \dots, n} (e_{i_a j}^k + x_j^0), a_{ii_a} + w_{i_a 1}^k \right) \right] \\ &= \min_{i_a \in A(i)} \left[ a_{ii_a} + \underbrace{\min_{j=2, \dots, n} (e_{i_a j}^k + x_j^0)}_{x_{i_a}^k \text{ (per ipotesi induttiva)}} \right], w_{i_a 1}^k \end{aligned}$$

- abbiamo dimostrato che il lemma delle distanze

$$x_i^{k+1} = \min \left[ \min_{j=2, \dots, n} (e_{ij}^{k+1} + x_j^0), w_{i1}^{k+1} \right]$$

- è vero se è vera l'equazione seguente

$$x_i^{k+1} = \min_{i_a \in A(i)} (a_{ii_a} + x_{i_a}^k)$$

- che è vera in quanto coincide con la definizione dell'algoritmo di Bellman e Ford

Figura 131 – Sostituzione e conclusione

*Conclusioni dell'analisi*

- ogni cammino minimo da  $i$  a 1 ha al più  $n-1$  archi
- dimostrazione:
  - un cammino da  $i \neq 1$  ad 1 con più di  $n-1$  archi contiene almeno un ciclo
  - visto che il ciclo ha lunghezza positiva può essere rimosso dando luogo ad un cammino più corto

Figura 132 - Proprietà (a)

- se  $x_i^0 \geq x_i^*$  per ogni  $i$ , allora l'algoritmo termina in al più  $m^*+1$  iterazioni dove
 
$$m^* = \max_{i=2,\dots,n} m_i \leq n-1$$
 e  $m_i$  è il massimo numero di archi in un cammino minimo da  $i$  a 1
- consideriamo solo il caso in cui per  $i \neq 1$  abbiamo  $x_i^0 = \infty$
- allora dal lemma delle distanze abbiamo che per ogni  $i \neq 1$  e per ogni  $k$ ,  $x_i^k$  è la distanza più breve da 1 a 1 con al più  $k$  archi
- quindi  $x_i^k = x_i^*$  per ogni  $i$  e  $k \geq m^*$

Figura 134 - Proprietà (c)

- per ogni insieme di condizioni iniziali l'algoritmo termina dopo un numero finito  $k$  di iterazioni con  $x_i^k$  uguale alla distanza più breve  $x_i^*$
- dimostrazione:
  - dato che tutti i cicli hanno lunghezza positiva, per  $k$  che tende a infinito
    - se  $i \neq 1$  e  $j \neq 1$   $e_{ij}^k \rightarrow \infty$
    - per  $k \geq n-1$   $w_{i1}^k = x_i^* < \infty$
  - dal lemma delle distanze, per  $k$  sufficientemente grande abbiamo  $x_i^k = x_i^*$

Figura 133 - Proprietà (b)

- le lunghezze  $x_i^*$  dei cammini minimi verso 1 sono l'unica soluzione delle equazioni di Bellman
- supponiamo per assurdo che esistano due soluzioni distinte delle equazioni di Bellman
- se lanciamo l'algoritmo di Bellman e Ford usando come valori iniziali una soluzione delle equazioni di Bellmann allora l'algoritmo termina in un solo passo
- la proprietà (b) impone però che l'esito dell'algoritmo di Bellman e Ford siano le distanze minime
- dunque la soluzione è unica e coincide con le distanze minime

Figura 135 - Proprietà (d)

*Efficienza degli algoritmi di distance vector*

Ci sono vari modi per valutare l'efficienza di un algoritmo di instradamento:

- contare il numero di passi che l'algoritmo impiega a convergere, immaginando che gli eventi siano sincronizzati;
- contare il numero di pacchetti che vengono scambiati sulla rete per convergere;
- valutare il tempo impiegato da ciascun router per portare la propria tabella ad una situazione stabile.

Supponiamo che in una rete ci siano  $n$  nodi ed  $m$  link e supponiamo che la metrica sia relativa solo al numero di hop. Tra i nodi più lontani ci sono al massimo  $n-1$  link (lunghezza massima del cammino più lungo):

- una buona notizia si propaga su tutta la rete in al più  $n-1$  passi;
- se attribuiamo a infinito il valore  $n$  allora una cattiva notizia si propaga su tutta la rete in al più  $n-1$  passi.

In ciascuno dei passi ogni router scandisce al più  $m$  distance vector (numero massimo di linee intorno a un router): ogni tabella ha al più  $n$  righe (entries). In ciascuno dei passi ogni router spende tempo  $O(nm)$ . Dato che la convergenza richiede nel caso peggiore  $O(n)$  passi, ogni router calcola per un tempo  $O(n^2m)$ .

## Algoritmi Link-State-Packet

In questi algoritmi ciascun IS ha una mappa completa della rete, perciò i router collaborano non per la costruzione della tabella di instradamento, ma per la costruzione di una mappa completa della rete, replicata su ogni router. Ogni IS calcola sulla mappa un instradamento ottimale, cioè cammini di costo minimo verso tutte le destinazioni (usando l'algoritmo di Dijkstra). La tabella di instradamento si ottiene considerando il primo hop dei cammini minimi. La mappa della rete viene costruita a partire dai link state packet (lsp):

- un lsp contiene informazioni sui nodi e sui link adiacenti ad uno specifico is;
- i lsp sono trasmessi in selective flooding da ogni is a tutti gli altri is della rete.

Esempio:

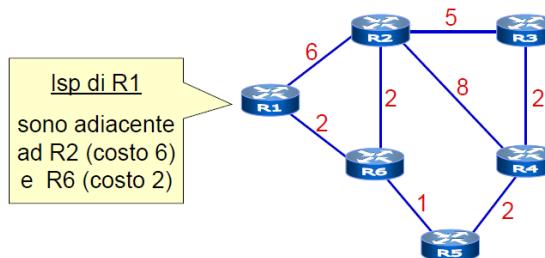


Figura 136 - Mappa della rete

Un esempio di lsp database (su ogni IS):

R1: sono adiacente a R2/6 R6/2;

R2: sono adiacente a R1/6 R3/5 R4/8 R6/2;

R3: sono adiacente a R2/5 R4/2;

R4: sono adiacente a R2/8 R3/2 R5/2;

R5: sono adiacente a R4/2 R6/1;

R6: sono adiacente a R1/2 R2/2 R5/1.

Un LSP database deve essere identico su tutti gli IS; la cosa non è marginale, perché se si ha un fault da qualche parte i nuovi lsp si devono propagare rapidamente nella rete, altrimenti si avrebbero informazioni disallineate e quindi situazioni di instabilità nella rete. Rispetto agli algoritmi distance vector negli algoritmi LSP la collaborazione tra gli IS è tesa a mantenere aggiornata una mappa della rete (non a calcolare direttamente le tabelle di instradamento). Questi algoritmi possono gestire reti anche con 10.000 nodi, e convergono rapidamente.

Nella rete precedente, se volessimo calcolare i cammini minimi a partire da R1, avremmo:

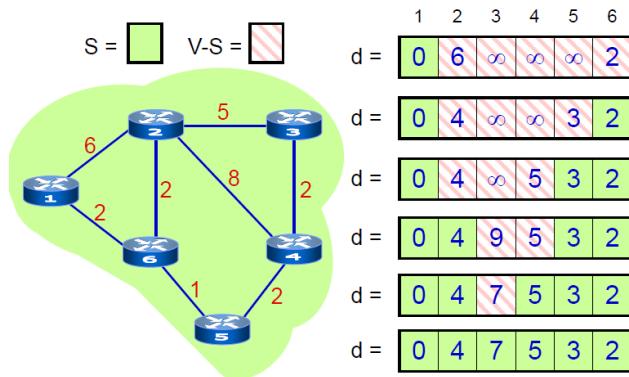


Figura 137 - Esempio di algoritmo di Dijkstra

Se  $d[v] = m(v)$  per ogni nodo  $v$  di  $S$ , allora quando  $i$  viene aggiunto ad  $S$ , si ha  $d[i] = m(i)$ .

Dimostrazione per assurdo:

- supponiamo che  $d[i] > m(i)$ ;
- sia  $p$  il cammino minimo da  $1$  ad  $i$ ; necessariamente  $|p| = m(i) < d[i]$ .
- sia  $j$  l'ultimo nodo di  $p$  appartenente ad  $S$  e  $k$  il nodo seguente:  $k$  diverso da  $i$  altrimenti l'algoritmo avrebbe computato  $d[i] = |p| = m(i)$ .

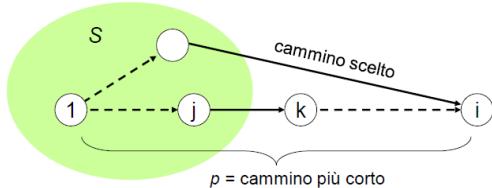


Figura 138 - Dimostrazione per assurdo

FINIRE SLIDE

## Protocolli di routing e la rete internet

Parliamo del livello 3 in termini di protocolli di rete e di routing. Un protocollo di rete:

- definisce uno schema di indirizzamento;
- determina il formato del pacchetto di rete;
- offre un servizio di consegna ai protocolli del livello di trasporto;
- può far uso di tabelle di instradamento.

Il protocollo di routing, rispetto al protocollo di rete sta tipicamente sullo stesso livello dal punto di vista delle funzioni svolte, ma possiamo guardare ai protocolli di routing indipendentemente dai protocolli di rete. In particolare, un protocollo di routing:

- si ispira ad un algoritmo di routing (esempio: distance vector, link state packet);
- fa riferimento ad uno specifico protocollo di rete;
- si occupa dell'aggiornamento delle tabelle di instradamento

Stiamo scindendo in maniera precisa i due ruoli: i protocolli di livello 3 (rete) e i protocolli di supporto al livello 3 (routing). I protocolli di rete contemporaneamente attivi sulla stessa macchina sono più d'uno (ad esempio IPX e IP possono convivere sulla stessa macchina, ma anche IPv4 e IPv6). Quindi su una macchina non c'è solo un livello 3, ma ce ne possono essere più d'uno. Questa è una notizia che in qualche modo già sapevamo: abbiamo detto in passato che gli LLC-SAP servono a identificare diverse pile protocollari di livello più alto, e questo è segno evidente che sulla stessa macchina ci possono essere più pile protocollari. Queste pile protocollari se sono distinte sono distinte anche per il livello 3. Possono essere poi attivi sulla rete più protocolli di routing, e i diversi protocolli di routing possono essere riferiti anche allo stesso protocollo di rete. Quando sugli end system e sugli intermediate system sono ospitati diversi protocollo di rete (e quindi poi anche di protocolli di livello superiore) si parla di macchine dual stack (due o più pile protocollari). Diverse sono le situazioni in cui un end system è dual stack e quella in cui un intermediate system è dual stack. Quando un end system è dual stack le applicazioni devono decidere quale specifica pila protocollare usare per una particolare istanza applicativa, e in base alla specifica protocollare possono essere portate ad utilizzare diverse librerie di socket. Tipicamente un'applicazione sceglie una pila protocollare, ma non è detto: si può scegliere una pila protocollare per una cosa e un'altra pila per un'altra. Sull'intermediate system non ci sono applicazioni (sempre che non ci vengano messe apposta), quindi per un end system essere dual stack significa essere in grado di instradare pacchetti diversi e che sono relativi a diverse pile protocollari. Il paradigma a cui ci si riferisce in presenza di più stack su un certo intermediate system è quello delle “navi nella nebbia” (oppure “navi nella notte”, si usano entrambi le espressioni):

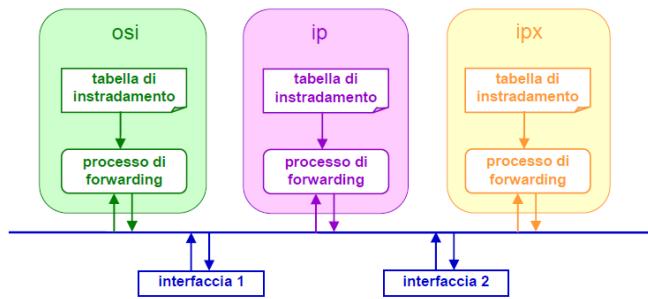


Figura 139 - Architettura router dual

I vari protocolli presenti sulla stessa macchina convivono ignorandosi completamente. Quando un pacchetto arriva su una certa interfaccia, in funzione di quello che c'è scritto nel pacchetto (leggi: valore di LLC-SAP) il pacchetto viene smistato al protocollo di livello superiore che se ne deve occupare. Il protocollo di livello superiore che se ne deve occupare ha associato sull'intermediate system un processo di forwarding, e a questo punto attingendo a quel protocollo il pacchetto viene spedito all'interfaccia corretta. **Ogni protocollo di routing fa riferimento ad un solo protocollo di rete, oppure diversi protocolli di routing possono far riferimento al medesimo protocollo di rete. In questa seconda situazione ogni protocollo di rete gestisce i suoi dati indipendentemente dagli altri protocolli.** Quali saranno questi dati? Se usiamo protocolli che nascono da algoritmi distance vector saranno dei distance vector, se nascono da algoritmi link state packet saranno dei LSP database, ma anche tabelle di instradamento (intese come tabelle che vengono costruite da un intermediate system per dire come deve essere fatto l'intradamento, e diventano tabelle di instradamento quando vengono rese operative). Le tabelle di routing dei vari protocolli di routing assieme vengono usate per determinare la tabella di forwarding vera e propria, e naturalmente quando ci sono diverse fonti da cui apprendere informazioni di questo genere si può aver bisogno di stabilire quale sia la diversa priorità tra di loro. Le priorità delle diverse tabelle, oltre ad avere dei valori di default, possono essere alterate dall'amministratore.

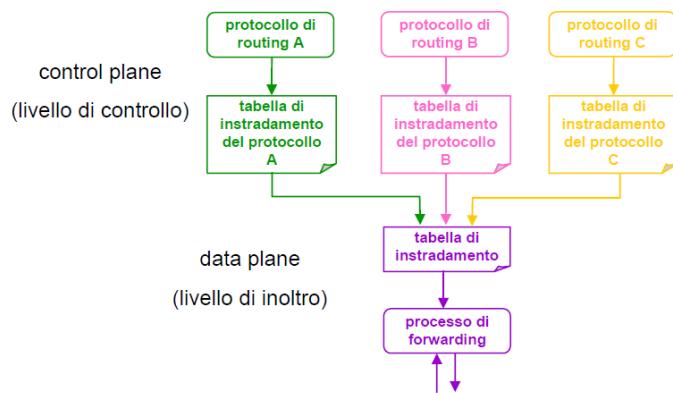


Figura 140 - Livello di controllo e livello di inoltro

c'è quello che succede tra protocolli di routing e protocolli di rete su un intermediate system. Sappiamo che uno specifico protocollo di routing ha una tabella di instradamento (con prefissi, netmask...). C'è poi

un'entità, il processo di forwarding, che quando arriva un pacchetto ne fa l'inoltro, in base alla tabella di instradamento. Queste ultime due parti si chiamano, in gergo, **data plane** (terminologia fondamentale per le reti). Tutto il resto si chiama **control plane**, e si occupa di costruire la tabella di instradamento. Questo control plane può avere contemporaneamente attivi diversi protocolli di routing (in figura abbiamo i protocolli A, B, C tutti e tre contemporaneamente attivi sullo stesso intermediate system), e tutti i 3 lavorano per costruire la tabella di instradamento del data plane. Ci sono quindi delle tabelle di instradamento che sembrano essere diverse dalla tabella di instradamento del data plane. Che cos'è quest'altra tabella di instradamento? Pensiamo agli algoritmi distance vector. Dentro le tabelle di instradamento di questi algoritmi c'è scritto anche il costo per raggiungere una destinazione. Questa metrica, è di interesse per il data plane? No. Dal punto di vista del data plane il costo del raggiungimento è irrilevante, possiamo anche scrivere il costo, ma non ha importanza. Ci sono quindi delle informazioni delle tabelle di instradamento degli algoritmi di routing che non sono di nessun interesse per il data plane, perché sono specifiche del control plane. Come si passa dalle tabelle di instradamento dei vari protocolli di routing a quella del forwarding? Quelle sono tabelle in un certo senso virtuali, non c'è nessun pacchetto che verrà instradato in base al control plane, ma tutti i pacchetti verranno instradati in base alla tabella di instradamento del data plane.

### Comunicazioni tra protocolli di routing

I protocolli di routing possono comunicare fra loro, e l'amministratore di rete può governare i passaggi di rotte da un protocollo all'altro. In questo caso si parla tipicamente di iniezione di rotte tra un protocollo e l'altro. In qualche modo i protocolli di instradamento di un intermediate system comunicano tra loro quasi senza accorgersene, perché contribuiscono in vario modo a costruire la stessa tabella di instradamento, ma ci può essere una comunicazione che non è soltanto implicita, come quella che abbiamo appena detto, ma che può essere anche esplicita: il protocollo A dice al protocollo B "Queste sono le rotte che io ho calcolato, usale nell'ambito del tuo lavoro".

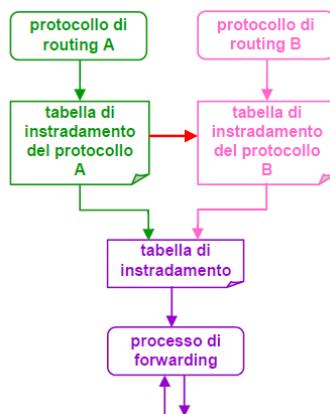
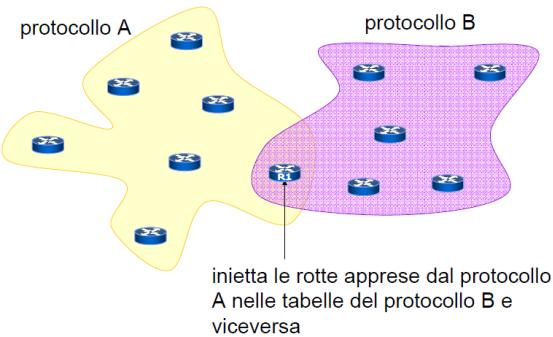


Figura 141 - Comunicazioni tra protocolli di routing

Le rotte inviate dal protocollo A al protocollo B (freccia rossa) vengono viste da quest'ultimo come delle rotte statiche (*route add...*); la possibilità di fare routing statico è sempre presente, possiamo pensare che il routing statico sia uno dei protocolli attivi su un intermediate system. Perché un certo protocollo dovrebbe iniettare delle rotte da lui lavorate su un altro protocollo? Questo aspetto ci sarà chiaro tra poco.



*Figura 142 - Comunicazione tra protocolli di routing*

Abbiamo una rete governata in parte dal protocollo A e in parte dal protocollo B. Sul router R1 sono attivi sia il protocollo A che il protocollo B, quindi R1 può iniettare le rotte apprese dal protocollo A nelle tabelle del protocollo B e viceversa. La figura già di per sé comincia a motivare quanto detto in precedenza; guardiamola con un atteggiamento concreto: abbiamo una rete che parte governata in parte dal protocollo A e in parte dal protocollo B. Immaginiamo una migrazione fra protocolli di routing. Siamo un ISP che ha una grossa rete su cui gira il protocollo A da molti anni, poi vogliamo fare una scelta più moderna, vogliamo passare al protocollo B (perché è più bello, più veloce, converge più rapidamente, più facile da gestire....), e quindi iniziamo a disegnare il protocollo B su un sottoinsieme delle macchine. Potrebbe valere la pena di passare dal protocollo A al protocollo B, in un certo istante, spegnendo A e accendendo B? Questo è un atteggiamento ingenuo: se abbiamo una rete con 10.000 intermediate system il fatto che si possa avere un D-day in cui si spegne tutto A e si accende tutto B è assolutamente impossibile, tra l'altro in quel momento si potrebbero avere tanti di quei problemi non considerati in fase di progettazione da mandare fuori uso l'intera infrastruttura per chissà quanto tempo. Si fa quindi una migrazione progressiva, facendo magari in modo che le porzioni di rete che stanno su diversi protocolli siano in qualche modo disgiunte, per evitare interazioni. Le interazioni però devono esserci, perché le macchine di una porzione di rete devono essere in grado di raggiungere le macchine dell'altra. Chi instrada questi pacchetti? Si prende un router e si disegnano su di esso ambedue i protocolli, così R1 fa da "ponte" fra le due porzioni della rete.

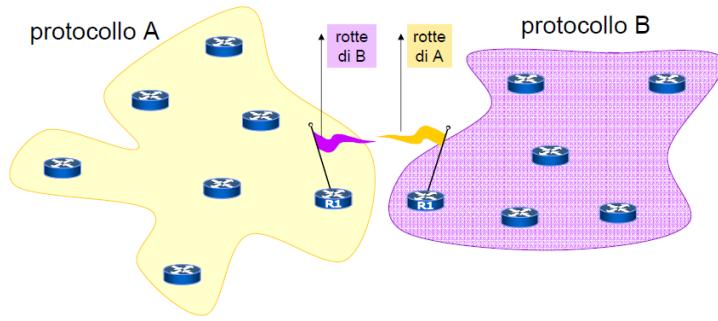


Figura 143 - Comunicazione fra protocolli di routing

In questo caso si dice non soltanto che R1 inietta, ma anche che R1 redistribuisce le rotte apprese, in particolare, nella figura R1 redistribuisce:

- nel protocollo A le rotte apprese dal protocollo B;
- nel protocollo B le rotte apprese dal protocollo A.

Iniettare le rotte ha un senso, e ha un senso farlo anche preservando le informazioni sulle metriche, che possono essere utilizzate dall'altro protocollo per fare i suoi calcoli per l'instradamento. **Tutto ciò dal punto di vista operativo non è banalissimo, e si possono avere anche delle perdite di ottimalità: il cammino scelto per raggiungere una destinazione, se si partiziona in questo modo la rete, non è necessariamente un cammino minimo.**

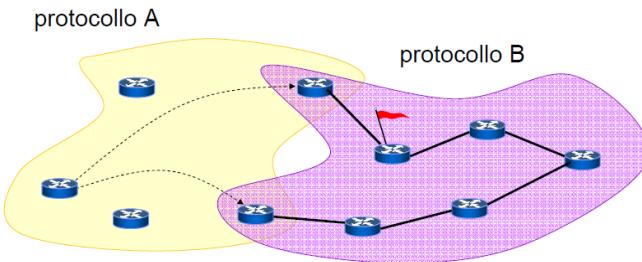


Figura 144 - Perdita di ottimalità

Prendiamo ad esempio un pacchetto che parte dal router contrassegnato; il traffico può percorrere due rotte, e nessuna delle due è necessariamente quella ottimale per raggiungere la destinazione. Potremmo parlare anche di perdita di connettività: in presenza di un fault da una parte della rete, in presenza di un congegno che non funziona proprio benissimo, possiamo arrivare a dire erroneamente che una destinazione non sia raggiungibile, e viceversa è raggiungibile facendo un opportuno cammino diverso nella zona gestita dal protocollo di partenza. Questa perdita di ottimalità può essere significativa se non abbiamo a che fare con solo due zone, ma con tante zone, il partizionamento può essere ancor più "spericolato", ancor più completo:

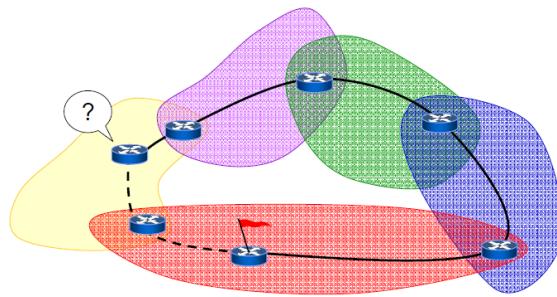


Figura 145 - Perdita di ottimalità

Parlare di partizione può essere improprio, perché ci sono dei router che fanno da perno, e sono gli intermediate system nei quali avviene l'iniezione e la redistribuzione, quindi è più corretto parlare di copertura. Ci possono inoltre essere dei problemi amministrativi: se diverse amministrazioni volessero scambiarsi pacchetti in questo modo dovrebbero cogestire le macchine dual stack.

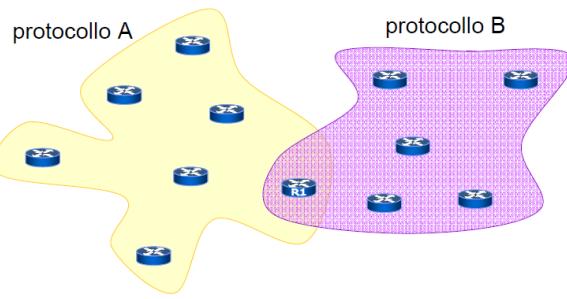


Figura 146 - Problemi amministrativi

Questo è il motivo per cui se ci sono contemporaneamente più amministrazioni con diversi protocolli si preferisce disaccoppiare i due domini amministrativi in maniera più efficace:

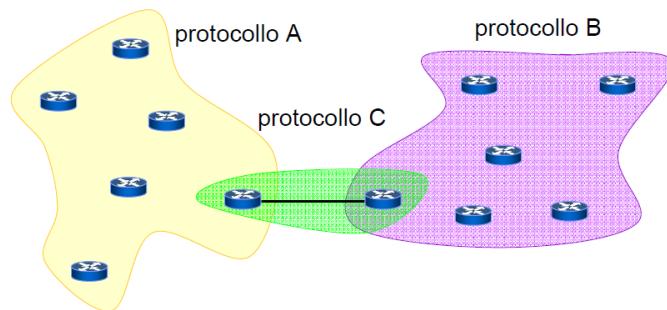


Figura 147 - Routing gerarchico

I due router di frontiera sono gestiti ciascuno da un'amministrazione, e la cogestione della rete si limita in questo caso al collegamento tra i due router.

## Protocolli IGP ed EGP

All'interno di un dominio amministrativo vengono utilizzati tipicamente dei protocolli chiamati IGP (*Interior Gateway Protocol*), e i protocolli usati per trasferire le rotte tra due domini amministrativi vengono chiamati EGP, *Exterior Gateway Protocol* (attualmente si usa il BGP, *Border Gateway Protocol*).

### EGP

Gli obiettivi sono:

- adottare un solo protocollo di connessione tra domini amministrativi diversi;
- prediligere le rotte che attraversano meno domini amministrativi;
- consentire l'implementazione di politiche commerciali: non tutte le rotte possono attraversare un dominio amministrativo e le rotte che attraversano uno specifico dominio amministrativo possono essere preferite.
- preseguire la trasparenza rispetto agli IGP.

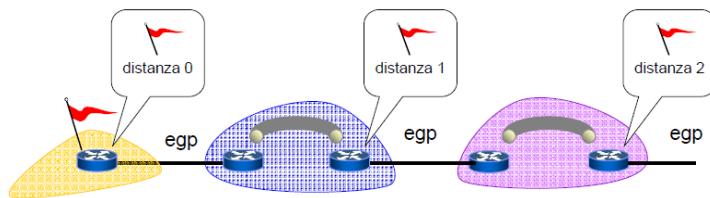


Figura 148 - EGP

Le informazioni apprese tramite EGP sono ridistribuite in IGP: qualora si vogliano rendere raggiungibili dal proprio dominio amministrativo. Inoltre sono trasferite agli altri router di frontiera dello stesso dominio amministrativo, generalmente tramite connessioni TCP.

### IGP

#### RIP

Il primo algoritmo IGP che vediamo è RIP, che è un distance vector sostanzialmente sincrono, lento a convergere (perché valgono tutte le cose dette finora sugli algoritmi distance vector). Il numero di hop convenzionalmente considerato come infinito è 15 (posso quindi usare questo protocollo in reti particolarmente piccole). I distance vector sono spediti broadcast (su tutte le LAN collegate ad un certo router, in RIPv1) o multicast (a tutti i router della rete, in RIPv2) ogni 30 secondi.

I pacchetti sono estremamente semplici:

- comando (richiesta, risposta);
- versione (RIPv1 o RIPv2);
- lista di subnet descritte da:
  - IP address;

- **subnet mask (solo RIPv2).**

Abbiamo detto che i distance vector vengono inviati ogni 30 secondi; a questi 30 secondi si aggiunge un piccolo offset. Questo perché nei primi tempi di Internet si osservò che se i distance vector venivano spediti stabilmente ogni 30 secondi si tendeva ad avere un'autosincronizzazione della stessa rete. Si conviene è che un prefisso su cui non arrivino informazioni aggiornate entro un certo tempo viene considerato non più raggiungibile.

#### IGRP

È un IGP disponibile soltanto su router CISCO. È un distance vector con metriche sofisticate (combinazione di ritardo, banda, affidabilità, lunghezza massima del pacchetto, carico) che permette di mantenere attive più rotte per suddividere il carico su più linee (*multipath routing*).

#### OSPF

*Open Shortest Path First.* È il più diffuso IGP per TCP/IP, è basato su link state packet (la rete viene rappresentata come un grafo) e dà la possibilità di gestire più criteri di ottimizzazione (grafi diversi in funzione del type of service: ottimizzazione del ritardo, del throughput, dell'affidabilità). Un'area di routing OSPF è suddivisa in aree, che sono porzioni diverse della stessa rete. In particolare, c'è un'area di backbone e delle aree che non sono di backbone ("normali") e per spostarsi da un'area ad un'altra bisogna sempre passare in un'area di backbone:

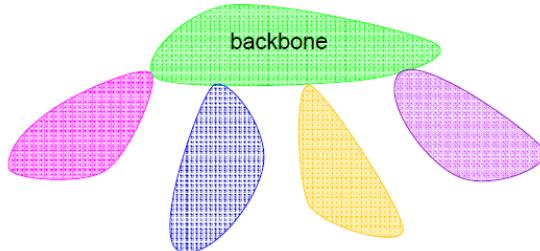


Figura 149 - Aree in OSPF

Questa suddivisione in aree dà luogo ad una diversificazione dei router, che possono essere:

- *internal*: tutte le interfacce sono interne alla stessa area non di backbone;
- *backbone*: ha almeno un'interfaccia sul backbone;
- *area border*: ha almeno due interfacce su due diverse aree;
- *as boundary*: ha almeno un'interfaccia verso l'esterno (verso altri domini amministrativi).

Queste suddivisioni non sono mutuamente esclusive,

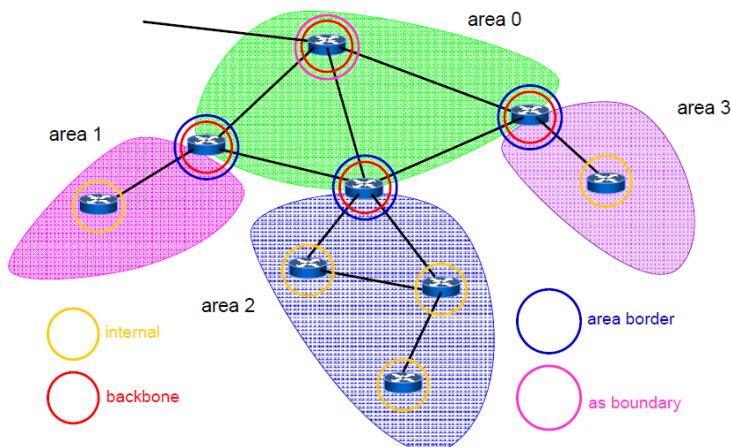


Figura 150 - ospf: tipi di router e aree

Si fa la distribuzione in aree tipicamente per motivi di scalabilità. Ricapitolando: stiamo guardando la rete di un singolo provider, che si rapporta con le reti di altri provider con i router *as boundary*. Stiamo guardando un singolo provider che utilizza un singolo protocollo di routing. In questo caso il provider sta utilizzando per tutti i propri apparati OSPF (la situazione potrebbe benissimo non essere questa, possiamo avere RIP per una parte della rete e OSPF per un'altra, per esempio). Il provider ha scelto, nel suo utilizzo di OSPF di suddividere la rete in aree. Questo non è obbligatorio, potevamo prendere tutti gli apparati e raggrupparli in un'unica area. Se l'area è una sola questa coincide con la rete di backbone (area 0); se l'area è una sola i LSP che vengono scambiati descrivono la topologia della rete (dell'area).[49:00]. Questo però pone problemi di scalabilità: con una sola area l'algoritmo che scambia i LSP deve fare un sacco di lavoro. Suddividendo in aree, in un'area ogni apparato usa i LSP (relativi alle singole aree, a tutti i link interni) per costruirsi una mappa delle aree.

Quando stiamo dentro l'area 1 però come facciamo a raggiungere tutte le altre destinazioni? Ci pensa il router di bordo area, che fa una specie di riassunto, in modo tale che ogni router veda le destinazioni che stanno dall'altra parte siano direttamente connesse al router di bordo area, con un'enorme semplificazione del grafo. Se abbiamo una rete veramente grande quindi è il caso di tagliare la rete in pezzi e fare in modo che il routing sia opportunamente partizionato tra le diverse aree.

## Bridge: calcolo dello spanning tree

In estrema sintesi: lo spanning tree algorithm (STA):

- è descritto nello standard ieee 802.1D;
- è un algoritmo distribuito che calcola un albero ricoprente della rete: al termine della computazione alcuni bridge mettono alcune porte in “blocking state” (le porte in blocking non vengono utilizzate);
- un bridge viene eletto radice dell’albero: tale bridge viene chiamato root bridge;
- l’amministratore può influenzare l’esito del calcolo modificando opportuni parametri di configurazione.

Il motivo per cui si calcola uno spanning tree in una rete con bridge è perché la presenza di cicli renderebbe impossibile l’operazione di filtering dei bridge. In questa rete:

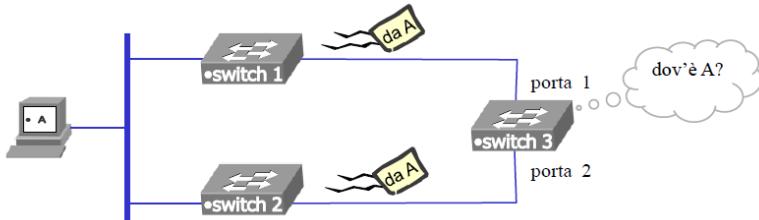


Figura 151 - Esempio di rete

Ci sono 3 switch. Un pacchetto emesso dalla macchina A viene inviato sia sullo switch 1 che sullo switch 2, e switch 3 non sa se A va visto sulla porta 1 oppure sulla porta 2. La ridondanza però talvolta è necessaria, quindi serve un algoritmo che calcoli l’albero ricoprente della rete. Grazie allo spanning tree si può effettuare il filtering, e si può conservare la ridondanza offerta dai cicli. Un algoritmo per il calcolo dello spanning tree deve garantire:

- *robustezza*: l’albero ricoprente deve essere computato in maniera distribuita;
- *efficacia*: la formazione di cicli deve essere esclusa sia in condizioni stazionarie che transitorie;
- *efficienza*: l’algoritmo deve fare un uso accorto delle risorse:
  - *bandwidth*: pacchetti scambiati dai bridge (BPDU) devono comportare un limitato consumo di banda;
  - *tempo*: la computazione dell’albero deve essere più veloce possibile per limitare il disservizio: tipicamente 30-40 secondi sono sufficienti allo sta per convergere;
- *flessibilità*: l’amministratore può assegnare priorità a ciascun bridge e a ciascuna porta per influire sull’output dell’algoritmo;
- *facilità d’uso*: l’algoritmo deve essere in grado di funzionare correttamente anche in assenza di configurazione dell’amministratore.

Lo STA riceve in ingresso:

1. una topologia costituita da:
  - un insieme di LAN (domini di collisione);
  - un insieme di bridge le cui porte afferiscono alle LAN;
2. un identificatore (detto bridge-id) distinto per ogni bridge della rete: un id più basso indica un bridge preferibile;
3. un identificatore (detto port-id) per ogni porta di ogni bridge:
  - unico all'interno del bridge;
  - un id più basso indica una porta preferibile;
4. un valore (detto costo) per ogni LAN: il costo viene effettivamente associato alle porte che afferiscono alla LAN e convenzionalmente sommato in ingresso

e produce in output: un insieme di porte che, una volta messe in blocking, garantiscano:

- che ci sia piena connettività (esista un cammino tra ogni coppia di LAN);
- che non ci siano cicli (il cammino tra ogni coppia di LAN sia unico).

È desiderabile che il costo dei cammini sia contenuto; la minimalità non può essere garantita, quindi si minimizza il costo dei soli cammini tra la radice dell'albero ricoprente e qualsiasi altro bridge.

Lo scopo del gioco è capire cosa c'entri questo algoritmo con tutto ciò che ci siamo detti finora. Come sono fatti i bridge-id e i port-id?

Il bridge-id è costituito dalla concatenazione di due parti:

- 2 byte di "priorità" scelti dall'amministratore. (il default è spesso: 80-00);
- 6 byte corrispondenti all'indirizzo mac della prima porta del bridge (per esempio: 23-efco-4b-93-ao): è dunque lungo 8 byte (per esempio: 80-00/23-ef-co-4b-93-ao).

Anche la port-id è costituita dalla concatenazione di due parti:

- un byte di "priorità" scelto dall'amministratore (il default è spesso: 80);
- un byte corrispondente al numero della porta sul bridge (per esempio: oa per la decima porta): è dunque lunga 2 byte (per esempio: 80/oa).

I costi associati alle LAN possono essere cambiati dall'amministratore, e hanno dei default inversamente proporzionali alla velocità di cifra della tecnologia utilizzata:

- costo 100 per una LAN a 10 Mb/s;
- costo 10 per una LAN a 100 Mb/s;
- costo 1 per una LAN a 1 Gb/s.

Lo sta funziona in questo modo:

1. Elezione del root bridge: un singolo bridge è selezionato per essere la radice dell'albero.

2. Identificazione della root port su ogni bridge: ogni bridge diverso dal root bridge seleziona una delle sue porte come quella “più conveniente” per connettersi al root bridge.
3. Determinazione delle designated ports: per ogni LAN una porta di un bridge è scelta come quella che conserverà la LAN allo spanning tree.
4. Blocco di porte ridondanti: le porte inutilizzate (non root ports e non designated ports) sono messe in blocking.

Per fare questa operazioni si scambiano dei pacchetti, che si chiamano BPDU; questi pacchetti viaggiano a bordo di pacchetti LLC con DSAP = SSAP = ox42, che a loro volta sono incapsulati in pacchetti MAC con sorgente l'indirizzo della porta mittente e destinazione multicast 01:80:c2:00:00:00. Sono previsti due tipi di BPDU:

- la configuration BPDU contiene tutte le informazioni necessarie per lo spanning tree algorithm;
- la topology change BPDU non contiene nessun dato: serve solo a segnalare che un cambio di topologia è in atto e ad abbassare i timer del protocollo.

Nelle configuration BPDU ci troviamo:

- root-bridge-identifier (l'attuale root dello spanning tree);
- root-path-cost (il costo del cammino verso il root bridge);
- bridge-identifier (l'id del bridge che invia questa BPDU);
- port-identifier (la porta da cui è uscita questa BPDU).

root-bridge-identifier	80-00/23-ef-00-a1-32-4d
root-path-cost	310
bridge-identifier	80-00/2d-12-d4-23-8e-5f
port-identifier	80/06

Figura 152 - Esempio di configurazione bpdu

## Fase 1

1. Ogni bridge all'inizio invia una configuration BPDU nella quale specifica il proprio bridge-id come root-identifier.
2. Quando un bridge riceve una configuration BPDU con un valore più basso di bridge-id: smette di produrre configuration BPDU con il suo bridge-id come root-identifier e propaga la nuova configuration BPDU su tutte le porte.
3. Il root bridge è quello che continuerà a produrre configuration BPDU con il suo bridge-id nel campo root-identifier.
4. Quando una configuration BPDU è prodotta dal root bridge il suo campo root-path-cost è posto a zero.
5. Quando una configuration BPDU è inoltrata da un bridge che non è il root bridge, i suoi campi sono aggiornati come segue:

- il root-path-cost è incrementato con il costo della porta del bridge che riceve la configuration BPDU;
- il bridge-identifier è sostituito con il bridge-id del bridge corrente;
- la port-identifier è sostituita con l'id della porta da cui la BPDU sarà inviata.

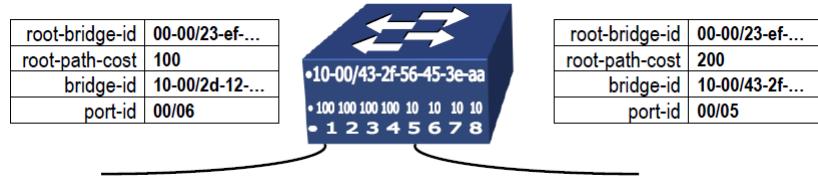


Figura 153 - Bridge in fase 1

## Fase 2

Abbiamo il root bridge che invia BPDU, sappiamo come sono fatte le BPDU, adesso dobbiamo identificare la root port. La root port è quella che riceve le configuration BPDU tali che:

1. il root-path-cost della BPDU sommato al costo della porta ricevente è il più basso (la cosa più semplice);
2. a parità di costo scelgo il bridge-identifier specificato nella BPDU più basso;
3. a parità di bridge-identifier scelgo la port-identifier specificata nella BPDU più bassa;
4. a parità di port-identifier scelgo la port-identifier della porta ricevente più bassa.

## Fase 3

Per ogni LAN una porta di un bridge è scelta come designated port in base alle configuration BPDU che sono inviate nella LAN da quella porta. La designated port è quella che invia le configuration BPDU con:

1. root-path-cost più basso;
2. bridge-identifier più basso;
3. port-identifier più basso.

La LAN in realtà non sceglie, sono i bridge che gli stanno attorno a farlo.

## Fase 4

Tutte le porte che non sono root-ports o designated-ports sono poste in blocco. Tutte le root-ports e designated-ports sono poste in stato di forwarding.

### Esempio di utilizzo dell'algoritmo

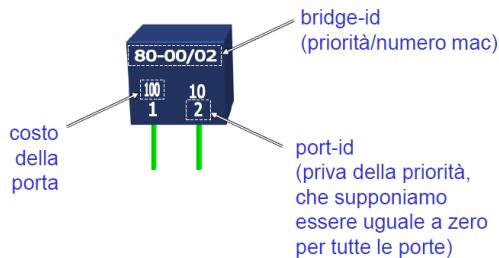


Figura 154 - Notazione utilizzata

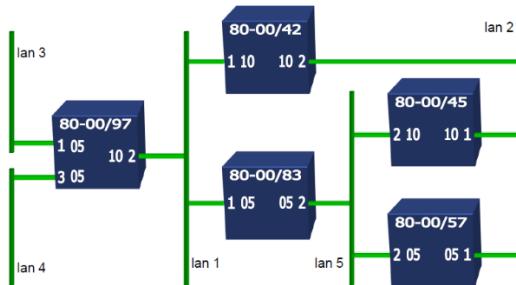


Figura 155 - Notazione utilizzata

Fase 1: chi diventa il root bridge? Le priorità sono le stesse, quindi vince quello col MAC address più basso.

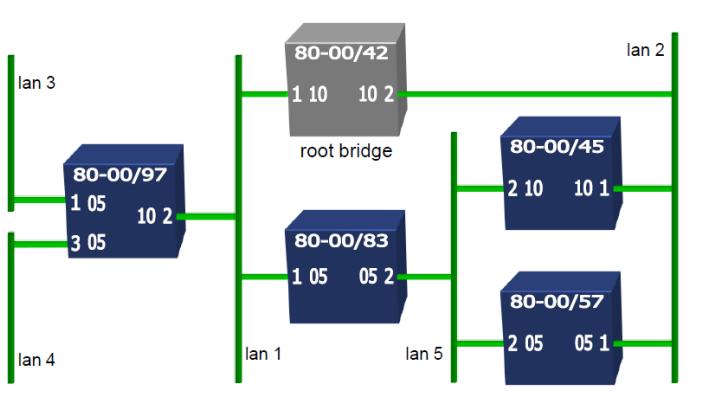


Figura 156 - Fase 1: elezione del root bridge

Adesso solo il root bridge manda BPDU, e gli altri router si limitano a reinviare quelle BPDU.

Fase 2: ogni bridge diverso dal root bridge sceglie la root port, la porta attraverso cui il root bridge si raggiunge più convenientemente.

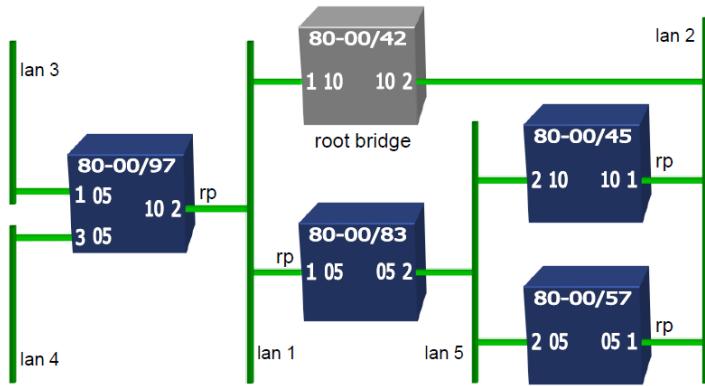


Figura 157 - Fase 2: identificazione delle root port

Fase 3: determinazione delle designated port (quelle che conserveranno le LAN allo spanning tree):

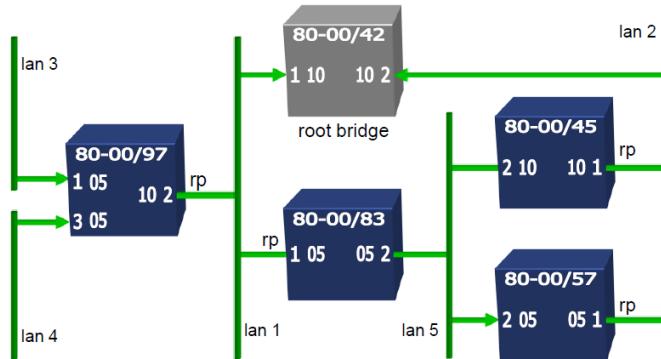


Figura 158 - Fase 3: determinazione delle designated port

Tutto il resto viene tagliato (fase 4).

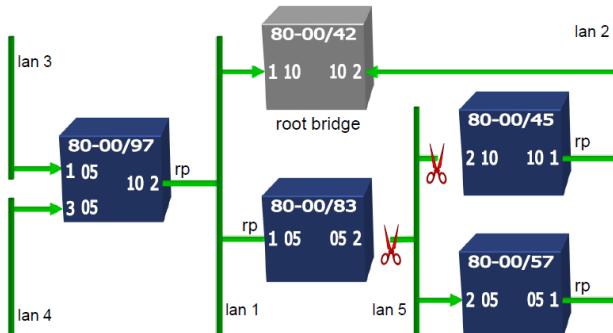


Figura 159 - Fase 4: blocking

Che algoritmo stiamo guardando? Qui non c'è neppure l'ombra dell'instradamento, eppure stiamo calcolando lo spanning tree usando un algoritmo di instradamento. Quale e perché? L'algoritmo per il calcolo dell'albero ricoprente non è altro che Bellman-Ford, in cui il costo di attraversamento della porta di ingresso è il costo per arrivare al vicino.

## Software Defined Networks

Software Defined Network o Networking (SDN) è un paradigma di routing centralizzato; divide fisicamente il control plane dal data plane. Il paradigma SDN (*Software Defined Networking*), secondo l'accezione più diffusa, propone l'ambiziosa visione di rendere i nodi di rete (ad es. router e switch) programmabili, introducendo opportuni livelli di astrazione, ai quali accedere attraverso l'uso di interfacce di controllo (API). Nell'ambito di questo paradigma, assume particolare rilievo anche il concetto di virtualizzazione di rete, ovvero l'idea di creare delle partizioni virtuali dell'infrastruttura di rete fisica, in modo da permettere a più istanze di controllo e le rispettive applicazioni di utilizzare le partizioni assegnate: questo permetterebbe coesistenza di più reti virtuali, completamente isolate, che insistono sulla medesima infrastruttura hardware. La centralizzazione logica del controllo, potrebbe permettere di attuare più facilmente azioni di configurazione e ottimizzazione delle risorse di rete. L'adozione di questo paradigma potrebbe abilitare lo sviluppo di nuovi ecosistemi.

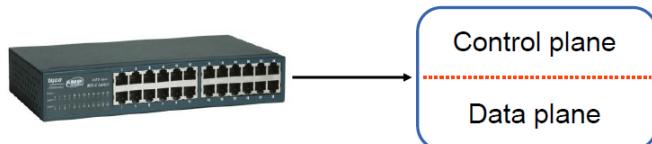


Figura 160 - SDN

Le funzioni del control plane vengono centralizzate in un server che è in grado di determinare per ogni flusso di traffico uno specifico cammino nella rete: l'approccio SDN permette all'amministratore della rete di avere un controllo fine su ogni cammino. Le funzioni del data plane rimangono distribuite.

Mentre in una rete tradizionale il routing viene realizzato configurando gli apparati distribuiti sulle reti, nelle reti SDN è calcolato da un software che è eseguito sul server codice vs configurazione.

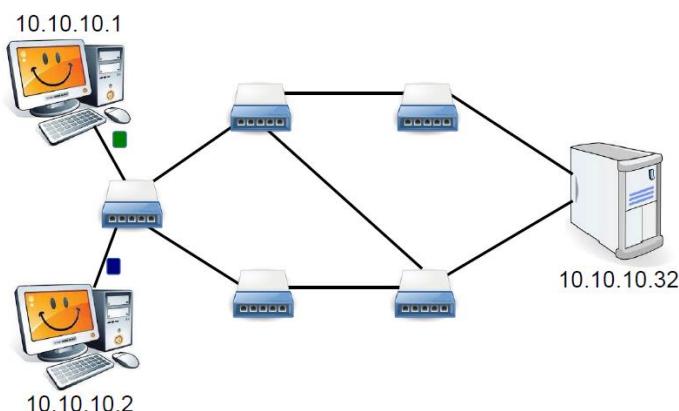


Figura 161 - Gradi di libertà nelle SDNs

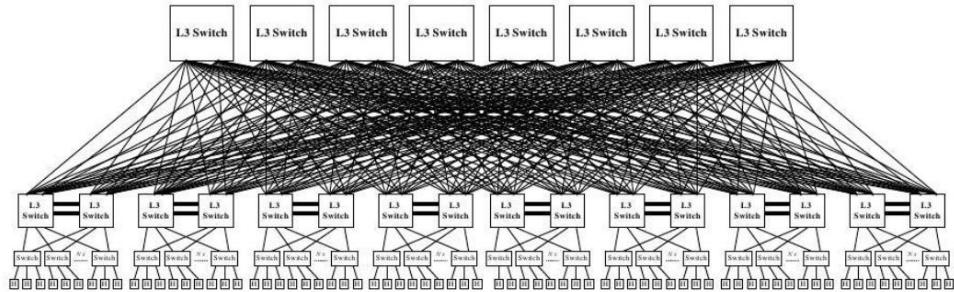


Figura 162 - Rete di switch

Il numero di switch, link e macchine virtuali (VM) in un data center è impressionante! se sulla rete fosse in funzione un solo spanning tree la maggior parte dei link sarebbe inutilizzato gestire contemporaneamente molti spanning tree per aumentare l'efficienza non consente di scalare.

Alcune applicazioni potenziali sono:

- *virtualizzazione delle appliance-middlebox* (firewall, bilanciatori di carico, content distribution);
- *qos* (routing ottimizzato per tipologie di traffico, traffic engineering policy based);
- servizi specifici per il livello delle applicazioni (reti SDN application-specific, banda application-specific);
- *cloud* (rete service-centric e user-centricpiuttosto che device-centric; virtualizzazione dei servizi di rete in ambienti fortemente dinamici con limitati costi di provisioning).

## Architetture delle SDN e OpenFlow

SDN separa il control plane dal data plane:

- controller: esegue il software, eventualmente algoritmamente complesso, di control plane; può essere eseguito su hardware “general purpose”, come un PC.
- switch(o datapath): fa solo forwarding del traffico (assimilabile agli attuali switch e/o router).

La comunicazione tra controller e switch avviene tramite apposita API.

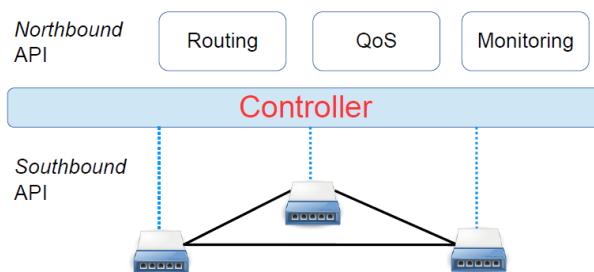


Figura 163 - Architettura di SDN

## OpenFlow

OpenFlow è un protocollo che implementa il paradigma SDN ed è basato sull'idea di flusso: un flusso può essere definito sulla base di diversi criteri (porta, indirizzo MAC, indirizzo IP, ...) ed ogni pacchetto appartiene ad un flusso. Lo switch inoltra traffico seguendo regole (flow entry) contenute nella tabella dei flussi (flow table).

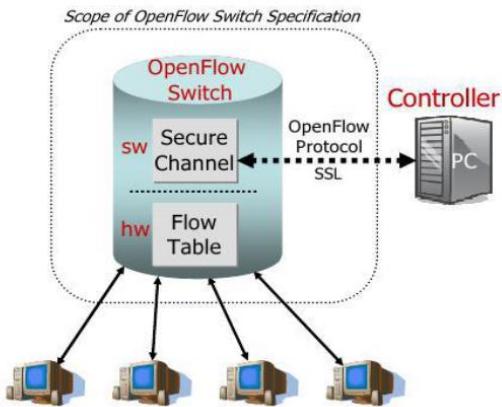


Figura 164 - Architettura di OpenFlow

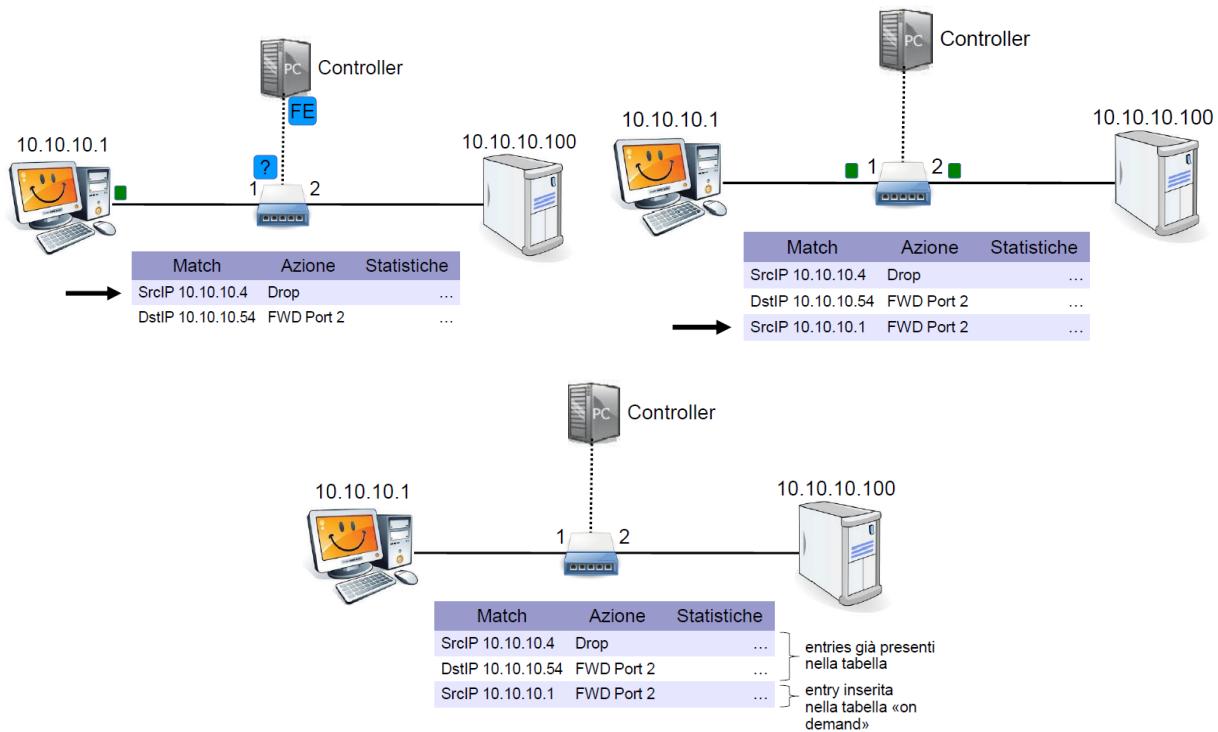


Figura 165 - Funzionamento di OpenFlow

### Attori

- Switch - data plane (solo forwarding dei pacchetti);
- Controller - control plane (il controller è un'entità software, non la macchina sulla quale essa è in esecuzione).

### Flussi di traffico

Ogni pacchetto appartiene ad un flusso: Porte TCP, Indirizzi IP e MAC, VLAN ID, Label MPLS, Porta di ingresso, ecc. Ogni switch ha una (o più) flow table usate per il forwarding; una flow table è composta da flow entry:

- flow entry: <match, action, stats>;
- es: <port1, forwardto port2, 184bytes14packets>.

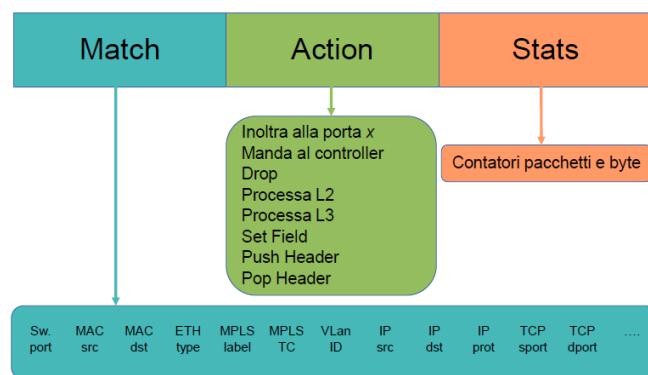


Figura 166 - Flow entry di tabelle in OpenFlow

Ogni Flow entry ha associati due timeout:

- *Idle timeout*: la regola scade dopo un tempo  $t$  di inattività;
- *Hard timeout*: la regola scade dopo un tempo  $t$ ; la regola può scadere anche se un flusso la sta usando.

Un controller è un'entità software che conosce l'intera topologia e produce flow entry da distribuire agli switch per il forwarding del traffico. I tipi di messaggi principali, per il routing:

- PacketIn switch → controller;
- PacketOut switch → controller;
- FlowMod switch → controller;
- FlowRemoval lswitch → controller.

## Messaggi

### Messaggio PacketIn

Usato dallo switch per notificare al controller il fatto che non sa gestire il flusso di traffico che gli è arrivato. Le alternative sono due, lo switch invia al controller:

- tutto il pacchetto che non sa come gestire;
- solo una porzione dell'header(di default i primi 128 bit): in questo caso dichiara al controller un ID del buffer nel quale mantiene il pacchetto in attesa della regola.

No.	Time	Source	Destination	Protocol	Length	Info
66	0.567074	f2:e6:9a:e7:f5:a0	NiciraNe_00:00:01	OFPHLLD	131	Packet Out (CSM) (65B) => Chassis Id = dpid:1 Port Id = 1 TTL = 120
116	2.296189	9a:ce:20:38:51:e2	NiciraNe_00:00:01	OFPHLLD	131	Packet Out (CSM) (65B) => Chassis Id = dpid:1 Port Id = 3 TTL = 120
126	3.956866	0a:7c:79:7d:ae:5b	NiciraNe_00:00:01	OFPHLLD	131	Packet Out (CSM) (65B) => Chassis Id = dpid:1 Port Id = 2 TTL = 120
130	4.019508	00:00:00:00:00:01	Broadcast	OF+ARP	126	Packet In (AN) (BufID=276) (60B) => Who has 10.0.0.2? Tell 10.0.0.1
131	4.043737	127.0.0.1		OFP	90	Packet Out (CSM) (BufID=276) (24B)

Packet In  
 Buffer ID: 276  
 Frame Total Length: 42  
 Frame Recv Port: 1  
 Reason Sent: No matching flow (0)  
 Frame Data: ffffffffffffff00000000001080600010800060400010000...  
 Ethernet II, Src: 00:00:00:00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 Address Resolution Protocol (request)  
 Hardware type: Ethernet (1)  
 Protocol type: IP (0x0800)  
 Hardware size: 6  
 Protocol size: 4  
 Opcode: request (1)  
 [Is gratuitous: False]  
 Sender MAC address: 00:00:00:00:00:01 (00:00:00:00:00:01)  
 Sender IP address: 10.0.0.1 (10.0.0.1)  
 Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Target IP address: 10.0.0.2 (10.0.0.2)

128 bit dell'header

Figura 167 - Messaggio PacketIn

### Messaggio PacktOut

Se è in risposta ad un PacketIn è usato dal controller per istruire lo switch sull'invio di un singolo pacchetto (non viene installata nessuna regola). Se lo switch ha inviato al controller l'intero pacchetto, PacktOut contiene il pacchetto assieme alla regola da eseguire, altrimenti assieme alla regola è specificato l'ID del buffer nel quale lo switch ha dichiarato di mantenere il pacchetto. Se non è in risposta ad un PacketIn, allora contiene un pacchetto *ad-hoc* creato dal controller, usato ad esempio per la ricostruzione della topologia.

No.	Time	Source	Destination	Protocol	Length	Info
66 0.567074	f2:6f:9a:e7:f5:a0	NiciraNe_00:00:01		OPPHLLDI	131	Packet Out (CSM) (65B) => Chassis Id = dpid:1 Port Id = 1 TTL = 120
118 2.236189	9a:ce:20:38:51:e2	NiciraNe_00:00:01		OPPHLLDI	131	Packet Out (CSM) (65B) => Chassis Id = dpid:1 Port Id = 2 TTL = 120
129 3.956866	0a:c7:79:7d:a5:b5	NiciraNe_00:00:01		OPPHLLDI	131	Packet Out (CSM) (65B) => Chassis Id = dpid:1 Port Id = 3 TTL = 120
130 4.019508	00:00:00:00:00:01	Broadcast		OPPHARP	126	Packet In (AM) (BufID=276) (60B) => Who has 10.0.0.2? Tell 10.0.0.1
131 4.043737	127.0.0.1		127.0.0.1	OFP	90	Packet Out (CSM) (BufID=276) (24B)

▷ Frame 131: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)  
 ▷ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 ▷ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)  
 ▷ Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 35632 (35632), Seq: 196, Ack: 61, Len: 24

▼ OpenFlow Protocol  
 ▷ Header  
 ▷ Packet Out  
 Buffer ID: 276  
 Frame Recv Port: 1  
 Size of action array in bytes: 8  
 ▷ Output Action(s)  
 ▷ Action  
 Type: Output to switch port (0)  
 Len: 8  
 Output port: Flood (all physical ports except input port and those disabled by STP)  
 Max Bytes to Send: 0  
 # of Actions: 1

} action, flood

Figura 168 - Messaggio PacketOut

### Messaggio FlowMod

Flow Modification è usato dal controller per installare una regola in uno switch: può essere causata da un PacketIn o da un atteggiamento pro-active del controller. I tipi di azione specificabili sono gli stessi specificabili in PacketOut. In questo messaggio è anche contenuta la specifica dei timeout per le flow entry. Inoltre, è possibile specificare una priorità sulla flow entry: se ci sono due regole per la gestione di un flusso prevale quella a priorità più alta, altrimenti se la regola che viene installata ha la stessa priorità di una regola già presente e un pacchetto potrebbe fare match con entrambe, allora tale regola viene sovrascritta dalla nuova (comportamento di OpenVswitch). Consente inoltre di chiedere allo switch:

- di notificare al controller l'istante dell'eventuale rimozione della regola: la notifica contiene anche le statistiche sull'uso della regola;
- di verificare se la regola si sovrappone logicamente ad altre regole già installate.

No.	Time	Source	Destination	Protocol	Length	Info
139 4.050604	00:00:00:00:00:02	00:00:00:00:00:01		OPPHARP	126	Packet In (AM) (BufID=277) (60B) => 10.0.0.2 is at 00:00:00:00:00:02
134 4.053408	127.0.0.1	127.0.0.1		OFP	146	Flow Mod (CSM) (80B)
135 4.054461	10.0.0.1	10.0.0.2		OPPHCM	182	Packet In (AM) (BufID=278) (116B) => Echo (ping) request id=0xc4d2, seq=1/256, t
136 4.056117	127.0.0.1	127.0.0.1		OFP	146	Flow Mod (CSM) (80B)

▼ OpenFlow Protocol  
 ▷ Header  
 ▷ Flow Modification  
 ▷ Match  
 ▷ Match Types  
 Input Port: 1  
 Ethernet Src Addr: 00:00:00:00:00:01 (00:00:00:00:00:01)  
 Ethernet Dst Addr: 00:00:00:00:00:02 (00:00:00:00:00:02)  
 Input VLAN ID: 65535  
 Input VLAN priority: 0  
 Ethernet Type: IP (0x0800)  
 IPv4 DSQoS: 0  
 Protocol: ICMP (0x01)  
 IP Src Addr: 10.0.0.1 (10.0.0.1)  
 IP Dst Addr: 10.0.0.2 (10.0.0.2)  
 ICMP Type: Echo (ping) request  
 ICMP Code: 0 (-)  
 Cookie: 0x0000000000000000  
 Command: New flow (0)  
 Idle Time (sec) Before Discarding: 10  
 Max Time (sec) Before Discarding: 30  
 Priority: 32768  
 Buffer ID: 278  
 Out Port (delete\* only): None (not associated with a physical port)

} matching, pacchetti dalla porta 1, con MAC mittente 00:00:00:00:00:01, MAC destinatario 00:00:00:00:00:02, Input VLAN ID 65535 (no VLAN), etc.

▼ Output Action(s)  
 ▷ Action  
 Type: Output to switch port (0)  
 Len: 8  
 Output port: 2  
 Max Bytes to Send: 0  
 # of Actions: 1

} action; il pacchetto specifica una sola azione

Figura 169 - Messaggio FlowMod

### Messaggio FlowRemoval

È usato dal controller per rimuovere una regola dallo switch. Una regola può dover essere rimossa a causa di variazioni topologiche (*fault*) oppure a causa di variazioni amministrative (uno o più flussi devono seguire un percorso diverso nella rete).

### Interazione con la topologia

Il controller ha bisogno di conoscere la topologia della rete; per farlo, attualmente usa *Link Layer Discovery Protocol* (LLDP), un protocollo di livello 2 link state, i cui pacchetti sono spediti periodicamente per esplorare la topologia. Il controller fa handshake con tutti gli switch: il primo pacchetto di handshake è inviato dallo switch, che dichiara al controller l'elenco delle proprie interfacce.

## Border Gateway Protocol (BGP)

### Routing interdominio

Abbiamo usato finora algoritmi di instradamento di due categorie principali: link state packet e distance vector. Per noi un dominio non è un sotto-albero dello spazio dei nomi, dominio significa una organizzazione, una rete, che è sotto una singola autorità amministrativa. Il routing interdominio quindi è il routing tra diverse reti, che sono sotto diverse autorità amministrative. Un'organizzazione è una collezione di router e LAN gestite da una singola amministrazione. Quando ci sono diverse organizzazioni che si uniscono per formare internet, in linea di principio all'inizio sono disconnesse. Dentro una rete in alcune zone di può usare OSPF o is-is, in altre RIP, non è detto che ci sia un solo protocollo. Le organizzazioni per dialogare tipicamente devono predisporre delle apparecchiature usate per il collegamento con altri provider. Queste apparecchiature vanno collegate alle singole reti, e poi tra loro.

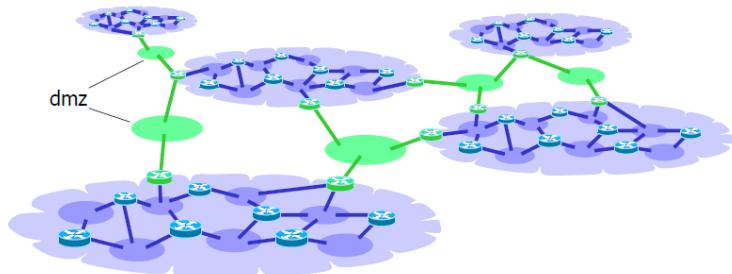


Figura 170 - Organizzazione che gestisce una collezione di router e LAN con un solo riferimento amministrativo

Le DMZ, zone di demarcazione, non sono sotto il vincolo amministrativo diretto dei provider, ma la loro presenza testimonia un accordo tra i provider. In questa rete al momento non si può scambiare traffico: dentro ciascuna rete c'è routing, ma non c'è routing tra una rete e l'altra. Se voglio avere connettività globale è chiaro che ogni router in tutta la rete deve avere una entry per ciascuna delle destinazioni che devono essere raggiunte. Questa riga può essere anche la o/o, la rotta di default. Indichiamo i prefissi con delle bandierine.

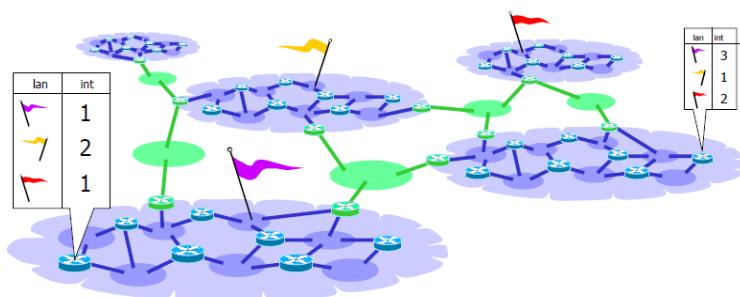


Figura 171 - Organizzazione che gestisce una collezione di router e LAN con un solo riferimento amministrativo

I protocolli devono essere tali che può entrare nella rete senza dare conto a nessuno. Ci deve essere una entry per ciascun prefisso in ciascuno dei router che compongono la rete. Come si fa a costruire queste tabelle di instradamento? In linea di principio ci sono 3 opzioni:

1. Possiamo far girare un singolo algoritmo di instradamento su tutta la rete;
2. Alle tabelle vengono aggiunte delle rotte statiche per l'instradamento esterno, che sono addizionali rispetto a quelle calcolate dal protocollo di instradamento interno;
3. Combinare un *Exterior Gateway Program* (EGP) con l'IGP delle reti (scelta giusta).

#### Singolo protocollo di instradamento

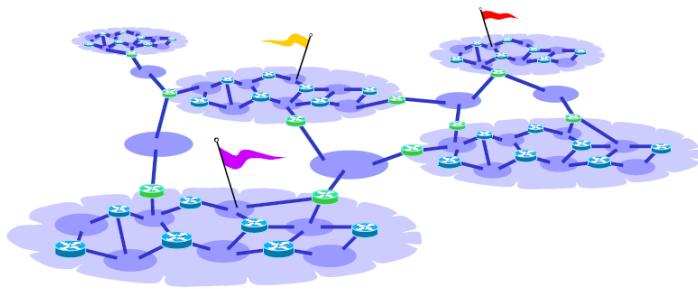


Figura 172 - Organizzazione di un singolo protocollo di instradamento

Chiedo a tutti i provider di realizzare un'unica rete con un unico protocollo di instradamento. Questa è la scelta fatta all'inizio di internet, quando è stato introdotto il concetto di EGP. Questa scelta ha problemi di natura tecnica e politica (scelte di natura economica, di concorrenza, geopolitica...).

- Problemi tecnici: con un unico protocollo dispiegato su una rete molto grande i tempi di convergenza sono molto lenti. Questi problemi di convergenza forse si potrebbero anche affrontare. Dove sono già presenti dei protocolli di instradamento, dispiegare e rendere operativo un nuovo protocollo è una sfida molto significativa; supponiamo che ci sia un bug su questo nuovo protocollo: le conseguenze sarebbero terribili.
- Problemi politici: l'obiettivo del routing è quello di minimizzare l'uso delle risorse di rete, ma questo minimizzare (la lunghezza dei cammini) non tiene conto del possesso effettivo delle linee. È molto probabile che provider di partenza di arrivo e intermedi non siano molto contenti di questa storia, per motivi di geopolitici. Se fai shortest path fai shortest path. Posso anche pensare di drogare le metriche, per forzare l'algoritmo a fare determinate scelte, ma chi mi garantisce che l'algoritmo funzioni come voglio? E chi se ne deve occupare? Questa scelta viene rapidamente abbandonata.

### Rotte statiche

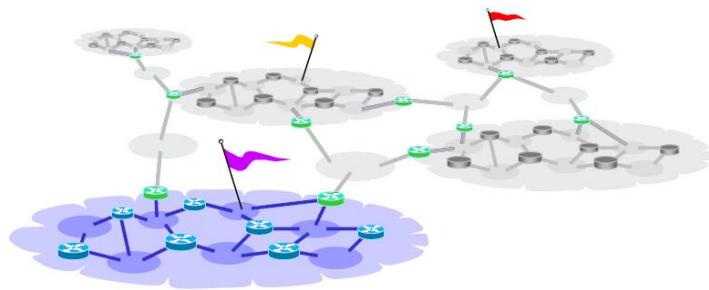


Figura 173 - Organizzazione per rotte statiche

Pensiamo al punto di vista di una delle organizzazioni, e ignoro la struttura interna delle altre organizzazioni, ma non ignoro i prefissi che ho intenzione di raggiungere. Faccio normalmente routing all'interno della mia infrastruttura locale, e dico che i prefissi esterni saranno raggiungibili dai router di frontiera, su cui metto le rotte statiche che mi dicono qual è il next hop per raggiungerle.

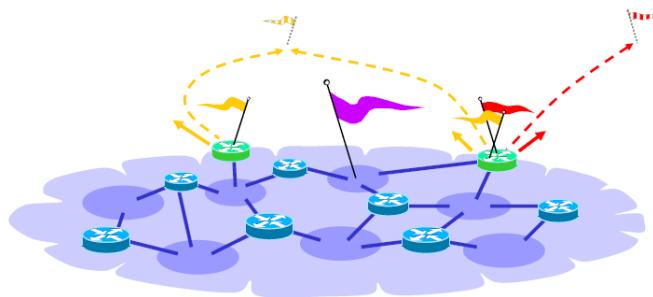


Figura 174 - Organizzazione per rotte statiche

Se quello che vedo fuori non è troppo articolato posso evitare di dire "il prefisso rosso va lì, il giallo va di là", ma posso cavarmela con una rottura di default. Posso ridistribuire le rotte apprese dal protocollo esterno (rotta gialla, rottura rossa) e ridistribuirle dentro il protocollo esistente dentro l'infrastruttura. Redistribuisco i prefissi dentro l'IGP, e sarà l'IGP che contribuirà a trasportare dentro la rete sia i target locali, sia i prefissi locali, sia i prefissi remoti, che sono stati installati sui router che stanno sull'altra frontiera.

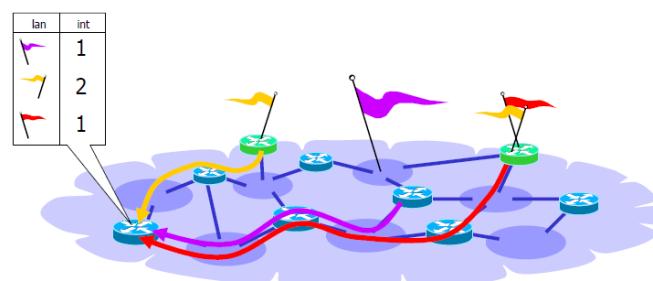
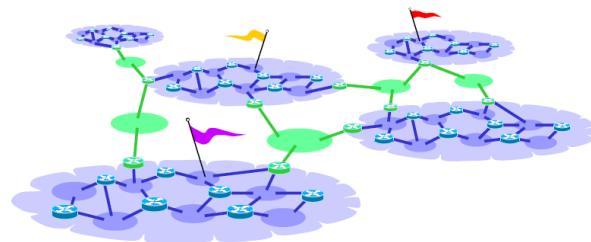


Figura 175 - Organizzazione per rotte statiche

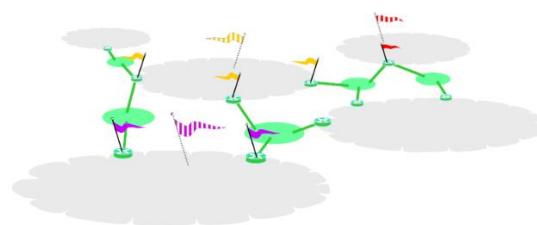
Anche qui ci sono problemi di natura tecnica e politica. Dal punto di vista tecnico ci sono tutti i problemi caratteristici delle rotte statiche: difficili da aggiornare, difficile fare il debug. Non solo: se c'è un fault la rotta statica continua a essere disponibile. Se un link è giù, a ridosso di quel router, o anche molto lontano, la rotta statica sta sempre lì. Problemi politici: che garanzia c'è che il next hop a cui mi riferisco sia disponibile a instradare traffico verso il prefisso giallo? Se è disponibile, chi mi garantisce che siano disponibili a farlo anche gli altri provider attraversati?

#### Exterior gateway protocol



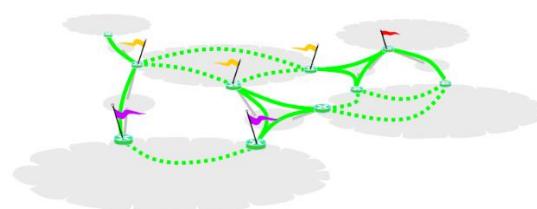
*Figura 176 - Organizzazione di una exterior gateway protocol*

Proviamo a trascurare la parte interna delle organizzazioni: questo non soltanto scala, ma è anche un vincolo ineludibile: un provider probabilmente non ha nessuna voglia di raccontare ad un altro provider come è fatta l'infrastruttura del suo backbone. La prima scelta che si fa quando si parla di un EGP è quindi ignorare la parte interna delle organizzazioni, ma a questo punto come si fa per i prefissi? Ogni router di bordo rappresenta i target interni alla propria infrastruttura come se fossero locali. Anche qui si può usare una rotta di default.



*Figura 177 - Rappresentazione dei target interni*

A questo punto il grafo della rete è estremamente semplificato: è un grafo in cui si può considerare soltanto la raggiungibilità esterna dei router:



*Figura 178 - Grafo di raggiungibilità della rete*

Si prendono tutte le reti dei provider, le si fanno diventare dei nodi, e intorno ci si mettono altri nodi (un nodo per ogni router di frontiera), e si usano gli archi per rappresentare i collegamenti tra i provider. Accanto a questo primo grafo dobbiamo aggiungere un insieme di archi (tratteggiati) che rappresentano la connettività tra i router di un provider. Il grafo viene gestito con connessioni TCP, chiamate peering. Questi peering sono roba pregiata, non possiamo perderci pacchetti. Sotto TCP, per poter far funzionare le cose, ci deve essere un livello 3 che funziona. Il fatto che ci sia un peering TCP che veicola le informazioni di EGP, impone che ci sia connettività anche indipendentemente dall'EGP. Come si fa? I router si scambiano le rotte, basta che si vedono da una parte all'altra di un filo. Ci deve essere connettività anche tra i router di bordo, indipendentemente dall'EGP. Il problema dell'instradamento viene quindi risolto su questo grafo semplificato.

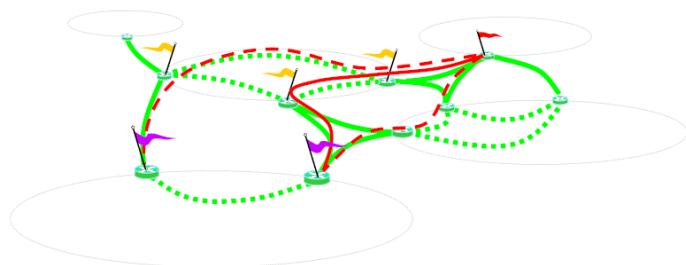


Figura 179 - Soluzione del problema di routing

Dal punto di vista del rapporto tra l'interno e l'esterno, come la giochiamo? Ce la giochiamo esattamente come per le rotte statiche. Il protocollo EGP ci porta le informazioni di raggiungibilità dei router esterni sui router di frontiera, e posso redistribuire all'interno queste informazioni. Abbiamo però delle rotte statiche, non è un problema? No, il protocollo esterno si occupa di dire per esempio "il prefisso giallo da qui non è più raggiungibile", e questa informazione viene poi redistribuita.

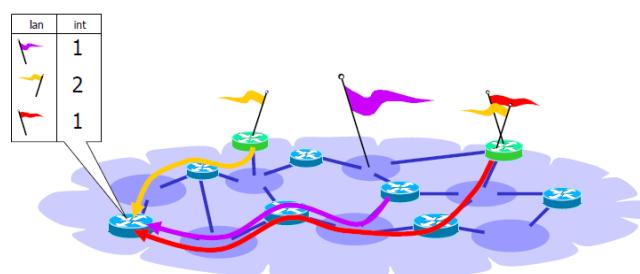


Figura 180 - Inject routes

## Border Gateway Protocol (BGP)

Il protocollo che fa tutto questo è BGP (Border Gateway Protocol): prende le tabelle aggiornate di routing e propaga le informazioni di instradamento. Questo è uno dei protocolli legacy di internet, come IP. Il fatto che sia legacy è la cattiva notizia, la buona notizia è che funziona bene; tiene conto della volontà delle organizzazioni di cooperare nel processo di routing (consente di esprimere accordi commerciali, preferenze locali, priorità, problemi legali).

### Aspetti essenziali

BGP è specificato nella RFC 4271, con una montagna di RFC intorno, ciascuno per un aspetto specifico, tra cui la RFC 4276 e la 4277. BGP è usato da:

- clienti connessi ad un ISP;
- clienti connessi a più ISP;
- provider di transito;
- ISP che scambiano traffico in un internet *exchange point* (IXP) o in un *neutral access point* (NAP);
- clienti con reti molto grandi.

Quasi tutte queste parole meritano un approfondimento. Un IXP è un punto fisico in cui i provider mettono i propri router di frontiera. Di questi IXP ce ne sono alcuni che si chiamano NAP. In un NAP nessuno paga nessuno, c'è un reciproco interesse a raggiungersi a costo zero.

### Autonomous system

Quando i player si presentano su internet assumono un nome ben definito. La rete autonoma, sotto una singola autorità amministrativa, per BGP si chiama autonomous system; sono quindi oggetti che parlano BGP. Il singolo provider, quello che è diventato un nodo nella rete internet, è un AS, un numero usato per identificare reti con una politica di routing comune. Questo numero non gira dentro i pacchetti IP, serve soltanto a BGP.

Ci sono due range di AS:

- 0-65535 (original 16-bit range);

65536-4294967295 (32-bit range - RFC4893) L'utilizzo è questo:

- 1-64511 (public internet);
- 64512-65534 (private use only);
- 23456 (to represent 32-bit range in 16-bit world);
- 0 and 65535 (reserved);
- 65536-4294967295 (public internet).

Gli indirizzi privati fanno un po' il paio con gli indirizzi di classe 10. Servono a strutturare la mia rete come se fosse una rete composta da diversi provider. Non se ne parla fuori, fuori dalla rete l'AS viene visto

comunque come un AS. A chi si chiede un numero di AS? Al regional internet registry (RIR) di riferimento. Se si vuole un indirizzo di AS privato ci si rivolge all'upstream ISP.

### Peering

BGP consente ai router di scambiare informazioni soltanto se c'è un peering, una connessione TCP sulla porta 179, sulla quale connessione si posso scambiare informazioni di routing.

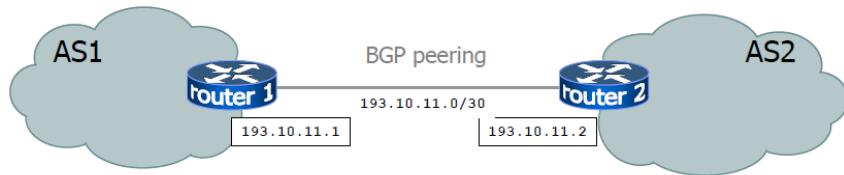


Figura 181 - BGP peering

### e-BGP e i-BGP

Di BGP ce ne sono 2:

- *e-BGP*: è fatto per BGP speaker che sono in diversi AS.
- *i-BGP*: si parla all'interno dell'AS, fra router di bordo dello stesso AS.

Si usa e-BGP tra due BGP speaker che sono in due AS differenti. Qui si fa l'assunzione molto forte che affinché si possa mettere in piedi una connessione i router siano direttamente connessi (single hop), e questo è fondamentale anche per motivi di sicurezza. C'è bisogno anche di modellare il rapporto tra i router interni di frontiera, e per fare questo si usa i-BGP. In questo caso non è richiesto che i router siano connessi direttamente (queste connessioni si chiamano connessioni multi-hop); quando si fa iBGP si ammette (anzi, si assume) che si attraversa l'infrastruttura del provider (che può avere in linea di principio tantissimi router da attraversare). Importante: i-BGP non è un IGP: anche se sta dentro l'AS il suo ruolo è assolutamente esterno, serve a sfruttare la connettività interna per fini esterni. Piuttosto, si basa su un IGP, i peering si sovrappongono a IGP sfruttandone la connettività. A che cosa serve un IGP? Un IGP serve a realizzare piena connettività all'interno di un AS, scelgo un IGP per fare spread delle rotte all'interno di un AS. Non posso usare BGP per fare lo stesso lavoro, è fatto per un'altra cosa. Il dialogo è fra i router di frontiera di un'organizzazione, non per i router che stanno all'interno. Che cosa vuol dire che i-BGP si basa su un IGP? se hai due router di frontiera che stanno tanto lontani tra loro, questi devono mettere in piedi un peering iBGP, devono parlare fra loro; ci sarà una connessione TCP che supporta questo dialogo, e in mezzo c'è un'infrastruttura sulla quale routing non ci sarebbe se non ci fosse un IGP.

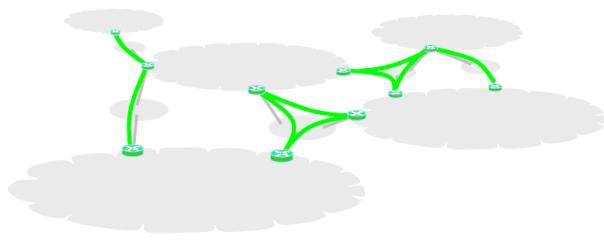


Figura 182 - e-BGP

eBGP è utilizzato da coppie di router in differenti AS per il routing interdominio; iBGP è utilizzato da coppie di router all'interno dello stesso AS per far passare le informazioni di routing attraverso l'AS:

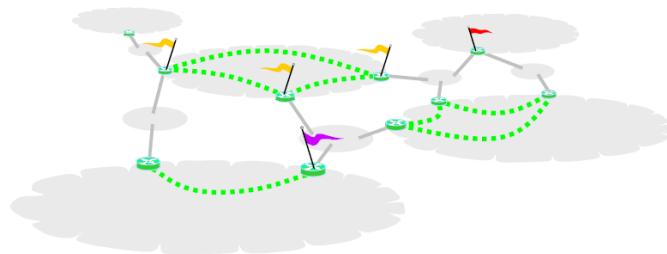


Figura 183 - i-BGP

e i router che stanno sulla frontiera di un AS devono fare iBGP peering in full mesh, cioè tutti con tutti, una specie di ragnatela completa (almeno in linea di principio, gli AS sono giganteschi 100 router in full mesh fanno 10000 connessioni), questo perché da ovunque arriva un'informazione tutti i nodi di bordo la devono sapere perché eventualmente la devono propagare oltre. Cosa fanno gli speaker iBGP? Si fanno attraversare dagli speaker appresi dall'esterno degli AS, non dai prefissi che sono stati appresi da altri iBGP speaker, perché la connessione è in full mesh, se apprendi una cosa dall'esterno la racconti dall'altra parte, se la apprendi dall'interno stai semplicemente replicando una cosa che è già nota; se non fosse con la full mesh avremmo dei cicli, questi prefissi sarebbero raccontati ciclicamente. Un router BGP tipicamente apprende varie rotte da BGP speaker interni o esterni, seleziona il cammino migliore per raggiungere ciascuna destinazione, lo installa nella tabella di forwarding e il migliore tra i cammini è spedito ai suoi vicini (non necessariamente a tutti). Si possono usare delle politiche per influenzare la selezione del cammino migliore.

## Zebra – Quagga

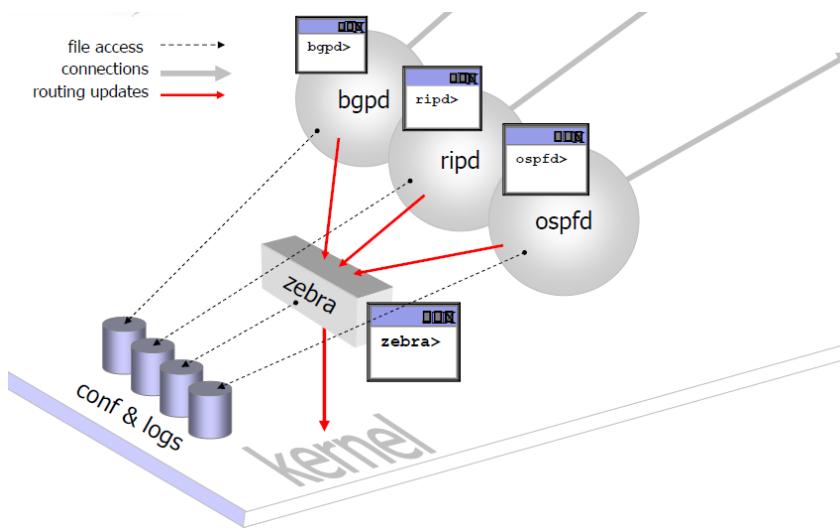


Figura 184 - Struttura Zebra

D'ora in avanti si interagirà col routing così. Sotto c'è il kernel, la macchina, con la sua tabella di instradamento (che abbiamo sempre chiamato data plane), che può essere popolata a mano o facendo uso di protocolli di instradamento. In Netkit i protocolli di routing sono implementati nel demone Zebra, e gli altri demoni (BGPD, ripd, ospfd) interagiscono con Zebra. BGPD, se è attivo, calcola la sua tabella di instradamento e la dice a zebra, che costruisce la sua tabella di instradamento e la mette nel data plane, che serve poi a fare effettivamente lo smistamento dei pacchetti. Ogni demone quindi calcola la sua tabella di instradamento dal suo punto di vista (sullo stesso livello 3, sullo stesso data plane possono essere attivi diversi control plane).

### Comandi di configurazione

---

```
router bgp <my-as-number>
```

---

Io posso dire "sono un router, parlo bgp e questo è il mio numero di AS"

---

```
neighbor <neighbor-ip> remote-as <neighbor-as-num>
```

---

Devi mettere in piedi un peering con un tuo vicino di cui questo è l'indirizzo IP, e che fa parte di questo AS.

---

```
neighbor <neighbor-ip> description <text>
```

---

Possiamo associare ad un vicino anche una descrizione. Questo comando non ha nessuna semantica operativa.

Esempio di peering

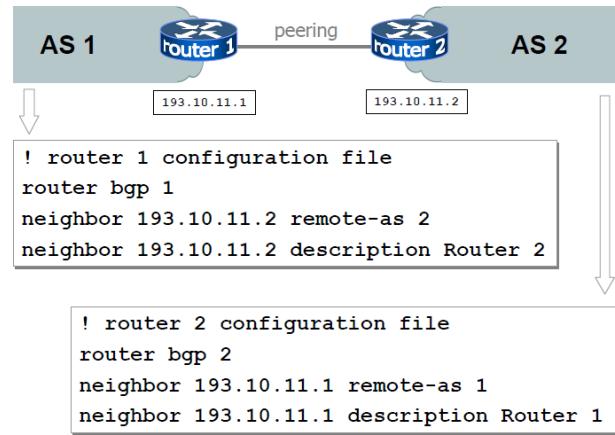


Figura 185 - Esempio di peering

Il router 1 dichiara di appartenere all'AS 1, e indica la volontà di instaurare un peering con 193.10.11.2, e analogamente fa il router 2.

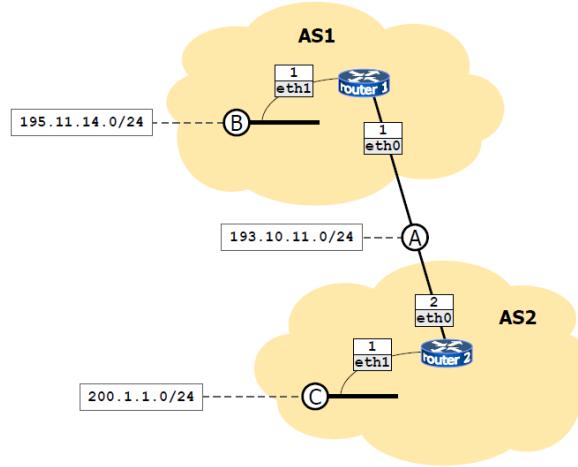


Figura 186 - Peering semplice

### Annuncio

Annunciare un prefisso in BGP significa offrire connettività ad un altro router, dire "posso recapitare il traffico relativo a quel prefisso".

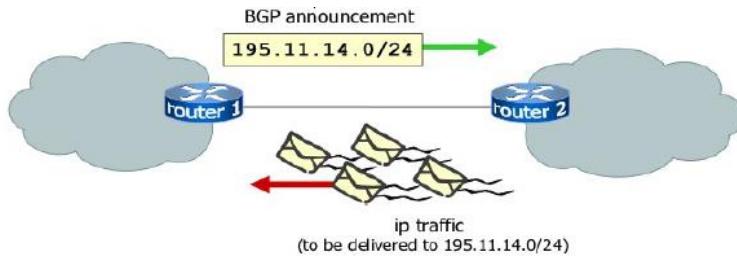


Figura 187 – Esempio

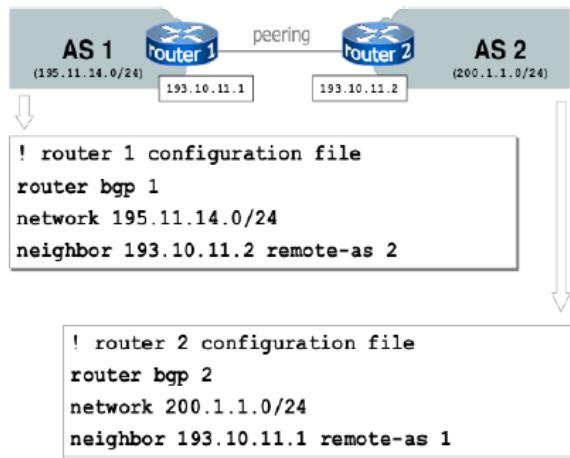
Questo esempio è molto semplice: ci sono AS1 e AS2 con dentro una singola LAN. È chiaro che se router 1 dice a router 2 "io ti annuncio quella LAN" è facile per lui portare il traffico a destinazione, perché quella LAN sta lì, ma bisogna pensare che questo è un processo che attraversa tanti AS, per cui io router 1 dico a te, router 2, che io posso raggiungere quella /24, perché quella /24 è mia, sta all'interno del mio AS. A questo punto tu, router 2, puoi raccontare questa stessa storia ad un router 3 di un fantomatico AS2, il quale sa a questo punto che quella /24 può essere raggiunta consegnando traffico a router 2, il quale sa che la può raggiungere consegnando traffico a router 1. Quindi questi annunci nascono nel punto in cui questo prefisso è effettivamente presente e poi si propagano attraverso la rete, da un AS ad un altro AS. Questo è il comportamento standard di BGP, poi quando si dice "da un AS ad un altro AS" si capisce che al di fuori può essere complessa: qui router 2 potrebbe avere direttamente degli altri peering con degli altri router, oppure potrebbe avere dei peering iBGP con degli altri router di frontiera dello stesso AS, per cui questo annuncio potrebbe passare attraverso l'AS prima di propagarsi verso il resto della rete. Il comando zebra per dire che un prefisso è locale ad un certo AS, e quindi per fare un annuncio, è:

---

```
network <network-ip/network-mask>
```

---

Dobbiamo fare un annuncio per ciascuna LAN presente nell'AS, ma nello stesso annuncio possiamo anche includere diversi prefissi.



Per risparmiare si tende a raggruppare prefissi nello stesso annuncio. I prefissi 195.11.14.0/24 e 200.1.1.0/24 sono rispettivamente locali ad AS1 e ad AS2. Se io ho tanti prefissi in AS1 che cosa faccio? Devo dire al router “network numero 1, network numero 2, network numero 3...” oppure posso automatizzare questo processo? Questo processo si automatizza disegnando un protocollo di routing all'interno dell'AS, ad esempio RIP, che funziona come ulteriore demone oltre a quello BGP che sta già funzionando. Questi prefissi possono arrivare a router 2, e che cosa può fare? RIP può ridistribuire tutti i prefissi che sono stati appresi in AS1 dentro BGP, e a questo punto li faccio annunciare automaticamente fuori. Però non si fa così. Molto (molto) difficilmente troviamo un provider che ridistribuisce l'IGP dentro BGP, perché se ti sbagli nella configurazione interna il danno lo fai grosso, qui quello che annuncia lo annuncia a tutta internet. Se sbagli a digitare un prefisso, e per sbaglio annuncia il prefisso di qualcun altro, stai dicendo che quel prefisso è tuo, invece che di qualcun altro, il traffico raggiunge te, quindi è troppo pericoloso. Questo è il motivo per cui il comando `network` visto prima, che sembra troppo “scemo” per essere effettivamente utilizzato, è quello che viene usato nella pratica. Chiunque abbia un gruppo di prefissi si guarda molto bene dall'annunciarli tutti all'esterno, è troppo pericoloso.

### Policy

Non tutti gli annunci devono essere necessariamente guardati, alcuni possono essere buttati. Non soltanto: quando si fanno degli annunci si annuncia il best, la scelta migliore per raggiungere un certo prefisso. Se vuoi questo annuncio lo puoi fare sennò no. Si possono anche modificare gli annunci, applicando delle policy. Per capire bene il significato di questo lavoro dobbiamo capire che un annuncio BGP è un bag di attributi. Questi attributi sono informazioni aggiuntive sul prefisso che viene annunciato; gli attributi possono essere well-known (obbligatori) oppure opzionali, e possono essere anche transitivi o non transitivi. Gli attributi transitivi sono passati (quando mi arriva l'annuncio passo a qualcun altro anche l'attributo transitivo), gli attributi non transitivi attraversano soltanto un hop. Manipolando opportunamente questi attributi BGP consente di esprimere quelle politiche di cui abbiamo discusso quando è stato introdotto il routing interdominio.

- Attributi well-known obbligatori:
  - as-path: transitivo, specifica la sequenza degli AS che sono stati attraversati da quell'annuncio;
  - next-hop: non transitivo, va messo obbligatoriamente nella tabella di instradamento. Dice come raggiungere la destinazione attraverso un next-hop. In iBGP rimane invariato;
  - origin: transitivo.
- Attributi well-known opzionali:
  - local preference: chiede ad un peer di preferire valori più alti di local preference;
  - atomic aggregate;
- Attributi opzionali:
  - multi-exit discriminator: non transitivo, chiede all'AS a cui ci si rivolge di preferire valori più bassi di questo attributo;
  - aggregator: transitivo;
  - community: transitivo.

### As-Path

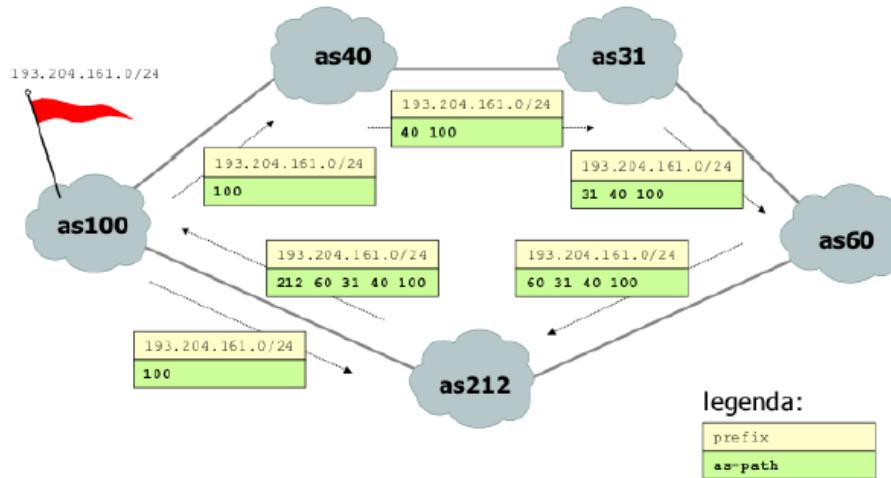


Figura 188 - as-path

AS100 ottiene un prefisso e lo annuncia ai suoi vicini. Dentro l'annuncio c'è scritto 100, che è il cammino che l'annuncio ha fatto fino a quel momento (un passo del cammino è l'intero AS). AS40, che riceve l'annuncio può raccontare questa storia ad AS31, e nell'annuncio ci sarà scritto 40 100. AS31 adesso annuncia ad AS60, quindi nell'annuncio ci sarà scritto 31 40 100, e così via. L'annuncio poi tornerà ad AS100. Si capiscono le due cose che si fanno con as-path. In primo luogo si evitano cicli: quando AS100 riceve un annuncio in cui nell'as-path c'è già il suo numero lo prende e lo butta, perché vuol dire che quell'annuncio gli è già passato davanti. Seconda cosa: ottimizzazione. Guardiamo AS212: riceve 193.204.161.0/24 da AS60 e da AS100; ha di fronte due annunci con due diversi as-path e può, se vuole,

scegliere il più breve. È un'ottimizzazione un po' buffa: la rete interna ad un AS può essere piccola piccola o gigantesca, non si sa, quando si passa dentro un AS, quello che ci sta, però questo è il prezzo da pagare al livello di astrazione molto alto che ha il routing in questo momento. Questa operazione di aggiungere al passaggio dentro un AS il numero dell'AS si chiama in gergo prepending, perché quel numero viene messo all'inizio dell'AS path. In questo modo posso anche applicare policy, posso osservare che il mio traffico per raggiungere una destinazione passa per un AS che io considero inaffidabile, ostile, pericoloso, allora tra le possibili scelte quella non la prendo. In iBGP l'as-path non cambia. Se ci pensiamo, un path è un distance vector un po' più arricchito, in cui non c'è solo il costo, ma anche tutti gli AS che devono essere attraversati (per questo si chiama path vector come algoritmo).

### Next-hop

Specifica dove spedire pacchetti per un prefisso specifico. Normalmente il next-hop è il router che spedisce l'annuncio, con delle eccezioni: shared-media e iBGP. Vediamo meglio di che si tratta.

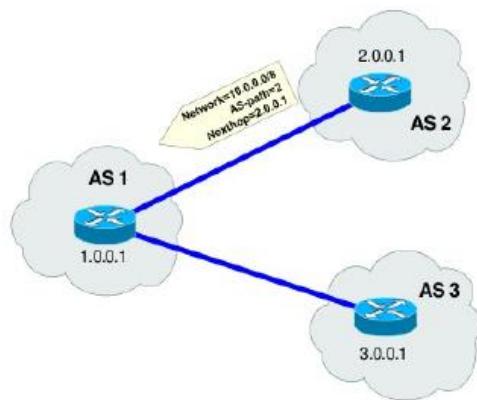
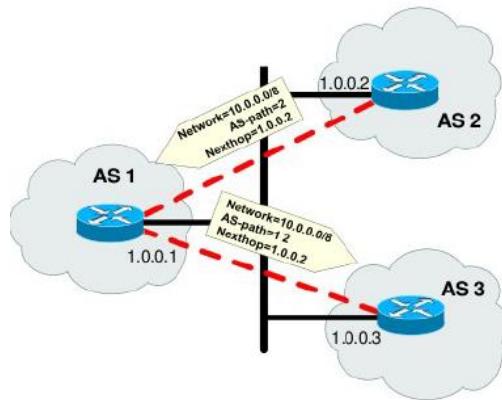
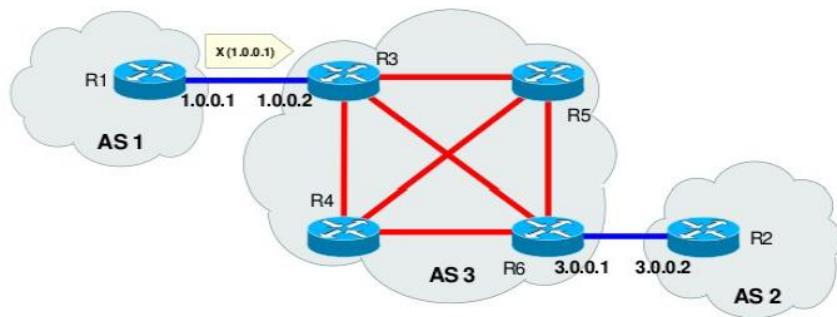


Figura 189 - Next-hop

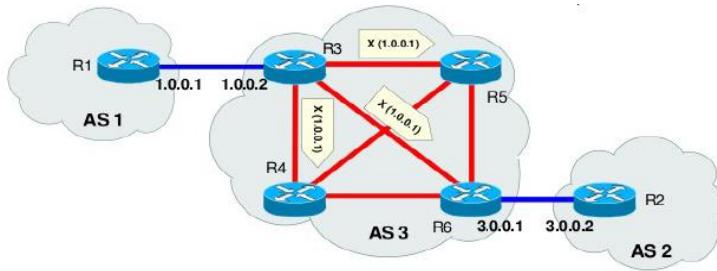
Qui c'è un annuncio della 10.0.0.0/8 (facciamo finta che non sia una rete privata) che parte da AS2. A questo punto quando si manda questo annuncio si specifica come as-path 2, e come next-hop 2.0.0.1, cioè il router che fa l'annuncio. Quando l'annuncio passa da as1 ad as3 non soltanto si modifica l'as-path facendo il prepending di 1, ma si modifica anche il next-hop. Questo è il comportamento standard del next-hop. Però ci sono delle eccezioni: shared-sequence, shared-media e iBGP.



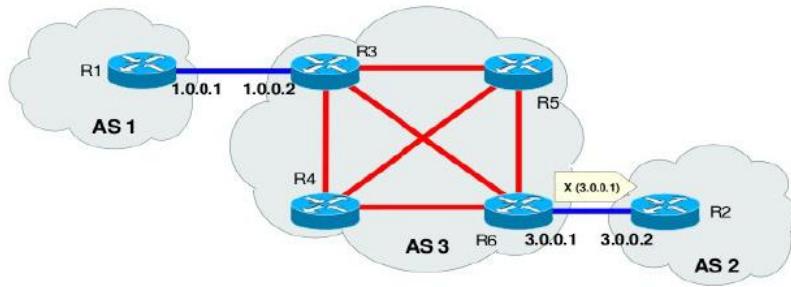
Qui stiamo su una LAN, AS2 annuncia la 10.0.0.0/8 ad AS1, il quale a sua volta lo annuncia ad AS3. Adesso, che dovrebbe fare questo annuncio? Dovrebbe cambiare il next-hop, in modo tale che il traffico che si vuole spedire vada su AS1 e poi su AS2, ma perché fare questo giro quando si sta sulla stessa LAN? Conviene andare direttamente su AS2. Se 3 router di 3 AS stanno sulla stessa LAN il next-hop non cambia. Quanto è realistica questa storia, cioè 3 AS che si incontrano su una LAN e fanno peering sulla LAN? Molto realista, questo è un internet exchange point (IXP). La situazione in iBGP è più difficile. Se una rottura che viene appresa da iBGP viene propagata utilizzando iBGP allora il next-hop rimane lo stesso, quindi in questo caso il next hop ha il significato di “next-hop del remote peer da cui hai preso la rottura”. Quello che fanno i router interni è seguire una cosa che si chiama recursive lookup, per verificare come quel next-hop possa essere raggiunto. Perché il recursive lookup funzioni naturalmente l'IGP deve funzionare correttamente. Guardiamo questa rete:



Qui c'è AS3 che ha vari router di frontiera, in full mesh iBGP tra loro (i collegamenti rossi non rappresentano necessariamente connettività diretta, lì dentro ci potrebbe essere un mondo). C'è un annuncio di un certo prefisso X da parte di R1, e viene specificato come next-hop 1.0.0.1; a questo punto questo annuncio attraversa l'AS, e il next-hop rimane invariato.



Quando l'annuncio si propagherà su AS2 a questo punto l'annuncio recupererà nuovamente l'interfaccia di R6.



Qual è il significato di questo passaggio in cui il next-hop non viene modificato? È un significato perfettamente consistente con la semantica di BGP: se AS3 è un nodo singolo, anche molto complesso, quello che interessa per recapitare il traffico è, quando il traffico ha attraversato l'AS, dove deve andare, e quindi il next-hop che viene conservato è 1.0.0.1. Il traffico però i router devono farcelo arrivare fino a quel prefisso. Cosa fa R5? Quando apprende un prefisso via iBGP, sa che il next-hop che sta dentro quell'annuncio non è un next-hop vero, ma è il next-hop di un router di bordo area di un altro AS, e come lo sa che il traffico dovrà passare all'interno dell'AS? Fa una verifica appena arriva l'annuncio, e verifica se quel next-hop è raggiungibile.

## Applicare policy

BGP permette di specificare politiche, ed in particolare per farlo può usare due strumenti: *announcement filtering* e *annoucement tuning*. Quando si fa filtering si può stabilire di fare certi annunci soltanto se certe condizioni sono verificate. Il comportamento standard di BGP è far girare annunci relativi a dei prefissi; posso prendere alcuni di questi prefissi e filtrarli, buttarli via, o ancora, conosco un certo prefisso ma non te lo dico (magari per motivi economici o di competizione). Posso filtrare anche degli AS. Un'altra cosa che si può fare è prendere un annuncio e manipolarlo. Abbiamo detto che dentro un annuncio BGP c'è un path, che viene costruito durante il percorso. Io posso manipolare questo path: c'è un 10? Lo faccio diventare un 28, o sostituirlo con 10 10. Per manipolare un annuncio, in linea di principio nel modo più completo possibile, si usano i comandi route-map. Nella tabella di instradamento di BGP ci sono campi come metric, local-preference, che io posso prendere e manipolare, condizionando in questo modo i

meccanismi di scelta che sono eseguiti da coloro che ricevono gli annunci. Un'altra cosa che posso fare è applicare una prefix-list in ingresso o in uscita per un certo neighbor.

---

```
neighbor <neighbor-ip> prefix-list <p-list-name> in  
neighbor <neighbor-ip> prefix-list <p-list-name> out
```

---

Le prefix list si specificano in questo modo:

```
ip prefix-list <p-list-name> permit <network/mask>  
ip prefix-list <p-list-name> deny <network/mask>
```

---

Posso permettere oppure impedire che un prefisso passi, e posso farlo in ingresso (filtro gli annunci in ingresso, alcuni li prendo altri li ignoro) oppure in uscita (filtro quelli in uscita, a questo questa cosa la dico, a quest'altro no). Quando si vede in e out bisogna sempre pensare così: in BGP se arriva un annuncio in ingresso, se io metto una prefix-list su quell'annuncio in ingresso, ciò che sto condizionando è il traffico in uscita (gli annunci vanno nella direzione opposta rispetto al traffico). Questo:

---

```
router bgp 1  
network 195.11.14.0/24  
network 195.11.15.0/24  
neighbor 193.10.11.2 remote-as 2  
neighbor 193.10.11.2 description Router 2 of AS2  
neighbor 193.10.11.2 prefix-list partialOut out  
neighbor 193.10.11.2 prefix-list partialIn in  
!  
ip prefix-list partialOut permit 195.11.14.0/24  
!  
ip prefix-list partialIn deny 200.1.1.0/24  
ip prefix-list partialIn permit any
```

---

è un esempio di configurazione BGP in cui si filtrano gli annunci. Qui si specificano delle prefixlist; rispetto al vicino 193.10.11.2 applico la prefix-list partialOut in uscita e partialIn in ingresso. Le politiche sono specificate più sotto:

- *ip prefix-list partialOut permit 195.11.14.0/24*: questo vuol dire che rispetto a quel peer delle due network che ci sono io ne annuncio soltanto una. In assenza di una specifica di prefix list si annuncia tutto. Appena si specifica una cosa del tipo permit, se non si specifica un altro permit gli altri prefissi sono automaticamente bloccati.
- *ip prefix-list partialIn deny 200.1.1.0/24 - ip prefix-list partialIn permit any*: anche se il vicino mi annuncia la 200.1.1.0/24 io non me la prendo, magari perché non voglio avere traffico da quella

rete, o magari perché il traffico di quella rete mi costa di più. Con il permit any dopo il deny significa che tutto il resto è consentito.

La semantica delle prefix-list è un pochino più articolata di questo (è la tipica semantica da firewall).

#### *Attributi*

##### *Origin*

Può assumere tre valori:

- igp;
- egp;
- incomplete.

Se l'annuncio è costruito a partire da un prefisso che è installato dentro il router con il comando network allora l'attributo assume il valore igp. È strano: mettere igp in quell'attributo se il prefisso l'avessi ottenuto ridistribuendo l'IGP dentro BGP, invece quando si usa il comando network si fa esattamente l'opposto, però questo è lo standard. Il valore egp è assolutamente legacy, serve per specificare che l'attributo è iniettato dentro BGP da EGP, il vecchio protocollo di routing interdominio (che ormai non esiste più). Il valore incomplete viene usato per specificare che il prefisso è stato generato ridistribuendo dentro BGP un IGP, che è una cosa che si fa molto (molto) raramente, perché molto pericolosa. Tutte queste cose stanno scritte nel pacchetto BGP.

#### *Aggregator*

Questo specifica l'indirizzo IP del router (o BGP speaker) che ha generato la rotta aggregata. Che vuol dire "aggregata"? CIDR: si prendono un po' di prefissi, li si appiccicano in un prefisso meno specifico e le tabelle di instradamento si comprimono. Che senso ha comprimere le reti così? Quando si tirano fuori delle reti, o meglio, un lotto di prefissi da un prefisso meno specifico (ad esempio un prefisso del GARR, la rete degli Atenei italiani) non ha senso annunciare le differenti reti se si vuole far riferimento al "blocco", perché si metterebbe in giro un sacco di spazzatura, ritardando l'instradamento. Il router che si occupa di questa aggregazione nell'annuncio verso l'esterno è l'aggregator. Questa aggregazione, in linea di principio, su CIDR, dove avviene in internet, e come si comporta BGP rispetto a CIDR? Nel caso del GARR quello che succede è questo: hai un AS, che è l'AS del GARR (AS137) in cui ci sono tante /24 distribuite un po' in tutta Italia. In questo AS il GARR ha dei router di frontiera che sono in vari posti. Un router, invece di annunciare diverse /24 manda fuori una /15 (il prefisso originario, quello che è stato spezzettato). A questo punto il resto di internet raggiunge l'AS137 attraverso il router in questione. In questo caso questo router è l'aggregator di questa /15, è lui che si prende la responsabilità di fare CIDR. Però, CIDR dice che chiunque sia un router che scopre che si potrebbe fare un'aggregazione dovrebbe farlo. Questo in BGP è un grosso problema. L'as-path che fine fa? Noi sappiamo che se mandiamo una /25 nell'annuncio viene via via costruito l'as-path; se adesso qualcuno deve aggregare due /25 in una /24, una

viene da un AS e un'altra da un altro AS. Come è fatto il path? È evidente che tutto questo non è fatto apposta per un cammino, è fatta “più apposta” per realizzare un albero, ma dentro l'annuncio BGP spazio per gli alberi non c'è, quindi questo è un problema serio nell'uso di CIDR insieme a BGP. Quando si scopre un aggregator in un annuncio BGP nel 99% dei casi l'aggregator è un router di frontiera di un AS che aggrega soltanto ciò che c'è nell'AS; fare delle aggregazioni al volo, nel passaggio degli annunci attraverso router che sono in diversi AS è molto problematico, ed è quindi un'eventualità rara.

### Selezionare la rotta migliore

Come si seleziona la rotta migliore per raggiungere un prefisso? Abbiamo detto che un router per ciascun prefisso riceve vari annunci, e deve selezionare il migliore (gli annunci relativi allo stesso prefisso possono venire dalle direzioni più disparate). Il processo decisionale di BGP è complesso e completamente deterministico, non c'è nessuna scelta random (questo per motivi essenzialmente di debugging). Il router sceglie la rotta migliore per raggiungere un prefisso e l'annuncia (se ne ha voglia) ai suoi peer. I criteri di selezione sono questi:

- prefissi più specifici e prefissi meno specifici sono considerati prefissi differenti; supponiamo che il GARR sia impazzito, e che in quell'AS137 invece che annunciare soltanto la /15 si annuncia sia la /15 e una /24 (gli è sfuggita e annuncia pure quella). Per BGP quei due prefissi sono completamente diversi, vengono trattati in modo completamente diverso, e finiscono tutti e due in tabella. Quello che si potrebbe pensare è che se BGP riceve una /15 e una /24 contenuta nella /15 potrebbe ignorare la /24, ma non è così. Questo in termini di data plane significa che quei prefissi finiscono tutti e due nella tabella di instradamento IP, e questo ha delle conseguenze notevoli, perché IP fa sempre best prefix match, quindi farà sempre match prima sulla /24 e poi sulla /15. Questo implica che se per caso sono disponibili due strade diverse per raggiungere per qualche motivo la /24 e la /15 i pacchetti per la /24 sceglierebbero sempre la strada più specifica, che è quella della /24. Su questo si possono fare dei trucchi notevoli.
- se il next-hop non è raggiungibile (fallisce il recursive lookup) l'annuncio non può essere selezionato.

### Il processo decisionale

Per selezionare un prefisso BGP sceglie, in ordine di preferenza, in base a:

1. Largest weight: peso più alto (proprietario Cisco);
2. largest local preference: local preference più alta;
3. locally originated: tra due annunci, di cui uno è locale, scegli quello;
4. shortest as-path length: lunghezza più breve dell'as-path;
5. lowest origin (IGP < EGP < unknown);
6. multi exit discriminator più basso possibile;
7. preferire eBGP su iBGP (hot potato routing);

8. lowest IGP metric (per raggiungere il next-hop); a parità di ogni altra cosa conviene mandare il traffico sul router di frontiera che si raggiunge con la metrifica interna migliore (significa utilizzare meno le risorse interne);
9. scegli in funzione del router-id (l'identificare del router che ha mandato l'annuncio).

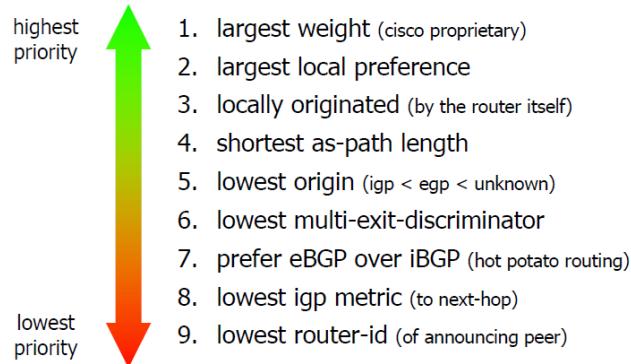


Figura 190 - Processo decisionale

Perché dovremmo preferire eBGP su iBGP? Cioè, tra due annunci relativi allo stesso prefisso, uno appreso dall'esterno dia eBGP, e uno appreso dall'interno, da un altro router di frontiera, si preferisce l'annuncio ricevuto dall'esterno, perché? Quando ricevi un annuncio, se scegli quell'annuncio rispetto ad un altro scegli una direttrice per il traffico che spedisci; conviene buttarlo subito fuori il traffico oppure fargli attraversare l'AS (il che si paga in termini di infrastrutture)? Te ne liberi subito. Il processo decisionale avviene in questo modo:

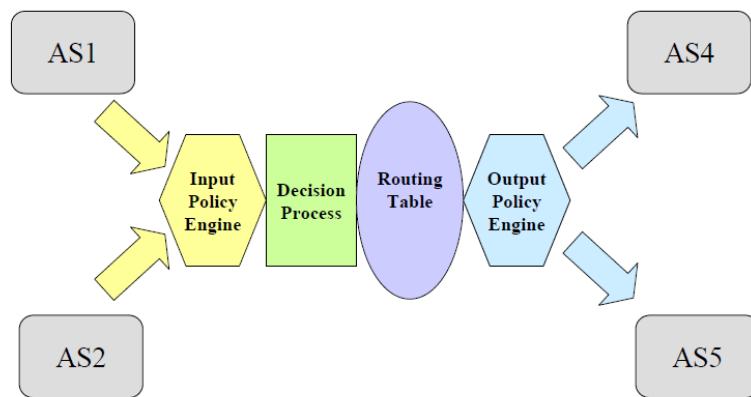


Figura 191 - Architettura del processo decisionale

Gli annunci che arrivano da diversi router di diversi AS vengono trattati dall'input policy engine. Questo IPE cosa fa di questi annunci? “Questo lo prendo e lo butto, perché c'è una prefix list. Questo me lo tengo. Questo me lo tengo e lo modifco”. La modifica degli attributi può avere senso per ingannare il decision process. Degli annunci ricevuti dall'IPE relativi ad un certo prefisso, diciamo 10, ne scartiamo magari 4. Dei restanti 6 bisogna sceglierne uno in base al processo decisionale, di cui ci sono vari step. Si possono manipolare gli annunci in modo tale che un certo annuncio, sfavorito fino a quel momento, da un certo

punto in poi sia favorito. C'è poi il processo decisionale, e a questo punto un solo annuncio di quelli relativi ad un certo prefisso (che possono essere 10, 6 o quanti sono) viene scritto nella tabella di instradamento di BGP. Nella tabella a questo punto c'è una best, la rotta migliore, che viene presa e data in pasto all'output policy engine, che prende la rotta e fa "a questo gliela dico, a questo no, a questo gliela dico ma gli cambio un po' di attributi, in modo di modificare io il suo processo decisionale".

## Modifica degli attributi

Per modificare gli attributi si usa il comando route-map, che può essere specificato in ingresso e in uscita:

---

```
neighbor <neighbor-ip> route-map <r-map-name> in
neighbor <neighbor-ip> route-map <r-map-name> out
route-map <r-map-name> permit <seq-number>
match <announce-property>
set <attribute-setting>
...
command syntax
route-map <r-map-name> deny <seq-number>
match <announce-property>
set <attribute-setting>
...
```

---

Un esempio:

---

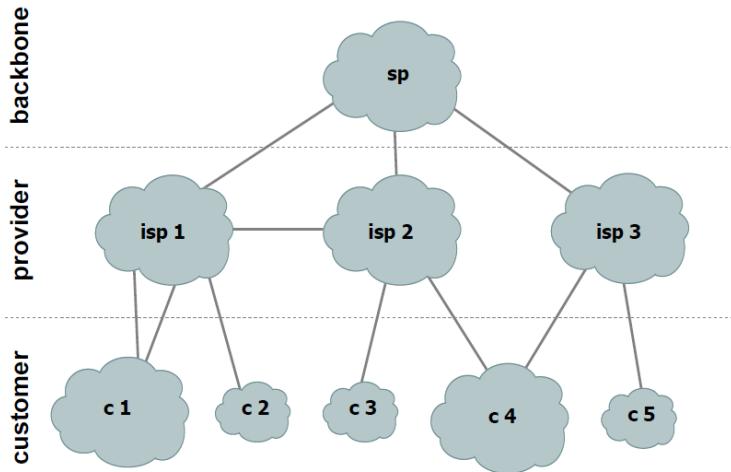
```
router bgp 100
network 100.1.1.0/24
neighbor 222.2.2.2 remote-as 200
neighbor 222.2.2.2 route-map myRouteMap in
!
route-map myRouteMap permit 10
match ip address myAccessList
set metric 5
set local-preference 25
!
route-map myRouteMap permit 20
set metric 2
!
access-list myAccessList permit 193.204.0.0/16
```

---

Qui, rispetto al neighbor 222.2.2.2 si specifica la route-map chiamata myRouteMap. Quei permit 10, permit 20 specificano che queste azioni devono essere fatte in questa sequenza (prima permit 10, poi permit 20). Lì si possono specificare delle etichette numeriche qualunque (tipo Basic). L'operatività della route map è questa: myRouteMap è una route map in ingresso; si fa un match sull'indirizzo IP relativamente alla lista myAccessList (specificata più in basso), che indica un prefisso (o una lista di

prefissi). La semantica complessiva è: fai un match su quella /16. Se l'annuncio in ingresso è relativo a quella /16 imposta la metrica a 5 e la local preference a 25.

## Una piccola rete internet



*Figura 192 - Una piccola Internet*

Ci sono degli AS che possiamo pensare siano dei customer, degli AS di basso livello sulla rete. Cosa fanno questi AS? Fanno diversi mestieri; ci sono quelli che hanno il compito di accogliere utenza domestica, quindi vendere accessi a internet, sotto diverse forme. Questi sono AS di basso livello, nel senso che il loro mestiere è semplicemente raccogliere traffico dei clienti, recapitarlo, e recapitare ai clienti traffico da internet. Ci sono poi attori come le CDN, che hanno del traffico che deve essere erogato a chi ne fa richiesta. Questi customer se vogliono trasportare traffico attraverso la rete non è che possono fare peering con tutti gli altri AS dello stesso livello, quindi quello che fanno è rivolgersi a dei mediatori, degli AS di più alto livello che spostano traffico da una parte del mondo ad un'altra. È chiaro che questi due livelli non sono veramente disgiunti, c'è anche chi fa due mestieri contemporaneamente (raccolta traffico e provider di transito), quindi questa gerarchia è un po' idealizzata. Sopra questi provider poi c'è quello che si chiama il backbone di internet, chi offre transito a chi offre transito, e chi offre transito a chi offre transito a chi offre transito, n volte.

### Classificazione dei customer

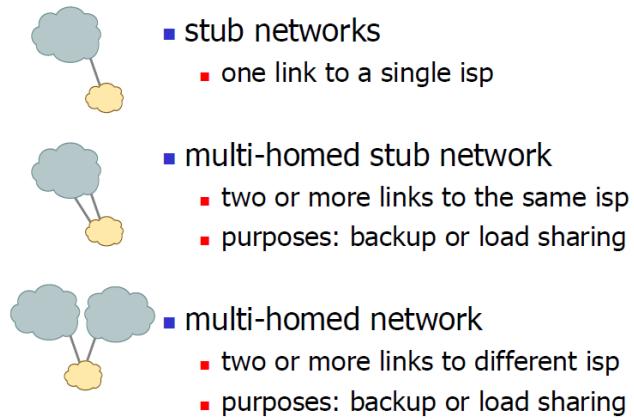


Figura 193 - Classificazione customer

Stub. Questo è un tizio che ha un AS e che rivolge la propria attenzione ad un altro AS, un ISP, chiedendogli l'accesso a internet. Verso lo stesso ISP di prima poi posso avere non soltanto un peering, ma più d'uno, per motivi di backup o di bilanciamento di carico. Invece di avere poi soltanto un AS che offre transito verso internet ne ho due; questo aumenta la mia resistenza ai guasti perché qui se l'intero provider dovesse non funzionare io continuerei a raggiungere internet. Rivolgandomi a più parti posso anche fare bilanciamento di carico in maniera più articolata: posso direzionare il traffico in funzione di quanto pago per un certo tipo di traffico.

### Stub

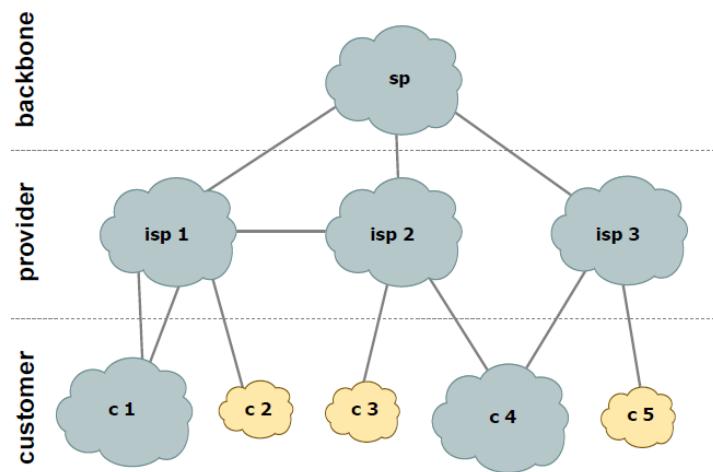
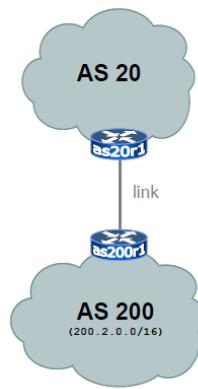


Figura 194 - Stub networks

Un solo link ad un singolo ISP. Nella situazione di prima stiamo guardando c2, c3 e c5. Questa situazione attualmente non è molto frequente: se qualcuno “taglia il filo” sei fuori gioco.

Qualche volta questi provider che hai “sopra” si chiamano upstream, e la situazione potrebbe essere questa:



*Figura 195 - Stub*

Io sono andato dal RIPE e mi sono fatto dare un numero di AS (200). Io uso la /16 assegnata per fare diverse cose. Uno dei miei router fa da default gateway per l'esterno, e scelgo un router del mio upstream provider per farci un peering, su cui AS200 annuncia la sua rottura. Per raggiungere il resto di internet non è ragionevole che questo AS a cui sono rivolto mi annunci tutti i prefissi. Visto che ho una sola possibilità d'uscita è sufficiente che l'upstream mi annunci solo la rottura di default, la o/o. Vediamo come può essere fatta la configurazione dei router:

---

```

! router as200r1 (customer side)
!
router bgp 200
network 200.2.0.0/16
neighbor 11.0.0.34 remote-as 20
neighbor 11.0.0.34 description Router as20r1

! router as20r1 (isp side)
router bgp 20
network 20.1.1.0/24
network 0.0.0.0/0
neighbor 11.0.0.33 remote-as 200
neighbor 11.0.0.33 description Router as200r1
neighbor 11.0.0.33 default-originating
neighbor 11.0.0.33 prefix-list customerIn in
neighbor 11.0.0.33 prefix-list defaultOut out
!
ip prefix-list customerIn permit 200.2.0.0/16
ip prefix-list defaultOut permit 0.0.0.0/0

```

---

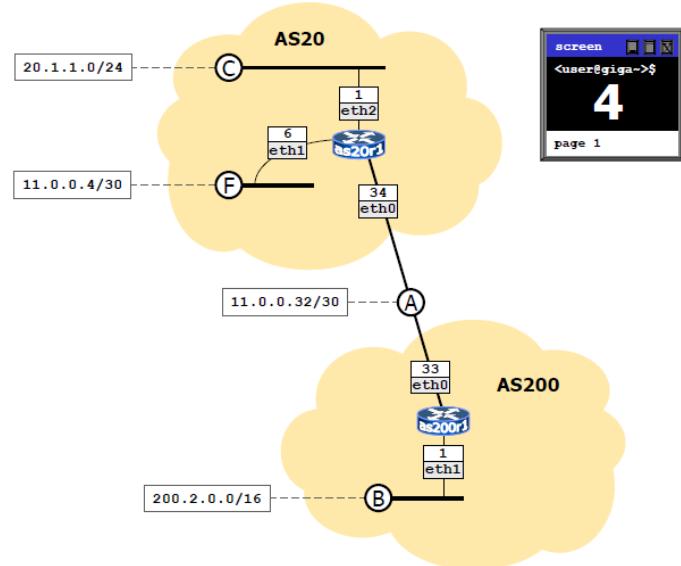


Figura 196 - Stub

La semantica di *network o.o.o.o/o* e di *default-originate* è leggermente diversa. La prima dice “la o.o.o.o/o è mia”, la seconda dice “io sono la tua rotta di default”. In altri termini, la o/o viene comunque annunciata, ma se c'è solo *default-originate* la o/o non viene messa nella tabella di instradamento di BGP. Qui non era necessario scrivere tutte e due le cose. Il significato di *neighbor 11.0.0.33 prefix-list customerIn in* è questo: tu sei in accordo commerciale con un customer, e il customer annuncia quella /16. Se questo customer se ne volesse approfittare, annunciando qualche altro prefisso non contrattualizzato, lo blocco, perché con *permit 200.2.0.0/16* io ti consento di annunciarmi solo quella /16, nient'altro. Nella direzione opposta c'è *neighbor 11.0.0.33 prefix-list defaultOut out*, e *ip prefix-list defaultOut permit o.o.o.o/o*. Questo vuol dire che al customer si annuncia soltanto la o/o (se c'è una prefix-list che annuncia soltanto un prefisso vuol dire che tutto il resto è bloccato). Questa è anche una protezione verso il customer, che non ha nessun interesse né vantaggio a vedersi annunciare i prefissi dell'intera internet. In questo caso, aver ottenuto un numero di AS dal RIPE significa essere stati molto convincenti, perché qui BGP è superfluo: bastava dire ad *as20or1* di mandare tutto il traffico relativo alla 200.2.0.0/16 su *as20or1*, e su *as20or1* era sufficiente mettere una rotta di default verso *as20or1* per tutto il traffico che non è locale all'AS.

Multi-homed stub network: two links to the same isp

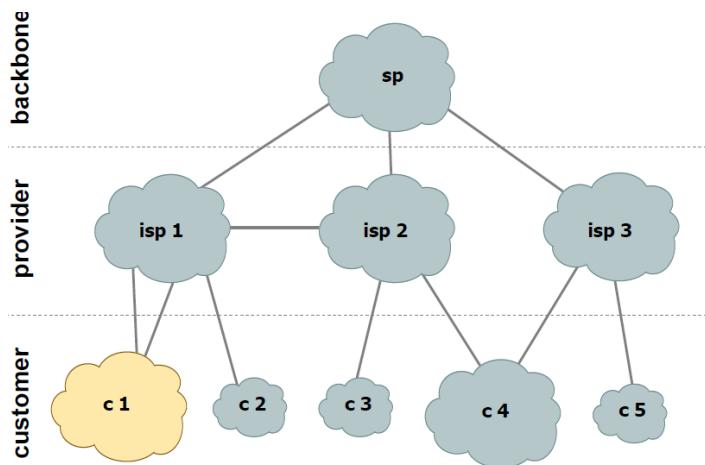


Figura 197 - Multi-homed stub network

Qui siamo nella situazione del customer  $c_1$ , che ha due peering con  $isp_1$ . Qui BGP comincia ad essere utile. La situazione in cui ci mettiamo è quella in cui lato  $c_1$  c'è un solo router che ha due peering verso due router diversi di  $isp_1$ . Questa situazione è una situazione realistica, ma non troppo. Se vuoi avere resistenza ai guasti la cosa migliore è utilizzare due router di frontiera, e non uno solo, perché significherebbe avere un single point of failure; quel router potrebbe rompersi e questo ti escluderebbe dal resto di internet. Se poi paghi per due link fisici, che sono molto costosi, è ragionevole che tu possa pagare anche per due router. Posso riuscire ad evitare l'uso di BGP in questa situazione? Voglio configurare il link 1 come primario, e il link 2 come link di backup. Posso farlo con OSPF o RIP, o delle rotte statiche? Con le rotte statiche proprio non ce la posso fare, dovremmo annunciare la /16 su tutti e due i link, ma se va giù uno dei link chi glielo dice ad  $isp_1$ ? Potrei mettere RIP su tutte e 3 le macchine, oppure OSPF con costi molto bassi sul link 1, in modo da utilizzarlo come link primario. Faccio poi girare lo stesso protocollo di routing su tutti e due gli AS (cosa poco realistica). Se OSPF me lo permette, una parte del traffico potrebbe fare questo percorso:

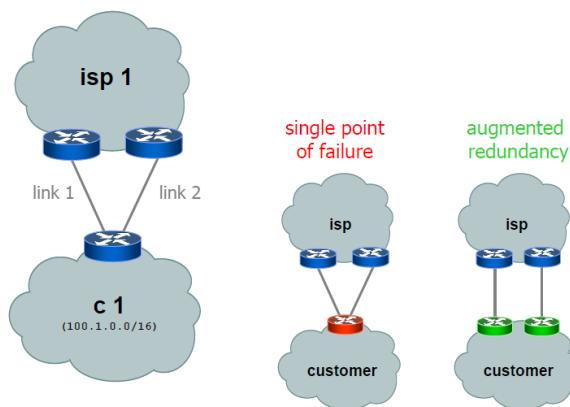
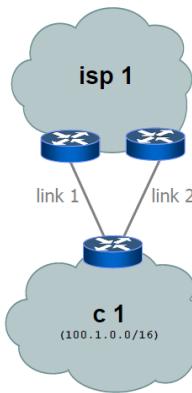


Figura 198 - Multi-homed stub networks

Questa cosa il customer non la vuole, ha acquistato quei due link per portare i propri pacchetti verso internet e viceversa, lui non vuole che **isp1** possa sfruttare le sue stesse infrastrutture e il suo router per far fare traffico ai suoi clienti. Cominciamo adesso ad apprezzare la parola “politica” in BGP. La politica principale di **c1** è non comprare infrastrutture per offrire un servizio a chi il servizio glielo deve offrire. Qui la necessità di BGP è palese, e non posso esprimere questa politica con RIP e OSPF.



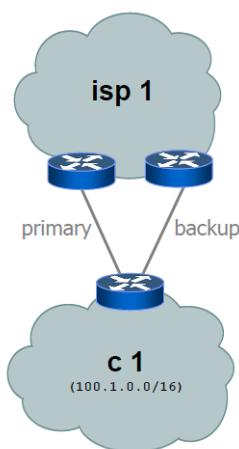
*Figura 199 -Funzionamento del BGP*

Vorremmo quindi che fosse realizzata questa politica:

- link 1 è primario;
- link 2 è di backup;
- non vogliamo flusso di transito per **isp1**;
- per il traffico in ingresso vogliamo usare il link 1, e il link 2 solo quando il link 1 non è disponibile;
- per il traffico in uscita vogliamo usare il link 1, e il link 2 solo quando il link 1 non è disponibile.

Questa è una cosa che con BGP va fatta sempre:

- cosa succede per il traffico in ingresso;
- cosa succede per il traffico in uscita.



*Figura 200 - Uso del BGP*

Per realizzare una politica primario-backup devo in qualche modo usare gli attributi di BGP. Sul link primario posso fare degli annunci standard, e sul link di backup incremento la metrica per gli annunci in uscita e riduco la local preference per il traffico in ingresso. In questo modo tutto il traffico passerà preferibilmente per il link primario.

---

```
! router as100r1 (primary, customer side)
!
router bgp 100
network 100.1.0.0/16
!
neighbor 11.0.0.2 remote-as 20
neighbor 11.0.0.2 description Router as2or2 (primary)
neighbor 11.0.0.2 prefix-list mineOutOnly out
neighbor 11.0.0.2 prefix-list defaultIn in
!
!
neighbor 11.0.0.6 remote-as 20
neighbor 11.0.0.6 description Router as2or1 (backup)
neighbor 11.0.0.6 prefix-list mineOutOnly out
neighbor 11.0.0.6 route-map metricOut out
neighbor 11.0.0.6 prefix-list defaultIn in
neighbor 11.0.0.6 route-map localPrefIn in
!
ip prefix-list mineOutOnly permit 100.1.0.0/16
!
ip prefix-list defaultIn permit 0.0.0.0/0
!
route-map metricOut permit 10
match ip address myAggregate
set metric 10
!
route-map localPrefIn permit 10
set local-preference 90
!
access-list myAggregate permit 100.1.0.0/16
```

---

Per ciò che riguarda gli annunci in uscita quindi si incrementa l'attributo metrica rispetto al default. Questo vuol dire che gli annunci in uscita hanno una metrica da una parte, e un'altra metrica da un'altra. Il comportamento di default di isp1 è di preferire gli annunci in ingresso che hanno una metrica più bassa. Quando un AS riceve due annunci con due metriche diverse preferisce la metrica più bassa, però l'AS non è che decide, l'AS non è niente dal punto di vista tecnologico, è piuttosto l'insieme dei suoi router di frontiera. Questo ci fa capire per la prima volta l'importanza di iBGP. Guardiamo cosa succederà là dentro. Tutti i router di frontiera devono fare peering iBGP in full mesh, quindi quei router si parlano. Il fatto che i router si parlano significa che ad un certo punto si diranno "io ho ricevuto questo annuncio con una

metrica bassa" - "io ho ricevuto questo annuncio con una metrica alta" - "sono relativi allo stesso prefisso" - "scegliamo quello con la metrica bassa". Quindi in realtà non è l'AS che decide, ma sono i router di frontiera dell'AS che lo fanno. L'attributo metric ha un'importante caratteristica: quando usi l'attributo metric, e lo modifichi, questo metric fa riferimento soltanto al rapporto che hai con uno specifico AS, e non con più AS. In altre parole, l'attributo metric non è transitivo, il confronto sulla metrica si ha soltanto per annunci verso lo stesso AS, e se sono relativi allo stesso prefisso. Quando metric viene ricevuto dall'altra parte si chiama, dal punto di vista del provider, multi-exit discriminator, e non a caso: l'ISP ha più uscite, e seleziona l'uscita che ha la metrica più bassa.

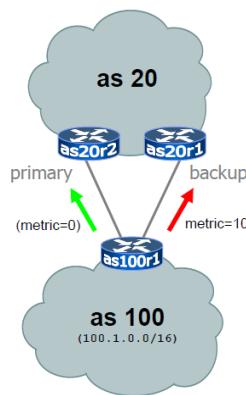


Figura 201 - Atributo metrica

La seconda parte della politica primario-backup riguarda il traffico in uscita da c1. Qui facciamo un'altra cosa: riduciamo la local preference su annunci in ingresso lato backup. Si sceglie la local preference più alta, quindi tra due annunci che arrivano, uno sul primario e uno sul backup, relativi allo stesso prefisso, scelgo quello che arriva sul primario. Ho "imbrogliato" il mio stesso processo decisionale, e in questo modo riesco a regolare primary-backup il traffico in uscita. In assenza poi degli annunci che hanno la metrica più bassa e la local preference più alta BGP userà il link di backup, e abbiamo quindi realizzato una politica primario-backup.

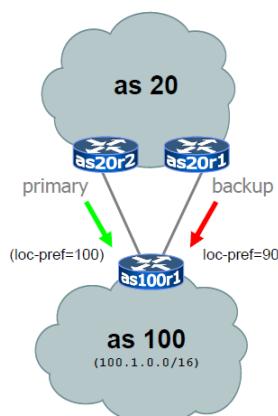


Figura 202 - Attributo local-preference

### Multi-homed stub network: two links to two different isp

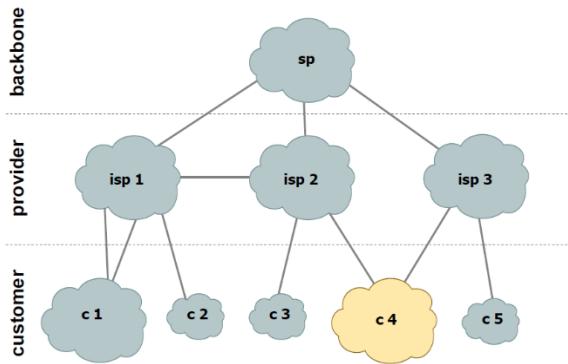


Figura 203 - Multi-homed stub network: two links to two different isp

Siamo su c4; questa è una soluzione assolutamente realistica, un multihoming verso due provider. Questo customer fa peering con due diversi provider. Due link a due provider differenti, e la cosa migliore è utilizzare due router come router di frontiera invece che uno, per evitare un single point of failure.

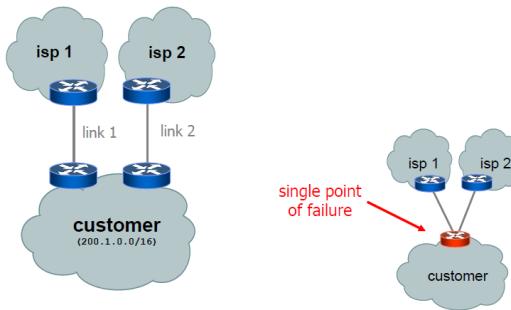


Figura 204 - Multi-homed network

Quali sono i gradi di libertà? Un pacchetto in uscita può essere spedito indifferentemente su uno dei due link per raggiungere il resto di internet.

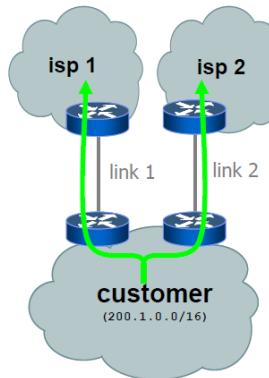


Figura 205 - Primo grado di libertà

Altro grado di libertà: un pacchetto in entrata può usare indifferentemente uno dei due link per raggiungere c4.

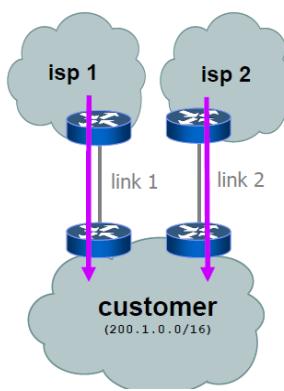


Figura 206 - Secondo grado di libertà

Quello che ci dà veramente fastidio è questo:

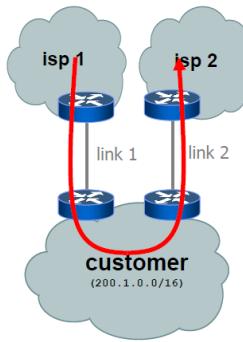


Figura 207 - Terzo grado di libertà

Un pacchetto di internet (dell'intera internet) attraversi i link 1 e 2.

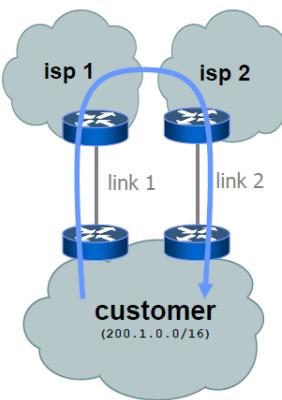


Figura 208 - Quarto grado di libertà

La cosa un po' più strana che potrebbe capitare è che un pacchetto che è puramente locale attraversi link 1 e link 2, rimbalzando all'esterno per arrivare fino a noi; è vero che questo non è un dramma, perché paghiamo per traffico di questo genere, ma è anche vero che normalmente si paga anche in funzione del traffico che si manda su questi link, quindi se si riesce a non utilizzarli è meglio. In questo caso utilizziamo una politica di load sharing, non utilizziamo i link come primario e come backup.

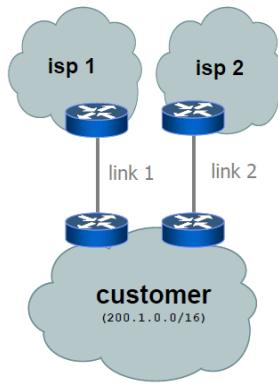


Figura 209 - Loadsharing

In particolare facciamo questo: tagliamo via tutto il traffico di transito, perché quello proprio non ci piace. Per ciò che riguarda il traffico in uscita potremmo far utilizzare ad una metà degli host interni il link 1, e all'altra metà il link 2. Per ciò che riguarda il traffico in ingresso potremmo pensare di fare la stessa cosa. In realtà però questa idea di load sharing va bene, ma solo entro certi limiti, dobbiamo comunque cercare di fare backup. Per fare questo bilanciamento di carico annunciamo la /16 aggregata su tutti e due i link, e poi facciamo questo trucco: tagliamo in due la /16 e la facciamo diventare due /17; di queste /17 una la annunciamo sul link 1, e l'altra sul link 2. Questo vuol dire che se le macchine all'interno sono distribuite in maniera abbastanza uniforme tra le due porzioni della /16 gli annunci in uscita porteranno traffico verso i router dei due ISP in maniera abbastanza distribuita. Questo chiaramente è un load sharing abbastanza approssimativo. Se vogliamo essere più bravi, e non abbiamo link di uguale capacità possiamo modificare questo split: posso ad esempio fare quattro /18, per poi annunciarne tre da una parte e una dall'altra. Queste divisioni le posso fare a tavolino o anche sperimentalmente, vista la capacità di BGP di modificare in tempo reale gli annunci. Che cosa succede nel verso opposto? Io posso accettare la default dall'upstream (da tutti e due gli upstream) e il bilanciamento di traffico in uscita lo faccio utilizzando l'IGP, e le macchine sceglieranno il punto d'uscita più vicino (meccanismo nearest exit). In altri termini: mi faccio annunciare la o/o dall'upstream, e la propago nell'IGP attraverso tutti e due i punti d'uscita. A questo punto la o/o arriverà da un punto oppure da un altro in funzione del fatto che tu sia più vicino ad un punto d'uscita piuttosto che ad un altro, e questo realizza sostanzialmente un bilanciamento di carico.

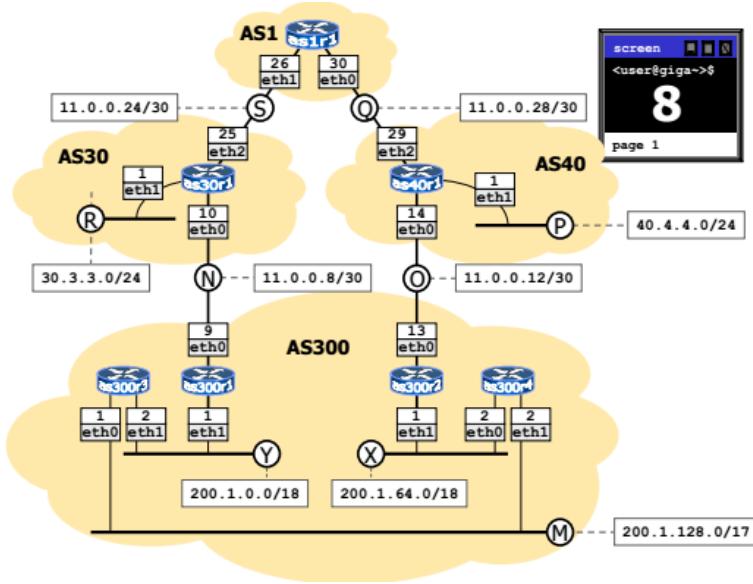


Figura 210 - Rete di esempio

Configurazione di AS300r1:

---

```

!as300r1
router bgp 300
network 200.1.0.0/16
network 200.1.0.0/17
!
neighbor 11.0.0.10 remote-as 30
neighbor 11.0.0.10 description Router as30r1
neighbor 11.0.0.10 prefix-list mineOutOnly out
neighbor 11.0.0.10 prefix-list defaultIn in
!
ip prefix-list mineOutOnly permit 200.1.0.0/16
ip prefix-list mineOutOnly permit 200.1.0.0/17
ip prefix-list defaultIn permit 0.0.0.0/0

```

---



---

```

!as300r2
router bgp 300
network 200.1.0.0/16
network 200.1.128.0/17
!
neighbor 11.0.0.14 remote-as 40
neighbor 11.0.0.14 description Router as40r1
neighbor 11.0.0.14 prefix-list mineOutOnly out
neighbor 11.0.0.14 prefix-list defaultIn in
!
ip prefix-list mineOutOnly permit 200.1.0.0/16

```

---

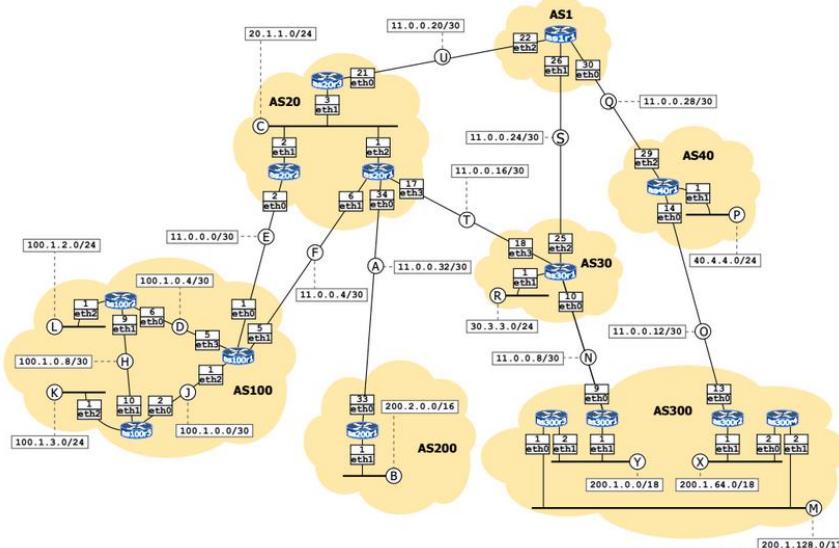
---

```
ip prefix-list mineOutOnly permit 200.1.128.0/17
ip prefix-list defaultIn permit 0.0.0.0/0
```

---

Per capire meglio come tutto funziona bisogna ricordare come funziona BGP per prefissi più specifici e meno specifici. Se annuncia una /16 e una /17 più specifica BGP considera questi due prefissi come completamente indipendenti l'uno dall'altro, li prende e li tratta separatamente, quindi tutti e due finiscono in tabella BGP. Quindi adesso succede che as300r1 e as300r2 propagano le due /17 e la /16 verso l'alto, verso internet, e tutti i router le mettono in tabella BGP; BGP poi parla con il data plane, la /16 e le due /17 finiscono tutte nel data plane. Che cosa succede poi quando un router spedisce un pacchetto? Fa best prefix match: cerca prima di parlare sulle più specifiche, e poi sulle meno specifiche. Questo vuol dire che se dentro la tabella c'è la /17 la /17 prevale. A questo punto se hai vari provider puoi fare questo: a tutti quanti dai un prefisso intero, e a ciascuna delle tue macchine ne dai un pezzettino; per raggiungere quel pezzettino prendi normalmente quella strada, se quella strada non è disponibile allora vai sul meno specifico. Le tabelle di instradamento più alte di internet, quelle della default free zone, sono piene di prefissi più specifici e meno specifici, che nascono da giochi di questo tipo.

#### Piccola rete internet



Abbiamo visto cosa può fare AS300 per il traffico in uscita. Per il traffico in ingresso, non potendo usare multi exit discriminator facciamo un altro trucco: utilizziamo il prepending. AS300 vuole essere raggiunto dal traffico in ingresso da AS40 (link O) e, solo se non c'è altra possibilità, da AS30. Cosa fa allora AS300? Negli annunci che fa sul peering con AS30 scrive 300 non una volta sola, ma tre, in modo tale che quell'annuncio parte già svantaggiato. Con il prepending si possono fare delle politiche molto raffinate: si

può mettere un prepending pazzesco in uscita, e questo fa sì che nessuno selezioni quell'annuncio, a meno che proprio non ci sia altra scelta. Si può anche studiare il traffico con un certo valore di prepending e usare il prepending come “rubinetto”, per far sì che magari solo una piccola parte di internet in condizioni normali ti raggiunga, oppure una parte un po' più grande, e questo lo sceglieremo mettendo un prepending più o meno significativo degli annunci che si fanno. Questo trucco lo vediamo in azione nella configurazione di as2or1:

---

```
router bgp 20
network 20.1.1.0/24
network 11.0.0.4/30
network 11.0.0.16/30
network 11.0.0.32/30
!
neighbor 11.0.0.33 remote-as 200
neighbor 11.0.0.33 description Router as2or1
    neighbor 11.0.0.33 default-originate
    neighbor 11.0.0.33 prefix-list as200In in
    neighbor 11.0.0.33 prefix-list defaultOut out
    !
    neighbor 11.0.0.5 remote-as 100
    neighbor 11.0.0.5 description Router as1or1
        neighbor 11.0.0.5 default-originate
        neighbor 11.0.0.5 prefix-list as100In in
        neighbor 11.0.0.5 prefix-list defaultOut out
        !
    neighbor 20.1.1.2 remote-as 20
neighbor 20.1.1.2 description Router as2or2 (iBGP)
!
neighbor 20.1.1.3 remote-as 20
neighbor 20.1.1.3 description Router as2or3 (iBGP)
!
neighbor 11.0.0.18 remote-as 30
neighbor 11.0.0.18 description Router as3or1 (eBGP)
neighbor 11.0.0.18 default-originate route-map dontUseMe
    neighbor 11.0.0.18 route-map dontUseMe out
    neighbor 11.0.0.18 prefix-list defaultOut out
    !
    ip prefix-list as200In permit 200.2.0.0/16
    ip prefix-list as100In permit 100.1.0.0/16
    ip prefix-list defaultOut permit 0.0.0.0/0
    !
    route-map dontUseMe permit 10
        set as-path prepend 20 20 20
```

---

C'è una route map con un nome molto evocativo: *dontUseMe*. L'annuncio che viene trattato con questa route map è un annuncio in uscita dal router, e quello che si fa è stampare 20 20 20 su quell'annuncio, in modo tale che chi riceve quell'annuncio lo usi solo se non ha altra alternativa, nessun altro annuncio che arriva per quel prefisso.

## Scalabilità di BGP

Best practices per rendere BGP scalabile.

### Route refresh

Ogni volta che si fa un cambiamento di una politica di un router BGP si dovrebbero resettare i peering e far ripartire completamente i peer, e questo perché il router non memorizza i prefissi che le politiche hanno rigettato. Quando un annuncio è filtrato viene buttato via, e non se ne conservano notizie nello stato del router; una variazione di politica potrebbe implicare che un annuncio che prima veniva scartato ora non lo è. In linea di principio quindi, poiché non si sa con esattezza cosa succederà col cambio di politica, sarà necessario far partire un peering, ma non è cosa da poco: se dentro quel peering ci passa la full routing table di internet quello è un diluvio di informazioni; il peering poi consuma CPU e questo ha un effetto negativo sulle prestazioni della rete. Per questo nasce la tecnologia soft BGP peer reset. Si può quindi fare restart solo relativamente ai prefissi impattati dal policy change, e ce lo possiamo permettere perché in BGP ciascun prefisso è trattato indipendentemente dagli altri. Per la maggior parte delle implementazioni non c'è necessità di modifiche della configurazione, e questa feature è negoziata automaticamente quando si stabilisce il peering; naturalmente per poterla negoziare i router devono supportare la route refresh capability (RFC2918). È vero che ciascun prefisso è trattato in modo indipendente da ciascun altro prefisso, ma è anche vero che ci sono comunque alcuni comandi di alcuni vendor che consentono di fare statement del tipo: se succede qualcosa su questo prefisso allora agisci su quest'altro prefisso. Un rapporto tra i prefissi può creare qualche problema nelle politiche.

### Next-Hop

Sappiamo che di default in BGP il next-hop deve essere propagato, e non modificato, quando si usa iBGP. Questo significa che l'IGP deve trasportare anche next-hop relativi a prefissi esterni, bisogna portare all'interno di un AS una montagna di /30. Rendendo visibili all'interno dell'AS anche i prefissi che stanno sulle DMZ rischia di non scalare, perché le DMZ possono essere tantissime, però se questo non lo facciamo non possiamo fare il recursive lookup. Gli ISP su questo talvolta fanno un trucco, modificando esplicitamente l'external next-hop facendolo diventare un next-hop interno, cioè il next-hop del prossimo router interno all'AS (ci sono vari vendor che consentono di fare questo trucco). Questo sostanzialmente corrisponde a disabilitare il recursive lookup, dopodiché però si deve stare attenti affinche la raggiungibilità dell'esterno dell'AS ci sia effettivamente.

## iBGP peering

Quando si mette in piedi un peering BGP non si fanno su un'interfaccia fisica, ma sulla loopback (lo possiamo fare perché i peering iBGP non devono stare necessariamente sulla stessa rete fisica). Questo ha un grossissimo vantaggio: se si ha un fault sull'interfaccia fisica, e un peering sull'interfaccia fisica, il peering sull'interfaccia fisica va a farsi friggere, ma un fault sulla loopback non si avrà mai. Il peering tra due router di frontiera via iBGP quindi non è un peering fisico, ma un peering logico; i pacchetti tra i due router possono fare le strade più stravaganti, in funzione dello stato del routing interno all'AS. Quand'è che questo peering non funzionerà più? Quando non ci sarà più connettività all'interno dell'AS, e come se ne accorgeranno le due loopback del fatto che non c'è più connettività? Quando su quel peering non arriveranno più i pacchetti di keep alive allora quel peering va giù. Ma allora perché non usiamo le loopback anche per fare i peering eBGP? Qui non avrebbe senso, anche perché l'RFC lo dice esplicitamente: i peering eBGP devono essere dei peering back to back, le due interfacce devono essere sulla stessa rete fisica, per poter dialogare.

## iBGP mesh

iBGP richiede che sia usato il full mesh per tutti i router di frontiera, ma questo, per un AS con 100 router di frontiera, richiede migliaia di peering, e questo non scala, anche perché ci vuole qualcuno che li configuri. Non solo: i peering poi vanno mantenuti attivi, quindi non si scala neanche sul piano tecnologico. BGP è un protocollo che deve assolutamente scalare, perché se BGP non scala internet non funziona. Ci sono due soluzioni per scalare: i route reflector e le confederazioni, ma vedremo solo i route reflector, perché le confederazioni sono più complesse e rispetto ai route reflector hanno pochi vantaggi. Prendiamo un computer, interno all'AS, e gli mandiamo tutte le informazioni che sono apprese dai router di frontiera.

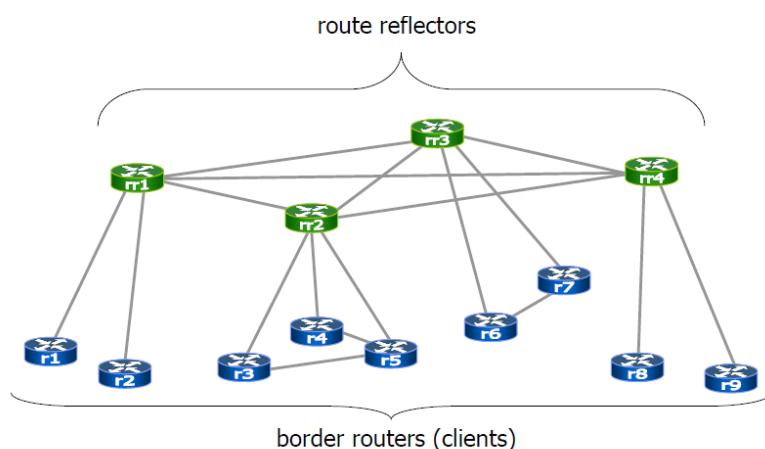


Figura 212 - Route reflectors

Questo computer calcola la rotta migliore per ciascuno dei prefissi e comunica la rotta ai router di frontiera. Questo scala molto di più, perché si passa da un numero di peering che è dell'ordine di  $n^2$  (full mesh) ad un numero di collegamenti col computer che calcola le rotte che è dell'ordine di  $n$ , ma questa è la morte delle reti, perché si ha un single point of failure. Come si rimedia? Ridondiamo il computer, con un'ulteriore variante: peering. Quanto ci vuole ad implementare un programma su quel computer che faccia il lavoro appena descritto? Poco: invece di re-implementare da zero, invece del PC ci metto una macchina che sa realizzare molto bene il decision process di BGP: un router che parla BGP. Invece di fargli fare da router gli faccio fare da computer. Questo router raccoglie tutte le informazioni, calcola la best e le riporta indietro. C'è anche già un protocollo apposito: BGP. Questo router/computer che calcola le rotte si chiama route reflector. Un route reflector riceve rotte da client e non client, cioè rispettivamente da router di frontiera e router ridondati. Due route reflector che parlano tra loro non sono client l'uno dell'altro, mentre un router di frontiera che parla con un route reflector e un client. I client possono essere non meshed. Se il percorso migliore arriva da un client viene propagato a client e non client, se invece arriva da un non client viene propagato solamente ai client. La situazione tipica è quella mostrata in figura. I route reflector vengono collegati tra loro in full mesh. Qual è l'effetto? L'effetto è che i router di frontiera hanno rotte, il route reflector li aiuta a trovare la best per loro, che quando è nota viene propagata alla periferia dell'AS. Il trucco in definitiva è: abbiamo preso la full mesh e l'abbiamo fatta diventare una full mesh di "molti pochi". Come si lavora per realizzare un'infrastruttura di route reflector? Si divide il backbone in vari cluster, e si mette almeno un route reflector e un po' di client per cluster. I route reflector vengono messi in full mesh (i client in un cluster possono essere messi in full mesh, ma non è necessario). Viene poi usato un IGP per trasportare i next-hop e le rotte locali (questo in realtà è quello che si fa normalmente, che ci siano o no i route reflector). C'è un problema di loop. Gli annunci provenienti dai client vanno propagati, ma vanno fermati ad un certo punto. Bisogna mettere un patch al protocollo, inserendo attributi che consentano di evitare loop di annunci nella gerarchia di route reflector:

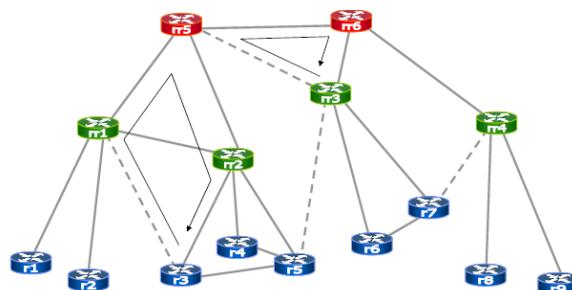


Figura 213 - route reflectors: possible loops?

- *Originator\_ID*: si porta l'identificatore del router che ha originato l'annuncio nell'AS locale; questo ID è creato dal route reflector, quindi non si deve fare nessuna modifica sul software che sta sui router client.

- Cluster\_list: ciascun gruppo di client che ha lo stesso route reflector e un cluster, al cluster è attribuito un cluster-id locale che viene aggiunto quanto il route reflector invia l'annuncio. Questo fa sì che quando l'annuncio torna indietro con lo stesso cluster-id tu lo possa buttare.

La cosa migliore da fare per scegliere il cluster-id è prendere il router-id, e il router-id lo prendi dall'indirizzo della loopback. Si possono configurare route reflector multipli configurati per lo stesso cluster (per scopi di ridondanza), ma questo può dar luogo ad alcuni problemi. Quello che si può fare è far fare da client ad un router per diversi cluster (e questo è quello che si fa attualmente). In questo modo ciascun client ha almeno due route reflector, che sono quelli relativi ai cluster a cui partecipa, e questo consente di fare un po' di ridondanza. Tutti i route reflector nel cluster devono avere lo stesso cluster-id (altrimenti apparterebbe ad un singolo cluster), e un router può essere client di differenti cluster. È facile oggi trovare ISP che sovrappongono due cluster per avere ridondanza (ciascun client ha due route reflector):

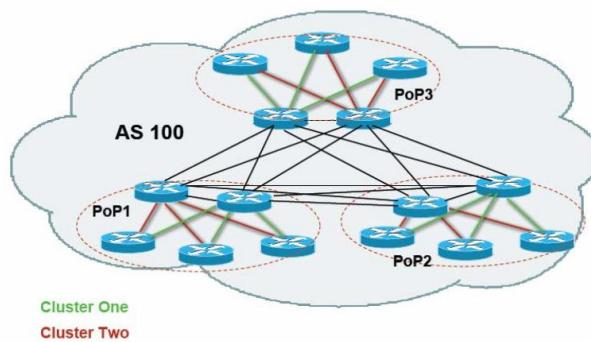


Figura 214 - route reflectors – deployment

A questo punto i router di frontiera di PoP1, PoP2 e PoP3 hanno due punti di riferimento, due cluster distinti con due diversi cluster-id. Siccome hanno due diversi cluster-id le informazioni se le possono scambiare, con un solo cluster-id non riuscirebbero a scambiarsi informazioni. Questa cosa è importante, perché se va giù un link vorremmo che le informazioni di routing arrivassero correttamente. A questo punto la situazione è un po' più complessa rispetto al punto di partenza, perché prima avevamo un router per ciascun gruppo, adesso ne abbiamo due, e dobbiamo continuare a metterli in full mesh, e lo facciamo nella maniera più ridondata possibile: ciascun route reflector è collegato a ciascun route reflector di ciascun altro cluster, quindi qui la full mesh si duplica. Un provider normalmente costruisce i propri cluster nei PoP. Un provider tipicamente ha un PoP in una certa città, in questo PoP ha un certo numero di router, e costruisce il cluster in quel PoP, in modo tale che ci sia anche contiguità geografica tra i vari client che stanno nel cluster. Vantaggi del route reflector:

- risolve il problema del mesh iBGP;
- il forwarding dei pacchetti non ne risente, e completamente irrilevante. I route reflector fanno i router per modo di dire: fanno i router perché sanno fare i conti BGP, ma non instradano neanche un pacchetto, per il data plane non sono utilizzati;

- coesiste con i normali BGP speakers;
- si possono avere reflector multipli (per avere ridondanza);
- la migrazione da una struttura senza route reflector a una struttura con route reflector e molto semplice;
- si possono avere piu livelli di route reflector.

Come possiamo fare a complicarci un po' la vita? Guardiamo la figura di prima. Quelli sono tutti PoP, e magari stanno tutti in Italia, e li collegiamo in quel modo; poi magari ho anche dei PoP in Germania, 4 per esempio, e in Germania metto in piedi la stessa configurazione. Che faccio? Collego ogni route reflector che sta in Italia anche con ogni route reflector che sta in Germania? Troppo complicato. Prendo allora tutti i route reflector che stanno in Italia e li metto sotto una coppia di route reflector gerarchicamente piu alta che si calcola tutte le rotte che girano per l'Italia, e metto questi due route reflector in corrispondenza con altri due route reflector che stanno in Germania, e ho organizzato una gerarchia su due livelli, ma posso farlo anche su tre, quattro livelli, quanto voglio, posso costruire una struttura di route reflector che è molto complessa. Di questo chi ne beneficia? I vendor, perchè riescono a vendere qualche apparato (non banale) in piu. Come si fa il deployment dei route reflector? Innanzitutto: dove li mettiamo? Il suggerimento è: scegli sempre di seguire la topologia fisica. Questo garantisce che il packet forwarding non sarà influenzato. Tipicamente si prendono i core router del PoP (ce ne sono almeno due) e gli si fa fare anche il mestiere del route reflector. Sopra questi due router ci si mettono due router virtuali e gli si fanno calcolare le rotte per i cluster corrispondenti. In questo modo la normale operatività del router è preservata, e si risparmia anche un po'. Che significa che il packet forwarding non sarà influenzato da queste scelte? Torniamo al processo decisionale di BGP. Quale potrebbe essere uno step influenzato in maniera negativa da ciò che stiamo mettendo in piedi? Lowest IGP metric, il cammino piu breve per arrivare al router di frontiera. Quando fa il calcolo del numero di hop il router guarda la tabella di instradamento dell'IGP che viene usato in quel momento, con la sua metrica. Il route reflector che fa i conti usa il normale processo decisionale di BGP, e quando arriva il punto della metrica IGP, a parità di ogni altra cosa, cosa sceglie? Per fare la scelta lui calcola qual è la distanza tra lui e i router di frontiera. Lui decide in funzione della distanza tra lui e chi gli ha dato l'annuncio. A questo punto il fatto che tu voglia che la metrica IGP sia importante nel processo di decisione (perchè vogliamo che il traffico fluisca il meno possibile attraverso l'AS) e completamente vanificato. Per risolvere questo problema il route reflector va messo molto vicino agli elementi che fanno parte del cluster, così sostanzialmente dal punto di vista delle metriche IGP ha lo stesso valore per i router di frontiera.

## Migrazione

Il deployment è veramente banale: si ha già un full mesh, i peering sono già tutti in piedi. Poi prendo un cluster, un gruppo di router e ci metto un route reflector. Faccio peering col route reflector, butto giù i vari peering che esistevano su quei router e li sostituisco con dei router che vanno sul route reflector.

Questo fa sì che non si abbia un d-day, in cui tutto succede contemporaneamente, si può fare la migrazione di un gruppo di router in un PoP indipendentemente dalla migrazione che fai sugli altri router. Non si deve far altro che creare un'ulteriore gerarchia se la core mesh è troppo grande, dividendo il backbone in regioni. Si devono configurare i cluster a coppie, eliminando i peering iBGP ridondanti e mettendo al massimo un route reflector per ciascun cluster.

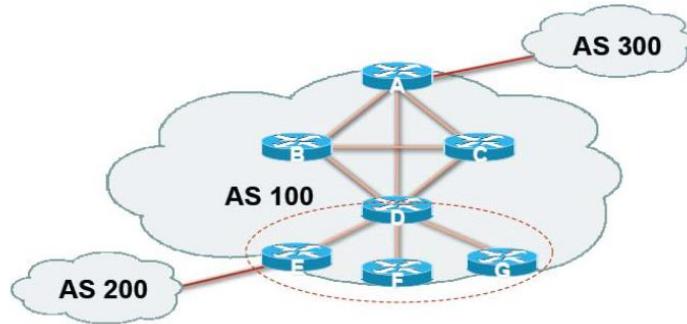


Figura 215 - route reflectors – migration

## BGP Communities

Un secondo meccanismo importante per scalare su BGP sono le communities. Non sono nient'altro che un ulteriore attributo. Consentono di raggruppare i prefissi in classi (communities) all'interno della rete dell'ISP. Ogni community ha un significato diverso e porta a differenti risultati dal punto di vista del calcolo delle rotte. Le communities sono una cosa piuttosto consolidata in BGP, e sono definite dall'RFC1998. Dobbiamo pensare ad una community come ad una stringa di bit che sta dentro un annuncio BGP, come tutti gli altri attributi. Questa stringa viene rappresentata come due numeri interi da 16 bit, per cui si trova che una community ha un formato del tipo *numero:numero* (e più un'etichetta che si appiccica sull'annuncio). Un tipico formato delle community è questo:

---

<local ASN>:xx

---

dove xx è un numero progressivo. Però questo è soltanto un common format, perché quelle sono label che si possono appiccicare sugli annunci, e uno le label se le sceglie un po' come preferisce. Si fa tipicamente l'assunzione che un certo numero di queste label siano riservate. Le communities sono utilizzate per raggruppare le destinazioni e non necessariamente una destinazione è membro di una sola community. Sono utili per applicare politiche all'interno dell'AS e tra AS.

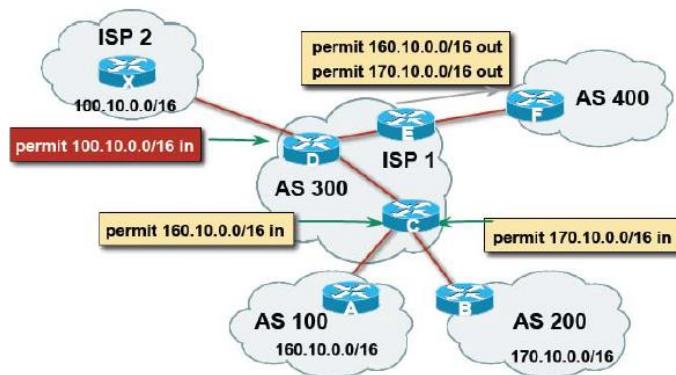


Figura 216 - community example: before

AS300 è un AS che riceve annunci da vari vicini, e magari li propaga ai vicini. Lui riceve l'annuncio della 100.10.0.0/16 da ISP2, dopodiché riceve la 160.10.0.0/16 da AS100 e la 170.10.0.0/16 da AS200. La 160, la 170 e la 100 sono tutte permesse in ingresso, però quello che si fa uscire sono le 160 e la 170, le altre non vengono annunciate ad AS400. Adesso, che cosa si dovrebbe fare quando si fanno queste politiche? Ci si devono costruire dei filtri specifici per ciascun prefisso, e questo può essere faticoso. Posso associare a certe tipologie di prefissi una etichetta.

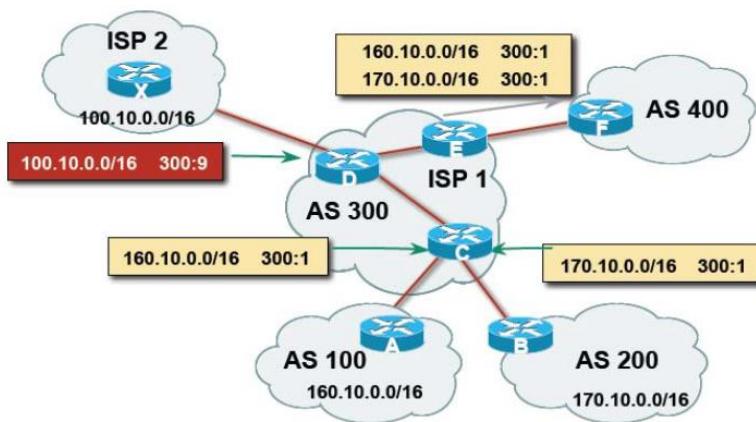


Figura 217 - community example: after

Alla 160 e alla 170 associo l'etichetta 300:1, e alla 100.10.0.0/16 associo l'etichetta 300:9. Questa etichetta identifica un prefisso che non deve essere annunciato, invece la 300:1 identifica un prefisso che deve essere annunciato fuori. A questo punto quando descrivi le politiche che consentono di fare l'annuncio in uscita oppure no piuttosto di fare il matching sul prefisso si fa il matching sulla community, e questo ci consente di scrivere le politiche in maniera più semplice. Questo delle community è un attributo che dovrebbe essere non transitivo, perché riguarda solo il rapporto che c'è tra me e un altro; però se

guardiamo gli annunci che girano in internet sono pieni di communities, perché BGP non pulisce l'attributo da solo, dovrebbe essere chi annuncia che pulisce dall'annuncio le communities che non gli servono, eventualmente inserendone delle altre. Alcune communities sono considerate *well known*. Di quelle well known “più well known” ci sono queste communities qua:

- no-export 65535:65281: tradizionalmente si usa per dire “non mandare questo annuncio a nessun peer eBGP”;
- no-advertise 65535:65282: “non mandare questo annuncio a nessun peer BGP” (sia che sia iBGP sia che sia eBGP);
- no-export-subconfed 65535:65283: l’idea è questa: “non fare un annuncio di questa cosa fuori da questo AS”, e si usa quando l’AS fa parte di una *confederation*);
- no-peer 65535:65284: “non fare l’annuncio di queste communities ai peer bilaterali”.

no-export, esempio di utilizzo

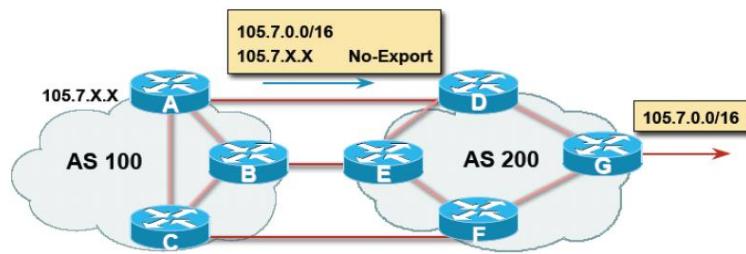


Figura 218 - no-export community

AS100 è un customer di AS200, con un certo numero di peering tra i router dei due AS. Adesso AS100 annuncia il proprio prefisso (105.7.0.0/16). Per i soliti motivi di bilanciamento di carico, assieme a 105.7.0.0/16 sui diversi link AS100 può annunciare prefissi più specifici (quel 105.7.x.x). Possiamo etichettare questi annunci con un no-export. Questo fa sì che nei peering che ha AS200 ha a sua volta con i suoi upstream questi prefissi più specifici non vengono annunciati. Questo consente di pulire un po' la tabella di instradamento di più alto livello. Del resto, se AS200 ha un unico peering con un upstream e inutile annunciargli sia le più specifiche che le meno specifiche, perché tanto qualunque cosa accada quello che tu vedi è l’insieme dei prefissi.

no-peer, esempio di utilizzo

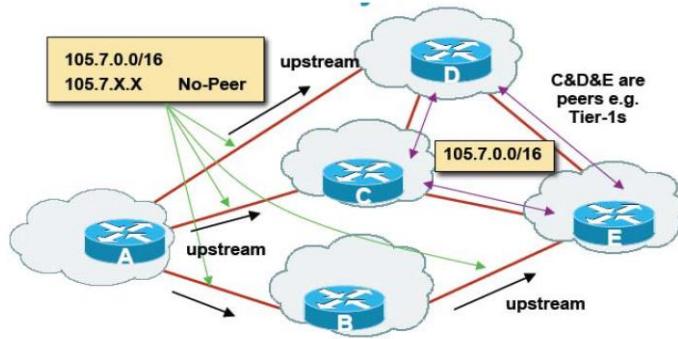


Figura 219 - no-peer community

C'è un AS (quello col router A) che ha 3 upstream, che a loro volta hanno altri upstream. Ci sono poi a livello più alto dei peering bilaterali, scambi tra pari, non c'è uno che paga un altro per raggiungere il livello più alto di internet. Io mi rivolgo ad un upstream, quello si rivolge ad un altro upstream ed ecco che arriviamo sul "tetto del mondo", dove non c'è un solo deus ex machina, ci sono diversi provider, più o meno tutti allo stesso livello, che si scambiano tra loro tutti i prefissi che arrivano dal basso. Puoi fare uso della community no-peer, in cui si usa ancora il "trucco" delle più specifiche (per fare bilanciamento di carico), ma a questo punto queste più specifiche vuoi che non inquinino la tabella di internet a più alto livello. A questo punto dici ad un certo AS (in figura quello col router C) "filtrami queste più specifiche, perché non voglio che girino per la tabella di instradamento di internet". Il fatto di avere dei peering bilaterali non avviene soltanto al top level della gerarchia, ma può avvenire anche a livelli più bassi della gerarchia di internet, come nei NAP (*neutral access point*) degli IXP.

## Il grafo di internet

Internet non è sotto il controllo di una sola autorità, è pieno di operatori. Questi operatori spesso hanno bisogno di sapere come è fatta la struttura in cui operano. Fino a che punto possono effettivamente conoscerne le caratteristiche? Ci interessa ricostruire almeno il grafo di internet, inteso come il grado delle interconnessioni. Ci sono vari motivi per capire come è fatto questo grafo, come per esempio network management e capacity planning, cioè, uno può chiedersi “ma internet, così come funziona adesso, è stabile? È sufficiente nella sua crescita a veicolare la crescita di informazioni che possiamo immaginare che ci sarà nel prossimo futuro?”. Perché uno si pone problemi di questo genere? Possiamo per esempio pensare ad un governo, che vuole assicurarsi che l'infrastruttura su cui ormai gira il 90% dell'economia sia robusta a sufficienza. Non solo: guardando la rete, possiamo ricavare informazioni sui rapporti commerciali tra chi ci lavora?

### Cos'è il grafo di internet

Posso pensare al grafo di internet come al grafo di interconnessione tra i router, oppure posso astrarre e pensare a questo grafo come al grafo di interconnessione tra gli AS. Parlare quindi del grafo di internet è una cosa che va fatta specificando ogni volta il livello di astrazione a cui vogliamo costruire la nostra rappresentazione. Siamo tipicamente interessanti al grafo semplificato, anche se vedendo gli AS come puntiformi qualcosa ce la perdiamo, anche soltanto per capire il routing interdominio, perché il routing interdominio si capisce bene quando si sa che dentro un AS c'è anche iBGP in azione, perché iBGP impatta le politiche. Tra due AS ci possono essere anche 20, 30 peering contemporaneamente attivi, in 20, 30 posti diversi del paese, quindi guardare al grafo significa poco, piuttosto si guarda al multigrafo, in cui per ogni coppia di nodi si vanno anche a contare gli archi, ma anche guardare solamente il grafo di interconnessione è piuttosto complicato.

### Sorgenti informative

Quali sono le sorgenti informative a cui rivolgersi per capire come è fatto questo grafo? Una di queste fonti sono sicuramente gli *Internet Routing Registry* (IRR); questi registri non soltanto hanno l'elenco dei prefissi così come sono assegnati, ma consentono anche di descrivere (e memorizzano) le politiche che sono utilizzate dai diversi AS. Le descrizioni sono specificate in un linguaggio chiamato RPSL (*Routing Policy Specification Language*). Ci sono anche sorgenti che non sono amministrative (provenienti dai registri), ma operative:

- tabelle BGP;
- annunci BGP;
- looking glasses;
- tabelle di forwarding.

[Internet Routing Registry \(IRR\)](#)

L'IRR è l'unione di un insieme di database utilizzati per descrivere politiche con RPSL. Questo:

---

```
inetnum: 193.204.160.0 - 193.204.161.255
        netname: ROMA3-NET
        descr: Terza Universita' di Roma
        country: IT
        admin-c: EV182-RIPE
        tech-c: GL965-RIPE
        tech-c: PC1670-RIPE
        rev-srv: mail.uniroma3.it
        rev-srv: decsrv.caspur.it
        rev-srv: dns.nis.garr.it
        status: ASSIGNED PA
        mnt-by: GARR-LIR
        changed: persi@caspur.it 19960209
        changed: querzola@cnaf.infn.it 19981229
        changed: gallo@garr.it 20011129
```

---

è un esempio di query che possono fare su whois per vedere cosa si sa di un prefisso. (nello specifico 193.204.161.0). Scopriamo che questo prefisso fa parte di un lotto che va dal 193.204.160.0 fino al 193.204.161.255 , e che è un lotto dell'Università Roma Tre. Sappiamo anche che chi gestisce questo prefisso ha certe caratteristiche (ci sono un certo numero di dati amministrativi). In particolare, guardiamo quel mnt-by: GARR-LIR; questo ci dice che c'è un sottoregistro da cui il prefisso viene, e quel LIR sta per Local Internet Registry, ed è il LIR dei prefissi che sono stati allocati a GARR, che gestisce queste informazioni in un registro più piccolo, in cui ci sono tutte queste informazioni che poi confluiscono in un registro un po' più grande che è il prefisso europeo. Successivamente si trova un elenco delle persone da contattare per avere ulteriori informazioni su questo prefisso. Quel tech-c: GL965-RIPE descrive una contact information, GL965-RIPE è un'ulteriore cosa su cui possiamo fare un whois per avere ulteriori informazioni. Possiamo anche fare un'altra query sul whois del RIPE, gli possiamo chiedere informazioni sull'AS137:

---

```
aut-num: AS137
as-name: UNSPECIFIED
descr: GARR external peering
.....
import: from AS5441
action pref=100;
accept AS5441
```

---

Ci viene risposto che AS137 è un AS, non ha nessun nome. La descrizione ci dice che questo è l'AS che il GARR usa per i peering esterni. Poi ci troviamo frasi come quell'import: quella è una import policy; quando

si ha una politica di import la si specifica attraverso quell'attributo import, e la semantica è questa: da AS5441, evidentemente attraverso un peering (si può specificare anche una preference), accetta AS5441, e la semantica di AS5441 lì è "tutti i prefissi che sono originati da 5441", e questa è una politica completa, rappresentata ad un livello di astrazione appena un po' più alto di ciò che consente di dire BGP. Un altro pezzetto di politica di AS137 è questo:

---

```
import: from AS9010
      action pref=100;
accept AS9010 { 157.130.0.0/16 }
```

---

Il significato in questo caso è "prendi tutto quello che è originato dentro AS9010, e in più accetta quella /16". Possiamo poi scrivere delle politiche di export:

---

```
export: to AS5441
      announce AS137
      export: to AS3269
      announce AS-GARR
```

---

qui annunciamo tutti i prefissi che sono originati da AS137 e da AS-GARR, che è una macro per raggruppare più AS (da qualche parte ci sarà scritto "AS-GARR contiene tutti questi AS"). Ci sono poi dei sistemi costruiti intorno ai registri, per rappresentare le stesse informazioni ma in maniera più gradevole. Uno di questi è Hermes (<http://tocai.dia.uniroma3.it/~hermes/>), e un altro è RPSL Analysis Service, che estrae le informazioni dall'IRR ed effettua anche delle verifiche di consistenza delle politiche così come sono osservate nei diversi punti della rete. Un IRR è un database di politiche espresse in modo "notarile": un ISP va da un IRR e dice "queste sono le mie politiche". Questo ha degli effetti positivi: chiunque voglia contattare quell'AS sa qual è l'atteggiamento di quell'AS nei confronti del resto del mondo. Ha però anche degli effetti negativi: ci sono AS che non valutano positivo raccontare a tutti come si comportano, perché altri potrebbero ricavarne un vantaggio competitivo. Quindi possiamo trovare da qualche parte informazioni molto dettagliate e da altre parti nessuna informazione, e del resto non c'è nessun vincolo per un ISP di descrivere le proprie politiche in questi registri, mentre c'è un vincolo che dice che se tu hai un lotto di indirizzi IP quel lotto di indirizzi IP deve essere correttamente rappresentato in questi registri.

### Looking glasses

È il tipico strumento operativo di chi fa routing interdominio. È un software che gira su un web server di un ISP e fa capire all'utente esterno come funziona il routing dal punto di vista di quell'ISP. Tipicamente è uno script che consente di eseguire dei comandi tipo show ip bgp dentro al router. Vediamo una query eseguita su un provider pakistano (Pakistan Internet Exchange) sul router di Lahore, in cui si chiedono informazioni su 193.204.161.0. Andare su un looking glass di un paese parecchio remoto non è male,

perché tu vedi a questo punto come i prefissi si siano propagati attraverso una struttura molto profonda, si riesce a vedere parecchio della propagazione dei prefissi attraverso internet. Quello che esce è (o meglio: era, le informazioni cambiano continuamente) questo:

---

```
BGP routing table entry for 193.204.0.0/15, version 54000344
  Paths: (2 available, best #1)
    Advertised to non peer-group peers:
      202.125.135.23
        7473 3549 137 137 137, (aggregated by 137 193.206.129.252),
          (received & used)
      203.208.147.85 (metric 35) from 202.125.135.1 (202.125.135.1)
        Origin IGP,localpref 200,valid,internal,
          atomic-aggregate, best
        8961 8966 3561 3549 137 137 137,
          (aggregated by 137 193.206.129.252)
      195.229.0.1 from 195.229.0.1 (195.229.0.1)
        Origin IGP,localpref 100,valid,external,atomic-aggregate
```

---

La prima cosa che vediamo è che quella /24 è aggregata in una /15 (evidentemente il GARR annuncia tutta la /15) e poi scopriamo una serie di as-path, che descrivono come abbia navigato il prefisso attraverso la rete per raggiungere il Pakistan (in particolare il router di Lahore). Questo:

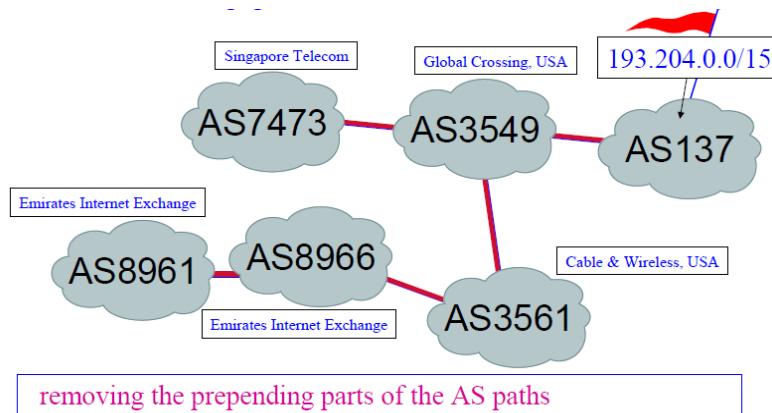


Figura 220 - Looking glass

è il pezzetto di internet che riesco a capire osservando il risultato di quella query. Quella query mi fa capire che 137 è l'origine di quella /15, e che in tabella ci sono annunci diversi, "catturati" dal looking glass: uno che passa per 3549, 3561, 8966, 8961 oppure 3549, 7473. Quello che stiamo guardando in realtà è uno show ip bgp su un router remoto. Per costruire il percorso del prefisso abbiamo eliminato il prepending, che è presente in modo massiccio attraverso 137. Tornando al risultato della query, osserviamo che di quei percorsi il migliore secondo BGP è il numero 1, presumibilmente dalla lunghezza dell'as-path, che è più corto; tutti e due i percorsi arrivano comunque con una montagna di prepending (per scoraggiare una via piuttosto che un'altra, in funzione del fatto che ci sono più vie). Qui sembra che

tutti e due gli annunci abbiano seguito la stessa strada, e questo è possibile su internet: tu GARR sopra di te hai qualche upstream, su uno fai prepending e sull'altro no, quindi passi per quello su cui non hai fatto prepending, però magari una certa porzione di internet la raggiunge soltanto l'annuncio su cui tu hai fatto prepending (e lì c'è poco da scegliere, è l'unica strada possibile). Qui si riesce anche in qualche modo a capire quale sia la politica di GARR attraverso questi annunci: GARR fa dei peering con degli upstream che sono di due tipi: ci sono degli upstream che sono istituzionali della ricerca europea; in particolare uno è GEANT, che è la rete della ricerca europea. I peering con GEANT non pagano (perché ci sono accordi di ricerca tra le varie istituzioni) quindi preferibilmente GARR vuole passare per GEANT, e sugli annunci che fa a GEANT non mette prepending. C'è poi del traffico per i commerciali, perché tu devi raggiungere tutto il mondo, e ti rivolgi ad un certo numero di AS. In particolare qui sembra che si rivolga volentieri a 3549 e sugli annunci che fa a 3549, per cui magari paga, ci mette un po' di prepending. Ora, la connettività che riesci ad avere attraverso GEANT evidentemente non è completa, gli annunci che mandi su GEANT non riescono ad arrivare in tutto il mondo; magari riescono ad arrivare su tutte le reti della ricerca in giro per il mondo, in maniera molto veloce, però non in tutto il mondo, e questo sembra che non avvenga in Pakistan in cui l'unico che ci porta lì è 3549. Qui si apprezza il fatto di essere abbastanza remoti rispetto a GARR, se fossimo stati più al centro della rete di questi salti attraverso AS ne avremmo visti di meno. Andando ad indagare meglio nel risultato della query scopriamo che il primo path ha una local preference più alta del secondo, quindi in realtà è stato scelto per questo motivo, e non per la lunghezza dell'as-path. Ci sono poi i next-hop e l'advertiser, quello che ha mandato l'annuncio. A questo punto ci chiediamo "sì, è vero, questa è la strada che ha fatto l'annuncio, queste sono le local preference che abbiamo visto rispetto a questo router sul quale stiamo utilizzando il looking glass, però qual è l'AS sul quale siamo effettivamente atterrati? Che vuol dire che da una parte c'è il next-hop che è uguale all'advertiser, e da un'altra parte no?". Tipicamente il next-hop e l'advertiser coincidono se sei su eBGP, perché il next-hop è quello che ti dà l'annuncio. Se sei su iBGP il nexthop è quello remoto, mentre l'advertiser è quello che ha mandato effettivamente l'annuncio, e quindi questa differenza potrebbe essere sinonimo del fatto che stiamo guardando un peering iBGP invece che un peering Ebgp. Per capire però effettivamente quale sia l'AS pakistano possiamo fare uno show ip bgp summary, e questo ci chiarisce che l'AS di cui parliamo è 17557:

---

```
BGP router identifier 202.125.135.21, local AS number 17557
BGP table version is 54174845,
main routing table version 54174845
110427 network entries and 219911 paths
.....
```

---

Questo mi dà dei dati operativi, quando faccio la query sul looking glass vedo qual è in questo momento lo stato del routing, e quello che posso fare è dire "questo è un dato operativo, come sta questo dato

operativo rispetto al dato amministrativo?" Cioè: cosa posso sapere sui registri di quell'AS, e le due informazioni (operativa e amministrativa) coincidono? Posso vedere cosa ne pensa Hermes.

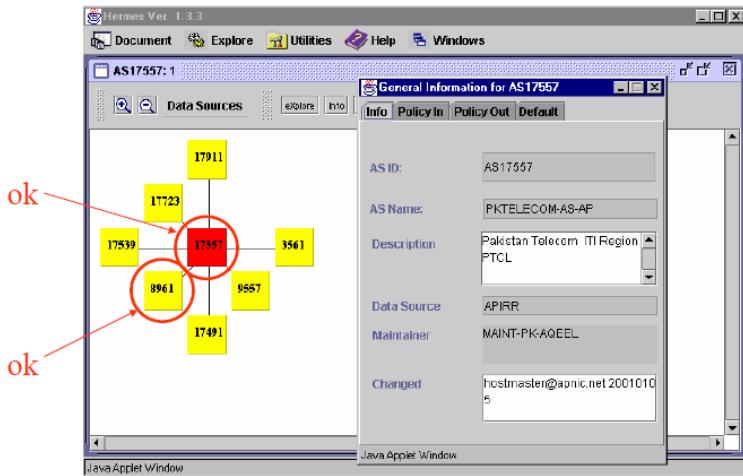


Figura 221 - Opinione di Hermes

Lì vediamo un peering con 8961, che è uno di quelli emersi con il looking glass. Dopodiché però c'è anche un peering con 7473, che nei registri non sembra esserci: abbiamo trovato un'inconsistenza tra le informazioni che sono nei registri e il dato operativo, e questo è un problema, perché se vogliamo ricostruire il grafo di internet in base ai registri abbiamo delle informazioni che sono assolutamente parziali. Il fatto poi di trovare anche 3561 lo posso considerare un'inconsistenza? No, perché stiamo guardando le cose su uno specifico looking glass che sta su uno specifico router dell'ISP pakistano. Questo non implica che non ci possa essere un altro router attraverso il quale questo peering è visibile, quindi che questa sia un'inconsistenza adesso è difficile dirlo, dovremmo conoscere tutti i path attraverso 17557. Per farlo andiamo a fare una query su un altro router, quello che sta a Islamabad:

---

```
BGP router identifier 202.125.135.41, local AS number 17557
.....
Neighbor V AS MsgRcvd MsgSent TblVer
      InQ OutQ Up/Down State/PfxRcd
202.125.135.1 4 17557 2507595 39623 15279186
      o o 3w6d 110095
.....
202.125.149.178 4 9557 18329 44508 15279183
      o o 5d18h 4
```

---

Scopriamo che 9557 è un altro neighbor di 17557, e questo ci dà la possibilità di verificare un'altra delle informazioni che vengono dai registri.

#### Route views

È un'altra sorgente di informazioni operativa. Quello che fa è dei peering eBGP multi-hop con router che sono dispersi in giro per il mondo. Qual è l'idea? Se vuoi sapere come è fatto il mondo che cosa puoi fare?

Puoi prendere accordi con un po' di AS dispersi qua e là e ti fai raccontare cosa loro sanno del mondo. Quando ho le informazioni le metto tutte assieme e a questo punto si ha una visione abbastanza integrata di come è fatto il mondo. Adesso, se i soggetti di cui stiamo parlando sono effettivamente interessati a collaborare, e il fatto poi di fornire tecnicamente queste informazioni, ci può essere un problema tecnologico della serie: quanto tempo ci vuole a realizzare il software che fa queste cose? Chiaramente l'azienda che partecipa a progetti di questo tipo vuole degli shortcut, non vuole mettere la sua operatività come

azienda appresso ad una cosa che semplicemente fa da mirror di informazioni che metti a disposizione e diventano di dominio pubblico.

Il modo migliore per fare questa cosa è usare BGP: prendo un router di un provider, ci faccio un peering BGP ed è il peering BGP che mi annuncia cosa vede il provider del resto del mondo.

Chiaramente i peering devono essere dei peering multi-hop, perché chissà dove stanno quei router, e devono essere dei peering eBGP, perché vuoi la decisione che ha preso quell'AS per raggiungere quello specifico prefisso. Naturalmente ci deve essere disponibilità da parte dei soggetti con cui si vuole mettere in piedi i peering. Il risultato di un'interrogazione è una cosa di questo tipo:

---

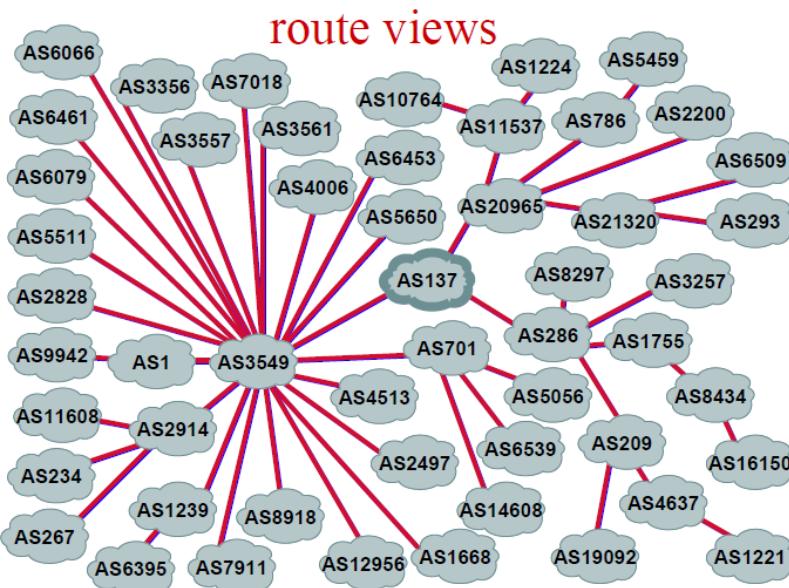
```
route-views.oregon-ix.net>sh ip bgp  
193.204.161.0  
BGP routing table entry for 193.204.0.0/15,  
version 168179  
Paths: (55 available, best #52)  
Advertised to non peer-group peers:  
       64.166.72.140  
4006 3549 137 137 137, (aggregated by 137  
       193.206.129.252)  
       205.215.45.50 from 205.215.45.50  
           (205.215.45.50)  
           Origin IGP, localpref 100, valid,  
           external, atomic-aggregate  
11608 2914 3549 137 137 137, (aggregated by  
       137 193.206.129.252)  
       207.246.129.6 from 207.246.129.6  
           (207.246.129.6)  
           Origin IGP, localpref 100, valid,  
           ...
```

---

è come guardare una serie di looking glass, dispersi in giro per il pianeta, con le informazioni messe tutte assieme. Guardiamo un frammento di questa query:

```
route-views.oregon-ix.net>sh ip bgp 193.204.161.0
BGP routing table entry for 193.204.0.0/15, version 168179
      Paths: (55 available, best #52)
        Advertised to non peer-group peers:
                      64.166.72.140
4006 3549 137 137 137, (aggregated by 137 193.206.129.252)
          205.215.45.50 from 205.215.45.50 (205.215.45.50)
Origin IGP, localpref 100, valid, external, atomic- aggregate
```

Ci dice che quello è un as-path che viene da Oregon Route Views, ed è 4006 3549 137 137 137. Questa riga ci dice che questa informazione viene da un peering tra route views e 4006, questo è quello che 4006 sa del mio prefisso. Che cos'è che 4006 fornisce a Oregon Route Views? C'è un peering tra questi due AS, e quindi c'è un annuncio, che riguarda la /15 che ospita la /24 di cui stiamo parlando. Questa non è la stessa informazione che avrei ottenuto con un looking glass su 4006. Siccome viene da un peering, quel peering potrebbe avere delle politiche specifiche che annunciano a Oregon Route Views soltanto una parte delle informazioni (è anche vero però che se c'è disponibilità a trasferire le informazioni possiamo immaginare che ci sia disponibilità a trasferirle tutte). Non solo: io tra tutti gli annunci vedo solo l'annuncio migliore. Quello che vedo sul looking glass quindi è un'informazione più completa: ci permette di vedere anche come si comporta l'upstream in presenza di fault (vediamo tutti i path a disposizione).



*Figura 222 - Route View*

Naturalmente il numero di AS in internet è molto più grande. Questa cosa che assomiglia ad un albero radicato su 137, che cosa ha sulle “foglie”? Gli AS con cui si hanno accordi, i peer del progetto Oregon Route Views. Posso fare query di questo tipo su tantissimi prefissi, uno appresso all’altro. Posso mettere insieme questi alberi e a questo punto ho un grafo dei peering di internet, che comincia ad essere un grafo di internet veramente significativo. Da quella figura si capisce anche un’altra cosa: AS3549 è Global

Crossing, e attraverso GEANT (AS20965) non raggiunge poi una parte così grande del mondo. La gran parte del mondo la raggiungere attraverso Global Crossing e l'altro upstream commerciale che ha, quindi malgrado tutti i suoi sforzi di fare peering con un prepending molto pronunciato si continua a raggiungere il mondo preferenzialmente attraverso i peering commerciali (evidentemente non c'è connettività sufficiente per raggiungere il resto del mondo).

Questo grafo è il grafo del traffico in ingresso oppure in uscita? Questo è il grafo del traffico in ingresso, perché è il grafo degli annunci che partono da 137 e raggiungono il resto del mondo, ed effettivamente per 137 questo grafo è interessante: 137 è un AS il cui core business non è fare content delivery, qui il core business è acquisire informazioni e dati dal resto del mondo. Però se invece di 137, cioè GARR, fossimo Akamai, probabilmente il grafo di interesse non sarebbe questo, ma l'opposto: vogliamo sapere come i nostri dati raggiungono il resto del mondo, e non è sicuramente questa la query da fare. Questo grafo però è sempre un albero oppure no? Innanzitutto possiamo pensare che questo grafo abbia un'orientazione, quindi quando ci chiediamo se sia un albero oppure no non ci stiamo chiedendo se ci siano cicli, ma quelli che nel gergo dei grafi si chiamano semicicli, cioè cicli non orientati. Non ci possono essere semicicli, perché BGP, ricevuto un insieme anche piuttosto ampio di annunci da un certo prefisso, ne propaga soltanto uno, e siccome ne propaga soltanto uno, come fanno a uscire due cammini da un AS? In realtà questo è possibile in maniera abbastanza semplice: posso prendere delle scelte verso due router di frontiera, e siccome sono due router distinti possono fare scelte diverse, magari mi arrivano due as-path diversi. Questo perché dal processo decisionale di BGP emerge chiaramente che non tutti i router di frontiera debbano necessariamente fare la stessa scelta, anche se loro via iBGP si parlano. Una cosa che si vede con estrema difficoltà è la presenza di multipeering. Non può essere che 137 e Global Crossing abbiano più peering, magari uno a Roma e uno a Milano? Per cercare di capirlo si può fare un trucco. Abbiamo capito che ci sono dei router che fanno più scelte diverse contemporaneamente. Se tra i due AS c'è un altro nodo, e se su questo nodo arrivano due scelte diverse, allora lì c'è un multipeering. Per costruire il grafo che esce da 137 posso prendere tutti gli alberi, con radice in diversi AS; in tutti questi alberi posso selezionare il cammino da 137 alla radice, se ce n'è uno, e fondere poi tutti i cammini così trovati. Possiamo trovarci da questo metodo una risposta effettivamente efficace? Probabilmente no, si riesce da qui ad avere un'informazione ricca se c'è traffico che tipicamente passa per 137, ma siccome 137 rischia di essere un customer nella rete, il traffico di passaggio con questa tecnica non si vede.

## Damping algorithm

Sempre dalla risposta di route views:

.....  
7660 9270 2200 20965 137, (aggregated by 137 193.206.129.252)

(history entry)

203.181.248.233 from 203.181.248.233 (203.181.249.19)

Origin IGP, localpref 100, external, atomic-aggregate

Dampinfo: penalty 3010, flapped 6 times in 00:30:13

.....

---

scopriamo che per ciò che vede 7660 di 137 ci sono delle cose un po' strane. C'è scritto che quella è una history entry della tabella. Che vuol dire? C'è scritto anche Dampinfo: penalty 3010, flapped 6 times in 00:30:13 . Per capire cosa significa Dampinfo dobbiamo immaginare cosa succede continuamente in internet. In internet ci son router che continuamente, da tantissime parti, vanno in fault, si accendono, si spengono, ci sono politiche che cambiano continuamente....e siccome poi gli annunci passano attraverso una pluralità di soggetti basta che uno dei passaggi subisca qualche modifica che la rotta cambia. Gli as-path in internet sono in continua modifica, l'instradamento in internet è tutt'altro che stabile. Tutto ciò può portare a delle conseguenze estremamente negative. Immaginiamo un AS che vede il nostro 137, e lo vede stabile. Poi ad un certo punto succede qualcosa, e il 137 (o meglio, quel prefisso specifico) non è più visibile per un po', poi torna ad essere visibile, poi è di nuovo invisibile...L'AS che ha questa informazione intermittente le deve annunciare ai vicini, perché questo dice BGP, quindi un po' le annuncia, poi dice "mi sono sbagliato", poi le annuncia di nuovo, poi dice di nuovo "mi sono sbagliato". I vicini che fanno? Ricalcolano la tabella di instradamento ogni volta che quell'AS si ricrede; questo aggiunge instabilità all'instabilità, con un meccanismo di propagazione terrificante su internet, che si difende inventando il meccanismo di dampening (o damping). In questo meccanismo, se tu vedi una rotta che fa flapping, il router automaticamente (ha un algoritmo per questo) prende quella rotta e la mette in penalità, con delle penalità che crescono secondo delle funzioni esponenziali, e dopo un po' quando queste penalità sono troppo alte, la rotta viene soppressa. Questa rotta rimane invisibile qualunque cosa accada, finché non passa un tempo di "purgatorio" per la rotta. Passato questo tempo la rotta viene annunciata di nuovo. La rotta di prima era "storica" nel senso che non era effettivamente usabile da quel router (perché ha una certa penalità). L'obiettivo dell'algoritmo è ridurre la portata della route flap propagation. L'algoritmo è pensato in questo modo:

- per convergere velocemente nel caso di normali route change;
- cerca di predire il comportamento futuro dei flap;
- sopprime le rotte che oscillano;
- annuncia soltanto le rotte stabili.

Ad ogni flap si aggiunge una penalità, e se la penalità sale sopra il limite di soppressione questa rotta non viene annunciata a nessuno dei peer. Recenti osservazioni però mostrano che la presenza del damping può avere effetti negativi sulla rete. In realtà c'è un rapporto costi-benefici da considerare. Tu vedi una serie di oscillazioni che dipendono da qualcosa sulla rete e sopprimi una rotta per tanto tempo; sopprimere

una rotta per tanto tempo è un problema per chi annuncia quella rotta: nel momento in cui termina la soppressione tu sei comunque costretto a riannunciarla, e in qualche modo anche tu stai facendo flapping della rotta. Quello che si è visto è che la sovrapposizione di questi fenomeni può essere una sovrapposizione negativa; questo è il motivo per cui in questo momento su un certo numero di router in internet si fa damping e su un certo numero no (questo è molto a scelta dei provider). Ci sono due diverse implementazioni del damping, più o meno morbide nella soppressione dei prefissi; alcuni tengono il prefisso per molto tempo prima di fare il riannuncio, altri riannunciano immediatamente.

### Ancora sorgenti informative

#### Routing information service

Finora abbiamo sempre potuto osservare la situazione della rete ad un certo istante, mancando completamente la prospettiva storica. A questa prospettiva storica pensa proprio il servizio Routing information service del RIPE, che ha un'architettura identica a quella di Oregon Route Views, soltanto che per scalare invece di usare un solo router per fare peering ne usa una molteplicità. Questi router si chiamano route collector, e le informazioni vengono memorizzate in un database. Qui tutte le informazioni BGP vengono memorizzate con un timestamp, quindi tu sai qual era lo stato del routing in un certo istante della rete anche tornando indietro di parecchio nel tempo. Per capire ancora meglio questo meccanismo, e la dinamica di BGP, dobbiamo parlare ancora di una cosa. BGP non ha soltanto annunci, ma ha anche withdrawal, cioè ritiri, e quello che si memorizza nel database del RIPE non sono solo gli annunci, ma anche i ritiri. Un ritiro funziona in questo modo:

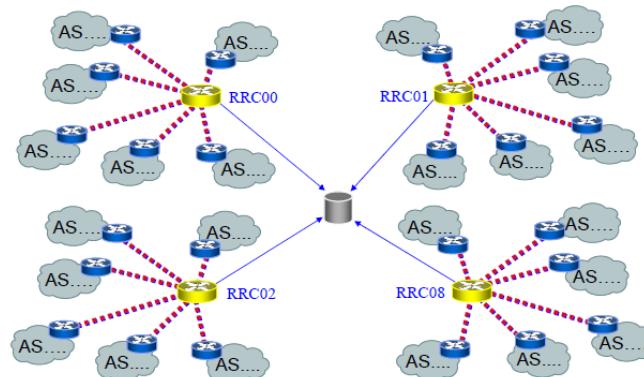


Figura 223 - routing information service

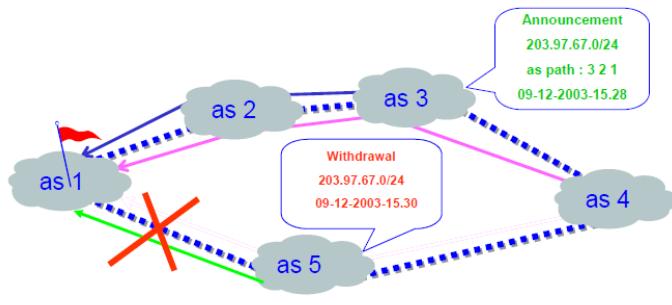


Figura 224 - bgp updates

AS1 manda fuori il suo solito prefisso, e questo prefisso è visibile attraverso AS5. Va giù quel link; se AS5 avesse un'altra rotta per raggiungere quel prefisso modificherebbe il proprio annuncio, farebbe soltanto un annuncio diverso. Ma se questa rotta non è più disponibile, cioè non sa più come raggiungerlo, allora AS5 fa un withdrawal, cioè dice “d'ora in avanti non so più darti quel prefisso”. Quindi a questo punto capiamo che se finora abbiamo parlato solo di annunci, dobbiamo parlare più in generale di update BGP, dove gli update sono sia i messaggi positivi che negativi. Quello che viene memorizzato nel database del RIS sono quindi gli update BGP, con l'istante in cui quella cosa è avvenuta. I risultati di queste interrogazioni sono molto articolati, e si capiscono meglio con un disegno:

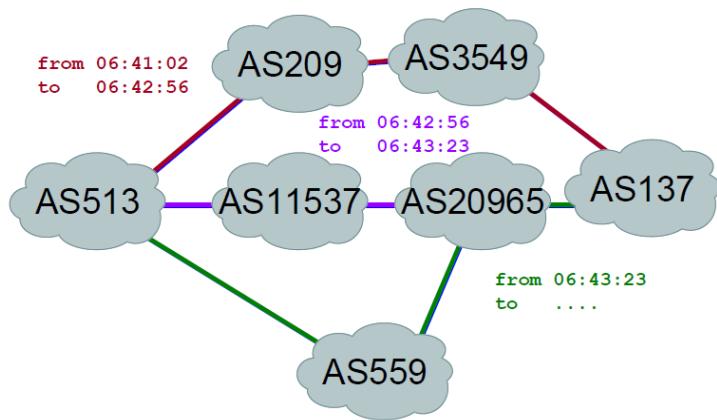


Figura 225 - routing information service

Questa è una fonte preziosissima di informazioni, perché si riesce a capire non soltanto quale sia il routing stabile, ma anche che cosa succede durante le instabilità, che si possono studiare veramente al microscopio. In questo modo si possono capire anche quali sono le rotte di backup degli AS, che normalmente non si vedono.

## Analisi dei dati di Internet

Tanto per iniziare: una definizione per Internet:

---

packet switched communication facility in which a number of distinguishable networks are connected together using packet communication systems called routers which implement a store & forward packet forwarding algorithm

---

## Crescita delle tabelle di instradamento

I router di backbone hanno delle tabelle di routing enormi, e questo è un problema, perché quando arriva un pacchetto dobbiamo guardare l'indirizzo IP del destinatario e fare un lookup su tabelle di centinaia di migliaia di elementi. La tabella può anche essere implementata in maniera efficiente, ma i pacchetti sono tanti (10, presto 100 Gb/s). Ce la fanno i router a gestire questa sfida? Non ci sarà un momento in cui le tabelle diventeranno troppo grandi? Sappiamo cos'è una tabella di instradamento, cos'è una rottura, sappiamo che una tabella di instradamento la possiamo vedere a due livelli d'astrazione:

- come Forwarding Information Base (FIB): è la tabella di instradamento del data plane;
- come Routing Information Base (RIB): è la tabella di instradamento del control plane (peraltro sappiamo che ce n'è un certo numero di queste); sono una parte della RIB diventa FIB.

Ci stiamo occupando non delle tabelle di instradamento di tutti i router, ma delle tabelle di instradamento nella zona di internet in cui non c'è la rottura di default (zona default free), i provider di questo livello si scambiano tutti i prefissi, tutti con tutti. Questa parte della rete è quella in cui i router sono messi più a dura prova. La backbone è essenzialmente una zona default free. Stiamo parlando di questo:

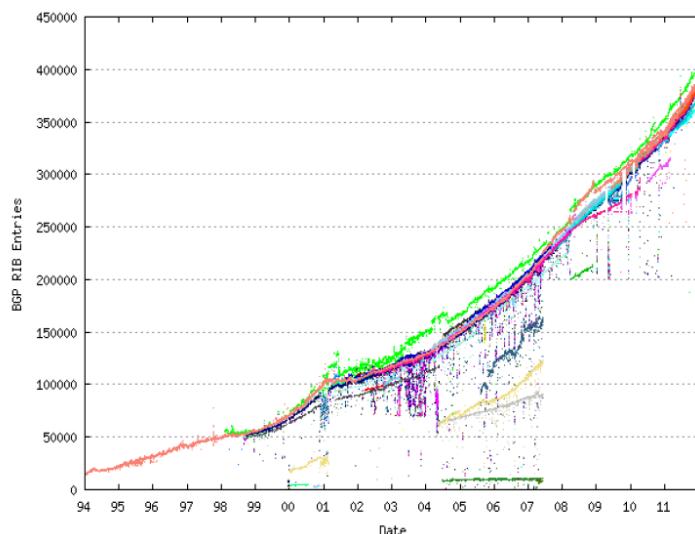


Figura 226 - Backbone

Questo è il grafico della tabella di instradamento di internet nella zona default free dal 1994 fino a qualche giorno fa. Ci sono diversi colori, perché questi sono diversi grafici, sono i grafici dei punti di osservazione di diversi router nella rete; da una zona all'altra può cambiare il numero dei prefissi, perché magari è stato usato CIDR un po' meglio, o un po' peggio, o ancora alcuni prefissi possono semplicemente essere non visibili. Le zone frastagliate possono corrispondere a dei fault, o dei momenti in cui alcuni dei router osservati non hanno vissuto nella zona default free, ma hanno avuto la default. Ci sono stati dei momenti in cui la crescita delle tabelle è stata lineare, e altri in cui è stata piuttosto significativa: più che esponenziale. Uno dei motivi può essere il crescente successo di internet, o l'arrivo in massa dei provider asiatici, con nuovi prefissi allocati e annunciati. Negli ultimi anni però non c'è solo l'arrivo dei provider asiatici, ma anche la mancanza di indirizzi IPv4. Perché però se gli indirizzi finiscono la tabella di instradamento cresce anche esponenzialmente? Pochi indirizzi a disposizione, tanto subnetting; se prima annunciavi su internet la tua /15, adesso la frazioni, la distribuisci in giro, una parte te la vendi (anche se non si potrebbe fare). A questo punto quello che esce fuori sono diverse /24, che escono fuori in diversi punti della rete. Questo è un problema che ci sarà in futuro: i router ce la faranno a sostenere la crescita delle tabelle di instradamento? Da un lato si può rimediare con router più potenti, dall'altro si può agire più di metodo, eliminando delle rotte superflue. Quando posso dire che una rottura è superflua, che ho fatto bene a buttarla? Quando in realtà la connettività non è cambiata. Questa è un'attività difficile: è strano il fatto di buttare un prefisso e tutto continua a funzionare come prima. È difficile non aver bisogno di una riconfigurazione degli apparati, in seguito alla rimozione di una rottura. Quindi eliminare una rottura potrebbe richiedere uno sforzo considerevole. Lo strumento principe per eliminare le rotte è CIDR, ma aggregare prefissi non è sempre facile, soprattutto con riferimento all'AS path. Supponiamo di avere due prefissi contigui, e che questi due prefissi abbiano lo stesso AS path. A questo punto aggreghiamo i due prefissi, ne tiriamo fuori uno meno specifico e lo annunciamo con lo stesso AS path che è arrivato più il prepending del nostro AS number. Ma supponiamo che le due /24 che vogliamo aggregare in due /23 arrivino con AS path diversi. L'aggregazione si fa, ma quando si fa l'annuncio, con quale AS path si annuncia il prefisso? Non è chiaro. Possiamo ripensare l'attributo AS path, e scriverci dentro un AS set, invece che un AS sequence. Con un AS set possiamo dire "sono passato attraverso AS 5, AS 4 e poi attraverso un insieme di AS", senza specificare la sequenza di attraversamento. In questo modo se si hanno due prefissi da aggregare con AS path diversi possiamo mettere dentro un AS set e fare l'annuncio con quello. Se andiamo a guardare effettivamente i record di potaroo sull'uso di AS set troviamo che l'uso di AS set adesso è basso, e l'uso di questo attributo adesso si inizia a deprecare (perché con questo attributo ci si può fare un po' di hacking delle tabelle di instradamento).

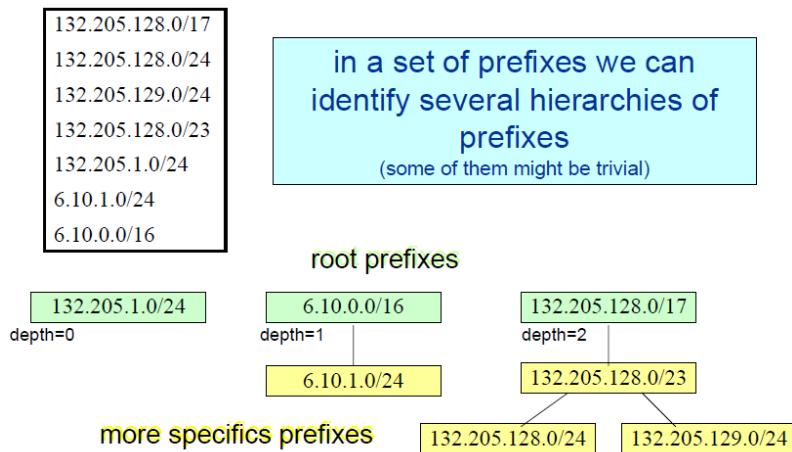


Figura 227 - Gerarchizzare i prefissi

Malgrado una notevole attività di aggregazione si trovano ancora prefissi superflui. Un buon modo per guardare ai prefissi superflui è questo: tu puoi prendere i prefissi, e gerarchizzarli. Per esempio, troviamo una /16, la 6.10.0.0/16, poi troviamo una /24 più specifica, la 6.10.1.0/24. A questo punto le mettiamo in dipendenza gerarchica, creando degli alberi che sono delle gerarchie di prefissi. L'ideale sarebbe avere nella tabella di instradamento di internet solamente prefissi radice. Per far sì che in internet ci siano solo processi radice di quegli alberi si può quantomeno affrontare problemi di cattiva configurazione, ma magari qualcuno ha fatto multi homing, e qualche volta i prefissi più specifici sono nella tabella di instradamento perché qualcuno ha fatto bilanciamento di carico, o perché qualcuno ha fatto backup recovery, qui è difficile fare piazza pulita. Uno strumento utile per evitare di propagare in giro i prefissi usati per load balancing e multi homing: le community. Possiamo utilizzarle per segnalare agli upstream che solo alcuni dei prefissi che vengono annunciati devono essere ulteriormente annunciati all'esterno. Questo fa sì che tu annunci i prefissi più specifici fino ad un certo livello della gerarchia di internet. C'è un altro fenomeno: quello della summarization, in cui c'è una violazione della gerarchia dei prefissi. Questo fenomeno è il seguente: io ho preso una /15, ad un certo punto qualcuno mi chiede una /24. È un mio cliente e gliela do. Questo cliente ad un certo punto decide di andare con un altro provider, e si porta via la mia /24, che viene annunciata da qualche altra parte in internet. Quindi in internet ci sono la /15 che io continuo ad annunciare, e la /24 che fa capolineo da qualche parte. Non posso tagliare via quella /24, altrimenti quel cliente non lo vede più nessuno. Questa è una tipica situazione di sommarizzazione, di violazione della gerarchia di internet. Quindi cosa succede nel tempo? Dal punto di vista della raggiungibilità delle reti non succede niente, perché IP fa best prefix match. Succede che hai dei grandi prefissi, con dei buchi che vengono annunciati da altre parti. Abbiamo, negli alberi visti prima, tanti livelli che non possono essere eliminati con l'aggregazione CIDR, dovrebbero essere eliminati facendo una redistribuzione degli indirizzi IP sulla rete, cosa che è molto complicata (soprattutto adesso che gli indirizzi IP scarseggiano).

## Tempeste di BGP update

I router di backbone devono processare un insieme e enorme di update, e devono ricalcolare continuamente la tabella di instradamento sia a livello RIB che FIB. Ce la faranno i router in futuro a continuare a fare questo mestiere? È chiaro che se il router calcola la tabella di instradamento, mentre calcola la tabella non fa il mestiere del router, che è quello di smistare pacchetti. Di che numeri stiamo parlando?

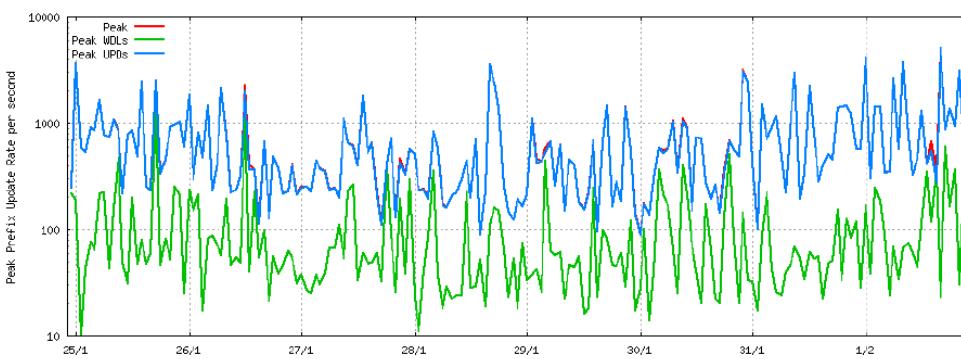


Figura 228 - BGP Update

Questi sono il numero di update al secondo (di picco) osservati nell'insieme dei router che potranno mettere sotto controllo. Questi update sono divisi in withdrawal (ritiri) e annunci. Possiamo vedere che i ritiri sono di meno, ma ci sono piuttosto molti update: 1000, anche 10000 al secondo nei picchi. Per ognuno di questi update il router si deve ricalcolare la tabella di instradamento. Per mitigare questo problema possiamo mettere sotto controllo un prefisso. Se questo prefisso cambia troppe volte possiamo usare l'algoritmo di dampening, assegnando delle penalità ad un prefisso che cambia troppo frequentemente, e marcandolo come "historical", cioè non più valido se la penalità diventava troppo alta. Non necessariamente però questa tecnica dà un effetto positivo. Da cosa dipende questo "rumore" che c'è continuamente su internet? Siamo su un router che vede come si raggiungono tutti i router del pianeta. I link vanno continuamente giù, per motivi assolutamente ragionevoli (guasti, riconfigurazioni, cambio di politiche). C'è anche un altro motivo, ed è un motivo che non abbiamo ancora visto: BGP, anche ben configurato, può oscillare, per un tempo indefinito, fra diversi stati (e quindi diverse raggiungibilità) di instradamento. Ad ogni stato corrispondono update a raffica.

## Esaурimento degli indirizzi IPv4

Che cosa vuol dire esaurimento degli indirizzi IPv4? Il pool degli indirizzi disponibili è sostanzialmente quasi vuoto. La maggior parte degli indirizzi possibili sono stati allocati, o sono riservati per motivi tecnici, e continuano ad essere riservati. Gli indirizzi stanno finendo molto rapidamente. L'insieme di indirizzi non ancora allocati è limitatissimo. Urge un intervento rapido per l'adozione di IPv6. C'è una raccomandazione agli ISP perché passino il più rapidamente possibile a IPv6, e c'è una raccomandazione ai governi, affinché spingano la comunità di internet a passare a IPv6, anche con incentivi.

## Variazioni sulla gerarchia di internet

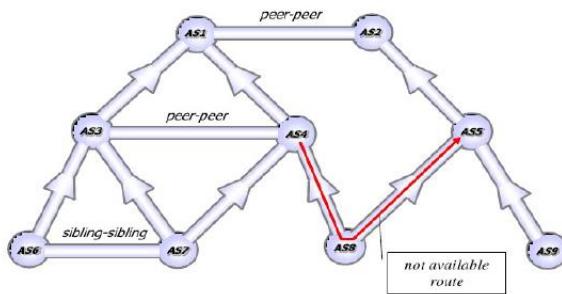


Figura 229 - Modello di internet

Questo è il modello di internet che ci siamo fatti nel tempo. Io, che sono AS6, pago AS3 per uscire su internet, gli consegno il mio traffico e ricevo il mio traffico da lui; questo in single homing, in dual homing...con una gerarchia stratificata su vari livelli. Qualche volta le relazioni non sono solo da customer a provider, ma da anche da pari a pari. Qualche volta le relazioni peer to peer vengono chiamate sibling to sibling, e si tratta semplicemente di relazioni peer to peer di bassissimo livello, come ad esempio quella tra due AS dello stesso proprietario. Nella gerarchia, delle rotte non sono disponibili. Perché? Quella è una rotta che va da AS4 ad AS5 passando per AS8. Ad AS8 non gli va di fare da transito tra AS4 e AS5, perché significherebbe che sta pagando AS4 e AS5 per fare da transito tra i due. Questi sono filtri sul traffico. Guardiamo con un po' di dettaglio a ciò che succede nelle relazioni peer to peer e customer provider:

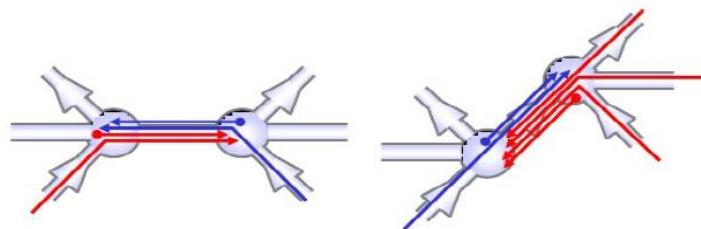


Figura 230 - Peer to Peer e Customer Provider

Nel peer to peer, i due AS si annunciano tra loro i propri prefissi e tutti i prefissi che gli arrivano dai propri customer. Nella gerarchia customer provider i due AS invece cosa si annunciano? Il customer annuncia al provider tutto ciò che ottiene dal livello più basso della gerarchia, i prefissi propri e i prefissi dei propri customer. Il provider annuncia al customer tutti gli indirizzi che gli vengono dall'alto (dai suoi provider), tutti i prefissi che gli arrivano dalle relazioni peer to peer e tutti gli indirizzi che gli arrivano dai propri customer, perché il customer deve avere una visione completa di internet. Cosa non c'è in questo disegno? Io non annuncio al mio provider ciò che proviene dai miei peer, perché nelle relazioni peer to peer c'è un accordo reciproco di annunciarsi soltanto i propri customer.

Questo è il modello che ci siamo fatti di internet:

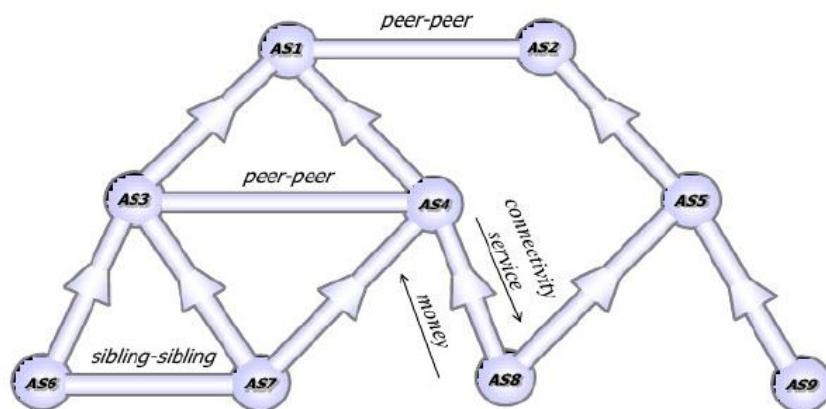


Figura 231 - Modello di internet

Con questo modello di fronte di capisce meglio cos'è la default free zone:

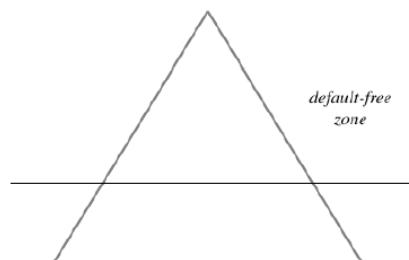


Figura 232 - Default free zone

Questa è una visione molto astratta della gerarchia, ma è proprio questa la situazione di internet? O internet sta cambiando? Internet sta cambiando pesantemente; la gerarchia più o meno rimane la stessa, ma a questa gerarchia si appiccicano dei content provider mettendo in piedi un numero enorme di relazioni peer to peer, con tutti i player, su tutti i diversi livelli della gerarchia. Questo sta cambiando la situazione complessiva dei rapporti di forza in internet.

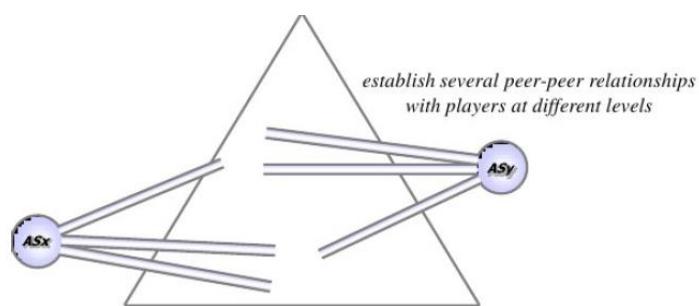


Figura 233 - Gerarchia con i content provider

Chi sono questi content provider? Per esempio Akamai, che si presenta a Links, dove sono presenti praticamente tutti i provider europei, e gli dice "qui c'è il mio AS, chi è che vuole fare peering con me? Peer to peer, non si paga niente, poi naturalmente tu hai tutti i miei contenuti immediatamente disponibili

dentro la tua rete, e questo traffico, che per te è in arrivo, non aggrava la bolletta del traffico che tu prendi dai tuoi provider". Dopodiché Akamai va in altri punti di scambio e fa la stessa cosa. Questa CDN si innesta trasversalmente sulla gerarchia di internet. Non soltanto Akamai fa peering a Londra con qualcuno e ad Amsterdam con qualcuno, ma magari mette insieme tantissimi peering con lo stesso provider, in modo tale che non soltanto il traffico arrivi immediatamente a quel provider, ma arrivi anche direttamente nella specifica porzione di AS che è vicina a quel punto di scambio di traffico. Dal punto di vista gerarchico, la rete diventa più piatta? Questa variazione della gerarchia può rendere più difficile entrare nel mercato per un piccolo provider?

### Robustezza dell'ecosistema di internet

Due incidenti abbastanza recenti che hanno condizionato il funzionamento di internet:

- Mediterranean Fiber Cable Cut (gennaio - febbraio 2008)
- YouTube Hijacking (24 febbraio 2008)

Internet, quanto è resistente a incidenti di questo genere? Per capirlo dobbiamo capire bene come è fatto l'ecosistema di internet, cosa ne fa parte, e capire quanto le singole componenti di questo ecosistema siano effettivamente robuste. Dal punto di vista architetturale, come è fatto l'ecosistema?

Ci sono:

- gli ISP e la loro specifica infrastruttura;
- le interconnessioni fra provider (che possono avvenire in luoghi pubblici oppure privati);
- gli Internet Exchange Point (dove si interconnettono molti provider);
- CDN (di importanza crescente);
- l'infrastruttura dei DNS.

Poi ci sono le componenti tecnologiche sottostanti, che possiamo vedere a vari livelli di astrazione, e uno può chiedersi per ciascuna di queste componenti di internet quale sia la robustezza. Possiamo vedere queste componenti a livello protocollare: noi sappiamo che ci sono dei protocolli fondamentali nella rete, e il firmware che implementa questi protocolli negli apparati. Questo firmware è molto spesso simile, perché in questo ecosistema i vendor che hanno posizione dominante (soprattutto nella parte di backbone della rete) sono pochissimi (2 o 3), e questo può porre delle sfide anche di robustezza. Se allarghiamo la prospettiva dobbiamo verificare se l'ecosistema complessivo regge anche dal punto di vista economico e sociale. È chiaro che tutto regge se reggono i profitti per i provider, se i profitti sono limitati questo mette in crisi uno dei componenti importanti dell'infrastruttura. Un altro aspetto fondamentale su cui internet è cresciuta nel tempo è l'apertura del mercato, che ha consentito grande competizione nei servizi (e questo ha alimentato la crescita di internet). Un altro elemento fondamentale è il trust tra i provider, e la loro voglia (e anche convenienza) nella cooperazione. Questo è un aspetto di quelli che colpiscono: ci sono molti provider che sono in competizione tra loro, interagiscono tutti attraverso BGP, ti aspetteresti di trovare in BGP dei presidi di sicurezza sofisticatissimi. Niente di tutto questo, quello che abbiamo visto di

BGP è sostanzialmente la sicurezza di BGP, c'è poco altro. Un altro elemento fondamentale è l'indipendenza dei provider, degli IXP ecc. dalle nazioni e dalle istituzioni di più alto livello. Questa parte di indipendenza diventa indipendenza anche nel veicolare i contenuti. Ci si può chiedere se tutte queste parti dell'ecosistema in futuro continueranno a reggere, ed in particolare ci si chiede quale sia il rischio di cascade failure in internet, cioè di failure significative che possano riguardare una grande parte di internet; potrebbero essere causate da hacker, attacchi terroristici o situazioni di guerra. In futuro tutto sommato gli hacker non saranno il problema, visto che è nello stesso interesse degli hacker che la rete continui a funzionare. I rischi invece derivanti da attacchi terroristici e guerre ci sono, e la rilevanza della rete cresce dal punto di vista dei servizi: ormai il cloud computing ci convince anche a mettere i propri dati in remoto, e questo fa crescere l'appetibilità per degli attacchi di natura terroristica oppure per la distruzione di una parte di internet. Bisogna anche fare i conti con i fenomeni naturali. Un'analisi recente su quello che accaduto in seguito al terremoto in Giappone, e relativo tsunami ci dice che le cose in termini di resistenza della rete sono messe in maniera positiva. La provincia di Sendai, quella maggiormente colpita, in effetti è stata isolata dalla rete per molto tempo (circa 15 ore), e gran parte dei collegamenti del Pacifico sono stati impattati dal terremoto. Però se guardi tutto dal punto di vista di BGP, dell'effettiva robustezza dei peering, soprattutto internazionali, non c'è tantissimo che si vede in BGP.

## Stabilità di un BGP

Data una configurazione BGP dei router in una rete, il routing è stabile? Rimane sempre lo stesso oppure può oscillare? Ci sono delle configurazioni che impediscono l'oscillazione? C'è un modello semplice che mi consente di studiare il problema? Nel caso di OSPF, una volta partito l'algoritmo converge verso una configurazione stabile, che cioè non cambia a meno di cambiamenti nella topologia. Stessa cosa per RIP.

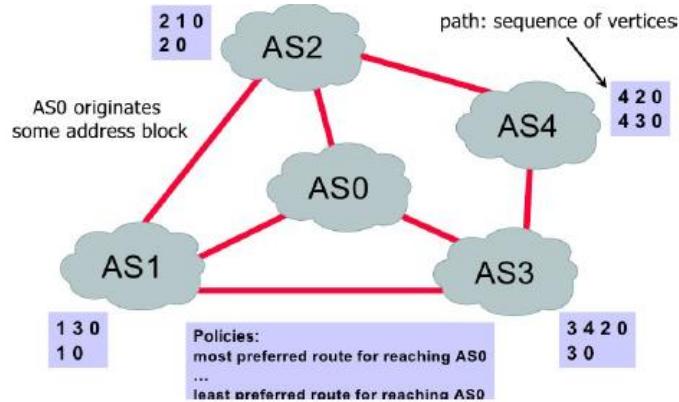


Figura 234 - Configurazione BGP

Prendiamo questa configurazione di BGP: è stabile? Qui c'è un AS, AS0 che origina un qualche prefisso (ne basta uno). Posso ignorare la presenza di altri prefissi nella rete, posso studiare quello che accade nella rete BGP discutendo ciò che accade su un singolo prefisso, indipendentemente da ciò che accade su tutti gli altri. Questo perché in BGP ciascun prefisso viene gestito separatamente da tutti gli altri. Quindi, se voglio studiare la stabilità di BGP, tanto vale studiare la stabilità di un prefisso. Nei rettangoli c'è una lista ordinata di path per raggiungere AS0. AS1 ad esempio preferisce fare la rotta 1-3-0, e in subordine la 1-0. Queste sono tutte e sole le rotte che considera. Ci si chiede: questa configurazione è stabile? Vediamo che succede in questa rete.

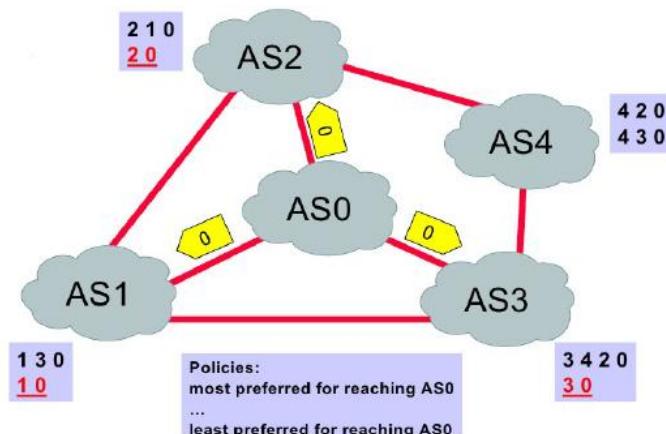


Figura 235 - Aggiornamento della rete

Ad un certo punto AS0 inizia ad annunciare sé stesso nelle 3 direzioni. Questo annuncio viene ricevuto da AS1, AS2 e AS3, che a questo punto scelgono la loro rotta per raggiungere o. AS3 sceglie 3-0, AS2 sceglie

2-0, AS1 sceglie 1-0, in questo momento è l'unica possibilità. Passa ancora un po' di tempo, e AS1-2-3 si annunciano l'un l'altro il fatto che hanno la raggiungibilità di un certo prefisso. 3 dice a 4 che attraverso di lui possono raggiungere 0, e una cosa analoga la fanno gli altri. Ulteriore update. Adesso 1, che prima diceva a 2 "attraverso di me si fa 1-0" adesso gli deve dire "attraverso di me si fa 1-3-0" (perché BGP propaga solo la best).

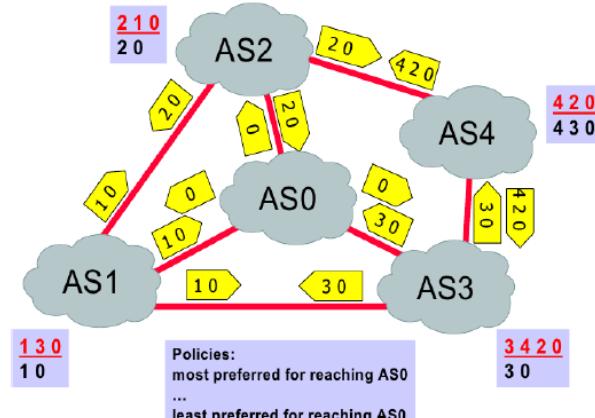


Figura 236 - Aggiornamento della rete

I router cambiano continuamente idea. Questo meccanismo è un meccanismo senza fine. Quella rete, configurata in quel modo, oscilla continuamente, e continua ad oscillare. Anche cambiando timing non si smette di oscillare. La rete oscilla per qualunque temporizzazione (magari oscilla in modo diverso, ma oscilla). Questo significa che ci sono delle configurazioni di BGP intrinsecamente instabili, che vanno evitate. Se è vero che la rete oscilla indipendentemente dalla temporizzazione degli eventi è anche vero che la temporizzazione può assumere un ruolo significativo. Quello visto si chiama "bad gadget". Come possiamo semplificarlo, ottenendo una rete che oscilla ancora più piccola? Qui si sceglie un cammino più lungo rispetto a quello più breve, ma non basta. Cos'altro c'è di tremendo in questa struttura? C'è una dipendenza ciclica: 2 conta su 1, 1 conta su 3, 3 conta su 4, 4 conta su 2.

### Disagree

Ci sono delle configurazioni in cui il routing oscilla continuamente e configurazioni stabili, ma anche configurazioni in cui c'è più di uno stato stabile e in funzione del timing c'è uno stato stabile oppure si può oscillare. Questa struttura si chiama "disagree":

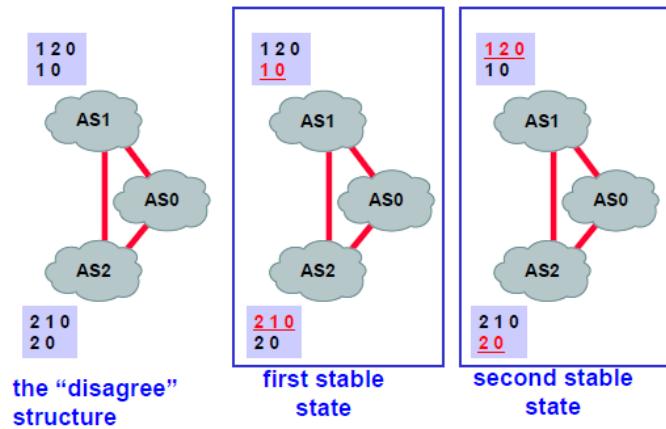


Figura 237 - Stati stabili del BGP

2 preferisce passare per 1 piuttosto che andare direttamente su 0, e 1 preferisce passare per 2 piuttosto che andare direttamente su 0. Supponiamo che l'annuncio arrivi subito a 1, che fa la scelta 1-0; l'annuncio, ancora per un attimo, non arriva a 2, ma che invece arrivi a 2 l'annuncio che passa per 1. 2 sceglie 2-1-0, e da lì non si schioda. L'alternativa è che succede esattamente l'opposto: arrivi prima 2, poi a 1 arriva l'annuncio di 2, e questo fa inchiodare il sistema su un altro stato stabile. Questa è la struttura disagree, e ha due stati stabili. Non ci vuole molto ad immaginare di replicare questa struttura in modo di avere quanti stati stabili si vogliono. Noi siamo preoccupati dalla presenza di più stati stabili, preferiremmo che ce ne sia uno. C'è un teorema che dice: se c'è più di uno stato stabile in una rete BGP sicuramente esiste una temporizzazione che fa oscillare la rete. La presenza di uno stato stabile quindi implica la presenza di temporizzazioni pericolose che possono far oscillare per un tempo indefinito il sistema

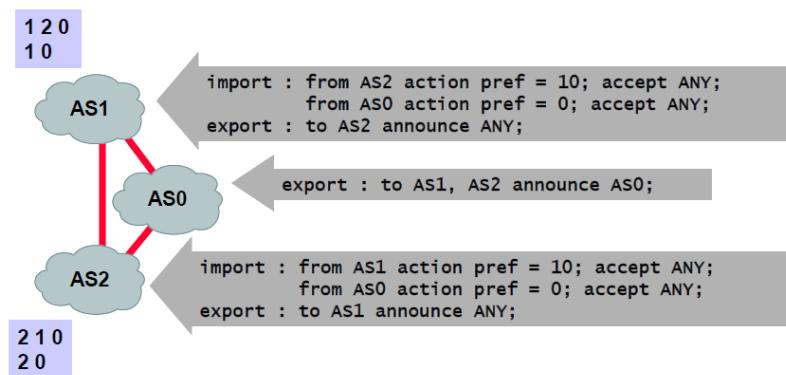


Figura 238 - Desagree

Queste sono delle configurazioni RPSL che implementano un disagree. La presenza di molti stati stabili fa sì che si possa avere un route triggering.

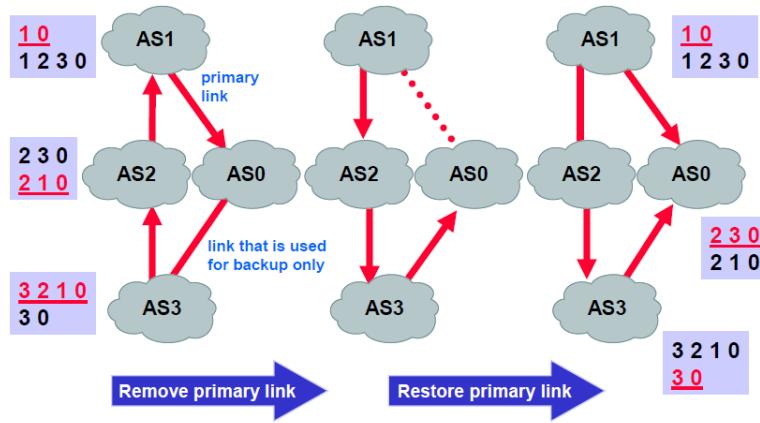


Figura 239 - Route trigger

Nella rete a sinistra c'è un link usato soltanto come backup. AS1 fa 1-0 e in subordine fa 1-2-3-0, ma lì c'è una linea di backup. AS2 fa 2-3-0 e poi in subordine fa 2-1-0. Le politiche funzionano in un certo istante, però possono essere messe in maniera un po' pericolosa. AS3 fa 3-2-1-0 e in subordine 3-0. Ad un certo punto un link va giù, e le politiche cambiano. Ad un certo punto c'è una restore di quel link, che torna operativo. A questo punto AS1 si prende l'annuncio su 1-0, ma AS2 rimane su 2-3-0, e AS3 su 3-0. A questo punto il routing si stabilizza in questo nuovo stato, che non è quello per cui tu avevi progettato la rete. Si era progettata la rete per avere un backup solamente sul link tra AS3 e AS0, e nel momento in cui il link primario dovesse essere nuovamente disponibile il routing dovrebbe tornare nella situazione iniziale, e invece di inchioda in questa situazione, che è una situazione intermedia.

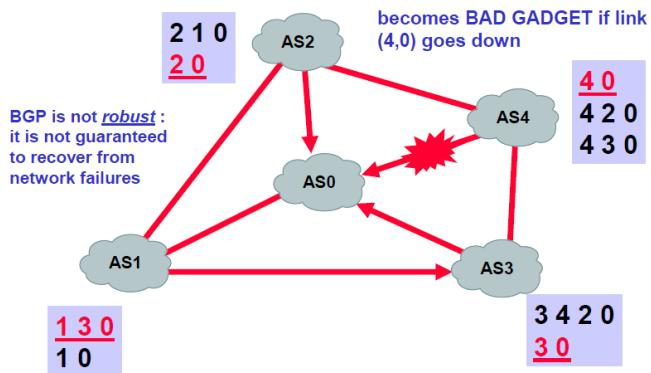


Figura 240 - Esempio di rete stabile

In questo esempio: c'è una rete che è perfettamente stabile. Stiamo guardando un bad gadget, ma con un qualcosa (AS4 che va verso AS0) che taglia le oscillazioni. Se questo link va giù 4 non può più fare 4-0, quindi la politica non è più utilizzabile. Le restanti politiche utilizzabili sono le politiche bad gadget. Questo vuol dire che BGP non è un protocollo robusto: anche se si trova una configurazione che non oscilla non è detto che questa configurazione rimanga stabile anche in seguito ad una failure di una porzione della rete.

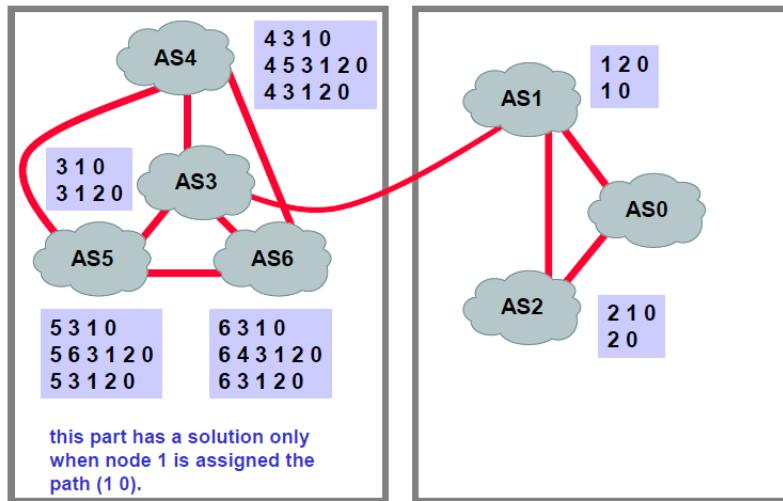


Figura 241 - Precarius

Questa è la struttura chiamata "precarious". Qui c'è un disagree. Le politiche della parte sinistra della rete consentono ad AS3 di scegliere 3-1-0. Questa soluzione funziona solo se al nodo 1 viene assegnato il percorso 1-0. Se AS3 va su 3-1-2-0 le prime righe dei router esterni (di AS4, AS5 e AS6) sono impraticabili, quindi i router devono passare per il vicino come prima possibilità, e come seconda si passa per la strada "diretta". Anche qui dentro si instaura un bad gadget. Se manca questa possibilità di andare dritti verso 1 parte un'oscillazione, quindi in funzione dello stato stabile che ha preso una parte della rete, l'altra parte può oscillare oppure no.

### Aspetti computazionali della stabilità di BGP (SPP, Stable Path Problem)

La questione di accertarsi che una configurazione BGP sia stabile oppure no ha certamente un costo. Questo problema è NP-completo, quindi è problematico verificarlo effettivamente. SPP appartiene ad NP, quindi esiste una macchina di Turing non deterministica che lo risolve in tempo polinomiale, e per ogni altro problema appartenente ad NP, questo problema deve essere Carp riducibile polinomialmente a questo problema. Per mostrare che SPP appartiene ad NP dobbiamo enumerare tutti gli stati del routing e verificare la stabilità di ognuno. Per fare la dimostrazione di completezza ci appoggiamo ad un problema che sappiamo essere NP-completo, 3-sat, e facciamo una riduzione di 3-sat a SPP; detta in altri termini, risolviamo 3-sat con SPP. Se ci si riesce vuol dire che anche SPP è NP-completo. È data un'istanza di 3-sat, che viene mappata su un'istanza di SPP. Ogni variabile di 3-sat si modella come una struttura disagree, che sembra proprio fatta apposta per assumere due stati, vero o falso, si passa per di sopra oppure si passa per di sotto. Gli stati stabili di questo disagree rappresentano i due valori di verità e falsità. Adesso associamo ogni clausola con un nodo e utilizziamo il bad gadget per far sì che il sistema sia instabile se non c'è soluzione per l'istanza di 3-sat.

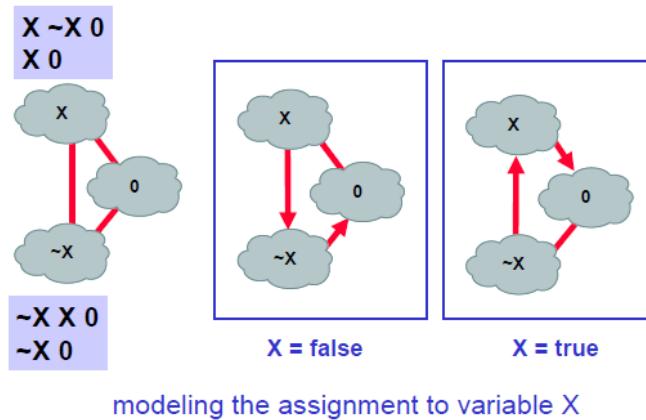


Figura 242 - Proof

Il disagree rappresenta la variabile X. Gli AS vengono chiamati X e  $\sim X$ . Posso fare allora queste politiche:

X sceglie X  $\sim X$  o  $\Rightarrow$  in subordine X o;

$\sim X$  sceglie  $\sim X$  X o  $\Rightarrow$  in subordine  $\sim X$  o.

Convenzionalmente, quello che dico è: se quello che si sceglie è di passare preferenzialmente per  $\sim X$  allora X è falso (perché  $\sim X$  è vero), altrimenti, se preferenzialmente si passa per X, allora X è falso (perché risulta vero  $\sim X$ ).

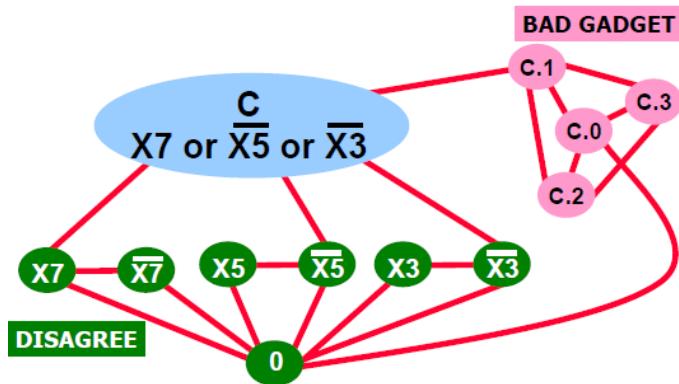


Figura 243 - Proof

Alla clausola  $C$  assegniamo le variabili  $X_7$ ,  $X_5$  e  $X_3$  ( $X_5$  e  $X_3$  appaiono in forma negata).  $X_7$  appare in forma diretta, quindi prendo la clausola e la attacco alla parte positiva al disagree per  $X_7$ ,  $X_3$  e  $X_5$  appaiono in forma negata, quindi li collego alla parte negativa dei disagree associati. Poi prendo un bad gadget (ultra semplificato, quello che ha soltanto 3 nodi all'esterno e un nodo al centro) e lo associo ad ogni clausola. Questo bad gadget è una specie di campanello, che si mette a suonare se la struttura è instabile. Se non diciamo quali sono le politiche però non diciamo niente. Le politiche sul disagree le conosciamo. La clausola passerebbe volentieri per  $\sim X_5$ -o, oppure passerebbe per  $\sim X_3$ -o oppure per  $X_7$ -o. Queste politiche le possiamo mettere in qualunque ordine. Affinché si passi per  $X_7$ -o c'è bisogno che  $X_7$  sia vero, altrimenti si passerebbe per  $\sim X_7$ -o; se non fosse così, la clausola non avrebbe a disposizione il cammino  $X_7$ -o, e per

essere soddisfatta la clausola ne ha bisogno. Stessa cosa per  $\sim X_5$ ,  $X_5$  deve essere falsa, oppure deve essere falsa  $X_3$  per poter passare per  $\sim X_3$ -o. La clausola si appoggia a una di queste tre variabili per andare dritta. Se la clausola riesce ad andare dritta allora passerebbe per C.1 e per una di quelle tre possibilità. Se la clausola non è soddisfatta allora abbiamo  $[X_7 \ X_5 \ X_3] = [\text{falso} \ \text{vero} \ \text{vero}]$ , non si passa per o. Se non si passa per o la prima, la seconda e la terza scelta in C.1 non sono praticabili, e ci sono soltanto la penultima e l'ultima scelta, e rimangono la penultima e ultima scelta anche per C.2 e C.3. Guardiamo queste clausole: passo per il vicino, e se non è disponibile vado dritto. Questo è un bad gadget che si mette a oscillare. Se la clausola è soddisfatta non oscilla più. Che succede se ci sono più clausole contemporaneamente?

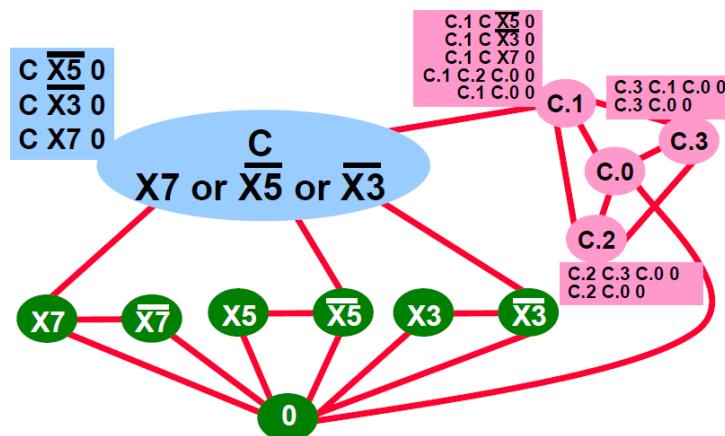
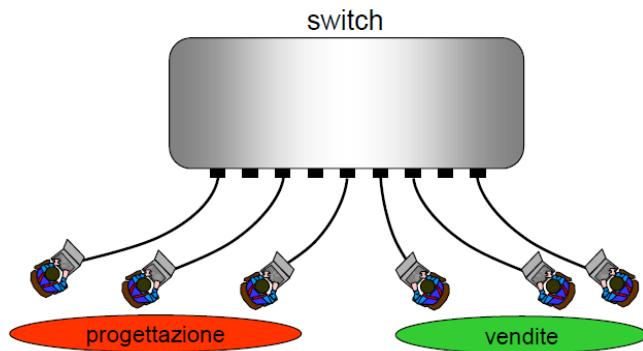


Figura 244 - Proof

Attacco ciascuna clausola alle variabili che le servono, poi vale quanto detto prima.

## Reti locali virtuali (VLAN)

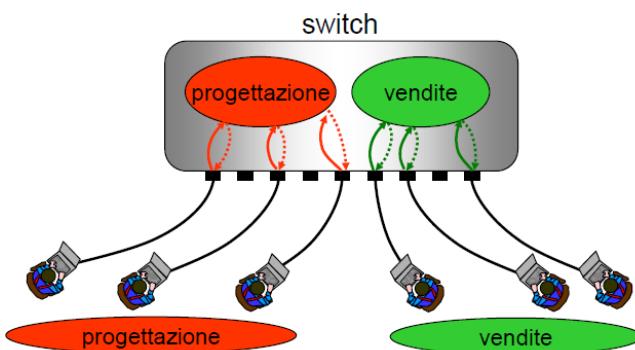


*Figura 245 - Esempio di VLAN*

Perché molto spesso in rete locale si utilizzano delle VLAN? Questo è un esempio molto semplice che fa capire quali possono essere le motivazioni. Un'azienda ha una LAN, e ha comprato un singolo switch a cui sono connessi vari calcolatori di due diversi dipartimenti. L'azienda vuole suddividere la rete in due LAN separate settori diversi, per motivi di sicurezza o anche solo per dividere il traffico broadcast (il traffico broadcast in una LAN sta nelle richieste ARP). Una soluzione possibile è acquistare un nuovo switch e distribuire i calcolatori su due switch, si mette il ramo vendite su uno switch e il ramo progettazione sull'altro. Questa soluzione è poco flessibile, poco adattabile ai cambiamenti, poco economica. Non è detto che i dipartimenti siano fisicamente allocati in zone omogenee (tutto il dipartimento A è al primo piano, tutto B al secondo, la soluzione dei due switch qui può essere ancora ragionevole), ma questa gente potrebbe essere distribuita nell'infrastruttura: non è pensabile far passare un filo per tutta la rete fino a raggiungere lo switch giusto a cui collegare il calcolatore. Le VLAN nascono per questo.

## VLAN su un solo switch

Come realizzare vari “switch virtuali” avendo a disposizione un singolo switch? L'idea è quella di definire una topologia logica che sia indipendente da quella fisica, mettendo sullo stesso switch diverse LAN virtuali (naturalmente lo switch deve essere tecnologicamente adeguato per realizzare meccanismi di questo genere).



*Figura 246 - VLAN con un solo switch*

Io conetto le varie porzioni della rete su porte diverse, e sulle porte vado a scrivere in qualche modo che una certa porta è relativa ad una certa LAN virtuale. Le frecce rappresentano il traffico che entra nella VLAN e che esce da essa. L'idea della VLAN è quella che sta alla base di tutte le reti virtuali: costruire un'infrastruttura logica su un'infrastruttura fisica che può essere anche molto diversa da quella logica. Vorremmo che il traffico di una VLAN sia completamente separato da quello delle altre. Le VLAN sono configurate dall'amministratore. Si possono definire le VLAN semplicemente in funzione della porta, ad esempio:

- le porte 1, 3, 4 e 7 appartengono alla VLAN rossa;
- le porte 2, 5 e 6 alla VLAN blu;
- le porte 8 e 9 alla VLAN arancione.

Lo switch a questo punto tecnologicamente in grado di tenere completamente separati i traffici delle diverse VLAN. La situazione è più complessa: io posso definire le VLAN anche in funzione del contenuto del pacchetto, ad esempio MAC address, protocollo di livello 3, ecc. Posso dire "dentro questa VLAN ci sono solo questi MAC address", oppure "dentro questa VLAN c'è solo il traffico IP". Diamo una semantica prevista per le VLAN: una VLAN denota un insieme di pacchetti che sono in transito su un certo switch. Posso quindi definire le VLAN come LAN rossa: tutti i pacchetti che entrano dalla porta 1, dalla porta 3, dalla porta 4 o dalla porta 7 la restrizione che si fa è che la semantica della VLAN indichi la partizione dei pacchetti, non ci deve essere ambiguità nella definizione. Le regole che descrivono le VLAN (specificabili con un linguaggio che dipende dal costruttore dello switch) attribuiscono ciascun pacchetto ad una VLAN e quindi l'insieme dei pacchetti che entrano in uno switch è partizionato in VLAN. Se l'amministratore non definisce nessuna VLAN tutti i pacchetti appartengono alla stessa VLAN e il comportamento è quello visto finora. Come si fa la configurazione di una VLAN?

- considera i pacchetti che arrivano allo switch da un certo insieme di porte (ingress port);
- di questi pacchetti, fanno parte della VLAN solo quelli con certe caratteristiche (es. tutti o solo quelli con certi MAC address o solo quelli che portano a bordo il protocollo ip o ...);
- i pacchetti che fanno parte della VLAN possono uscire dallo switch solo attraverso certe porte (egress port).

Finora abbiamo rappresentato le porte di ingresso e di uscita con tratti differenti. Nell'esempio iniziale tutte le ingress port sono anche egress port, e tutti i pacchetti ricevuti su un certo insieme di porte partecipano alla VLAN; vedremo esempi più complessi. Un pacchetto viene associato ad una VLAN nell'istante in cui entra in uno switch. Le VLAN sono identificate da un intero compreso tra 1 e 4094 (la VLAN di default ha id 1). Ad una VLAN è spesso possibile anche dare un nome di 32 caratteri alfanumerici, più facile da ricordare. Questo numero di LAN virtuali è troppo basso: pensiamo ad un internet data center: abbiamo tantissimi clienti, altro che 4000. Che cosa si fa? Potremmo dare a ciascun cliente una LAN fisica, ma costerebbe tantissimo. Possiamo dargli una VLAN, ma quello non si accontenta, per motivi di sicurezza non è ragionevole mettere tutto dentro una sola VLAN, ed ecco che 5 VLAN se ne sono andate.

Magari poi serve anche una VLAN di management, che si usa solo per entrare nelle macchine e configurarle. Ecco che 4094 finisce in un soffio, e questo sta creando enormi problemi negli internet data center, e si è costretti a partizionare in modo opportuno i clienti. Aggiungiamo il livello 3 a questa storia. Che i reparti progettazione e vendite siano proprio completamente separati non è una buona idea. Possiamo collegare allo switch un router, che ha 3 porte: 2 le attacco su due porte dello switch (una è una porta “rossa” e una è una porta “verde”), e la terza la mando verso fuori. A questo punto questi oggetti, dal punto di vista del router, è come se fossero due switch diversi, anche se le due reti stanno dentro lo stesso switch. Attribuendo prefissi diversi alle VLAN abbiamo due LAN diverse in tutto e per tutto, e questo si riflette nell'instradamento:

routing table (router con 3 interfacce)			
network	nmask	nexthop	int
200.2.2.0	255.255.255.128	d.c	eth0
200.2.2.128	255.255.255.128	d.c	eth1
0.0.0.0	0.0.0.0	193.0.0.2	eth2

Figura 247 - Tabella di instradamento

A questo punto abbiamo separato il traffico ARP, che non passa dentro al router. Cosa succede quando uno switch deve inoltrare un pacchetto? Il pacchetto ricevuto viene associato alla giusta VLAN (usando le regole di configurazione) dopodiché lo switch individua la porta dello switch da utilizzare per poter trasmettere il pacchetto (accedendo al filtering database), per poi trasmettere (eventualmente) il pacchetto.

#### Filtering database

Gli switch che realizzano VLAN possono lavorare in due modalità alternative:

- IVL, Independent VLAN Learning: un filtering database separato per ogni VLAN;
- SVL, Shared VLAN Learning: un solo filtering database condiviso tra tutte le VLAN.

Due possibilità per la trasmissione del pacchetto:

- la ricerca nel filtering database non è andata a buon fine ovvero non è stata individuata nessuna porta: lo switch trasmette il pacchetto in broadcast su tutte le egress port associate alla VLAN di appartenenza del pacchetto;
- la ricerca nel filtering database ha individuato una porta: lo switch, prima di trasmettere il pacchetto, controlla che la porta individuata sia stata configurata come egress port per la VLAN del pacchetto; in caso affermativo il pacchetto viene trasmesso, altrimenti viene scartato.

#### VLAN simmetriche e asimmetriche

Finora abbiamo fatto solo esempi di VLAN simmetriche. In queste VLAN le ingress e le egress port coincidono. Nelle VLAN asimmetriche l'insieme delle egress port è diverso da quello delle ingress port. Quand'è che è interessante mescolare ingress e egress port? Per esempio, guardiamo questo:

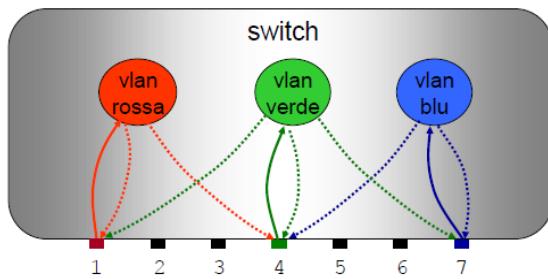


Figura 248 - Connessioni virtuali asimmetriche

Che tipi di traffico sono consentiti, e che tipi sono proibiti? Un pacchetto che viene originato nella VLAN blu non può raggiungere la rossa, e viceversa un pacchetto originato dalla rossa non può raggiungere la blu. La VLAN verde invece può essere raggiunta (e raggiungere) sia dalla rossa che dalla blu. Cosa posso farci con questa architettura? Tornando all'esempio visto all'inizio, in cui c'è un reparto progettazione e un reparto vendite dell'azienda. Potremmo avere anche un terzo reparto che offre servizi (ad esempio il reparto in cui ci sono i web server aziendali, i server di posta elettronica. Si fa allora in modo tale che quei server siano accessibili a tutti, indipendentemente dal reparto, e si mettono nella VLAN verde (quindi tutti ci possono accedere e tutti possono ricevere traffico da quei server); le VLAN relative ai singoli reparti sono tutte quante separate, e fra loro non possono interagire. Questa cosa non è male dal punto di vista applicativo. Un'architettura del genere non è detto che possa essere realizzata solo con VLAN, ma questa è una scelta di configurazione più efficiente (la si realizza facilmente) e più economica (abbiamo comprato un solo apparato anziché 4).

### Connessioni virtuali asimmetriche

Una VLAN è asimmetrica quando una porta egress per una VLAN è una port ingress per un'altra e viceversa (la configurazione simmetrica comunque è quella che si trova nel 99% dei casi).

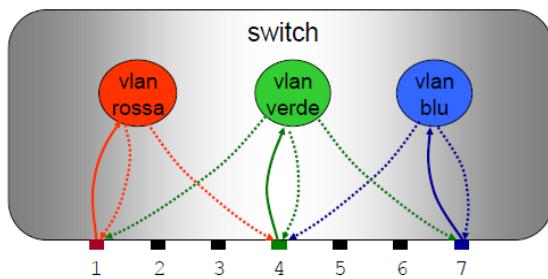


Figura 249 - Connessioni virtuali asimmetriche

Questa configurazione è fatta apposta per avere una VLAN verde che può comunicare con tutte le altre VLAN, però ci serve anche per piazzare su questa infrastruttura dei servizi condivisi, dopodiché mettiamo le egress port della VLAN rossa e della VLAN blu su quella porta verde, e questo fa sì che tutte le VLAN che stanno in quelle VLAN possano condividere i servizi offerti dalle macchine che stanno sulla VLAN

verde, pur non potendo comunicare tra loro (le porte che consentirebbero di comunicare tra la VLAN blu e la VLAN rossa in realtà non lo consentono). Questi:

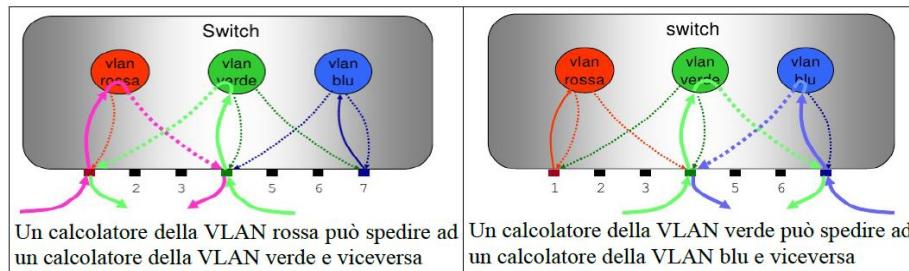


Figura 250 - I passaggi del traffico VLAN

sono i passaggi di possibili traffici sulle VLAN. La situazione complessiva è la seguente:

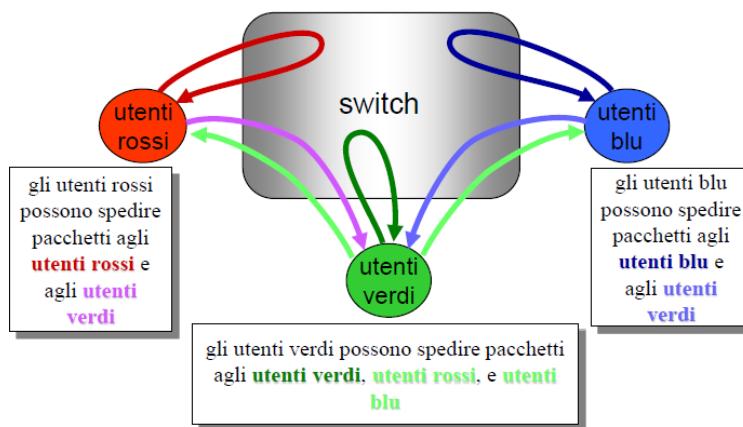


Figura 251 - Situazione complessiva

Questo ci consente di realizzare una struttura di VLAN con servizi condivisi senza dover acquistare un router. Rapporto tra VLAN asimmetriche e filtering database: in presenza di VLAN asimmetriche è preferibile usare switch in modalità SVL. Prendiamo questa semplice esempio di VLAN asimmetrica:

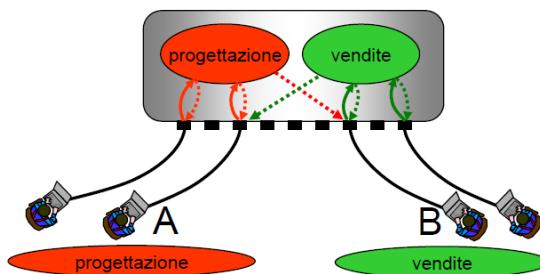


Figura 252 - Esempio di switch

Questo switch è configurato in modo che le macchine A e B possano parlare tra loro. A invia un pacchetto a B. Questo pacchetto viene classificato come appartenente alla VLAN rossa. Supponiamo che lo switch operi in modalità IVL. Il mac di B non viene trovato nel filtering database associato alla VLAN rossa e

viene inviato su tutte le porte egress della VLAN rossa. I pacchetti arriveranno oltre che a B anche a tutte le macchine della VLAN rossa (degrado delle prestazioni). Se lo switch invece opera in modalità SVL il mac address di B viene individuato (o meglio viene individuata la porta ad esso associata) ed i pacchetti vengono inviati solamente a B. Questo vuol dire anche che utilizzare le forwarding table in modalità IVL pare un po' una forzatura sulla semantica delle VLAN. La tabella di forwarding che ci aspettiamo è una semplice tabella di corrispondenza porta-mac. L'indicazione della VLAN non serve a nulla, perché abbiamo una tabella che ci dice quali sono le egress port per le diverse VLAN quando si cerca un MAC address si trova una porta, e in quel momento ci si chiede se quella porta è tra le ingress port di quella VLAN, e si fa il forwarding. Questo è consistente con l'idea di SVL.

### VLAN su reti con più switch

Finora abbiamo fatto riferimento ad una lan con un singolo switch, ma su una lan gli switch sono tanti; dobbiamo definire una VLAN anche in questo caso più complesso. Abbiamo una rete con tanti switch, connessi tra loro e su questa rete si costruisce sopra una LAN virtuale che attraversa le varie componenti delle reti fisiche (e questo coincide un po' con l'idea di virtualizzazione, da cui siamo partiti).

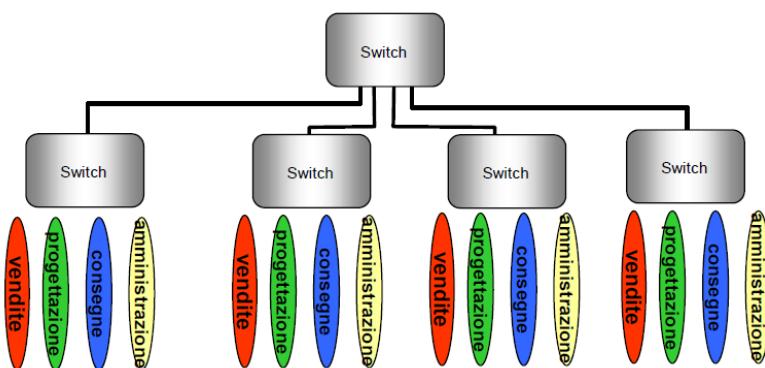


Figura 253 - LAN con più switch

Su ciascuno switch sono attestati calcolatori che afferiscono a diversi dipartimenti. Ci sono degli switch di distribuzione (collocati verso la periferia della rete) e degli switch di backbone, collegati agli switch di distribuzione. Non sappiamo come un pacchetto "rosso" possa finire su una rete "rossa" attestata presso un'altro switch. In questo esempio potrebbe essere utile avere 4 VLAN: vendite, progettazione, consegne, amministrazione. Prima possibile soluzione: prendo una porta che è sia egress che ingress per la VLAN verde su uno switch e la collego ad una porta che è sia egress che ingress per la VLAN verde sull'altro switch, e faccio la stessa cosa per la VLAN rossa. Purtroppo però su questa rete è sempre attivo uno spanning tree, quindi una porta viene messa in blocco.

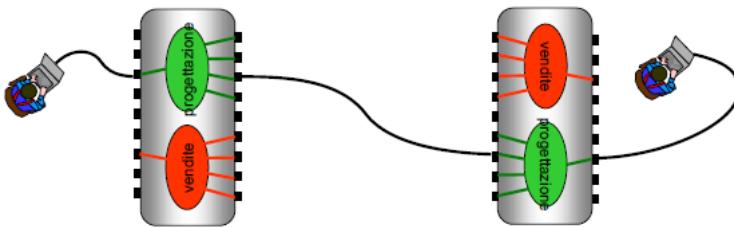


Figura 254 - Caso di due switch

sempre attivo, e sembra ignorare completamente la presenza delle VLAN. Ci viene in soccorso un altro standard, IEEE 802.1Q. Questo standard non parla proprio di VLAN, ma di trunk: in un collegamento tra due switch possiamo dichiarare un trunk 1q (cioè consistente con 802.1Q), su cui possono transitare pacchetti di diverse VLAN. Le porte in questione sono sia ingress che egress per tutte le VLAN. I pacchetti che attraversano un trunk 1q sono etichettati dallo switch trasmittente con l'identificatore della VLAN a cui appartengono. Questo è un campo addizionale della PDU di livello 2; i pacchetti vengono "colorati" col colore della VLAN di appartenenza. Questa tecnica è stata sostanzialmente generata per le tecnologie di virtualizzazione nelle reti. Un modo molto semplice per dire "questo pacchetto appartiene ad una struttura specifica" è quello di scriverci sopra un colore. Lo switch che riceve da un trunk 1q un pacchetto tagged lo attribuisce alla VLAN a cui appartiene, e il tag viene rimosso dal switch che riceve, quindi gli utenti non lo vedono. Così facendo non costringiamo le schede di rete ad imparare questa etichettatura (che comunque non le interessa). Una porta trunk 1q partecipa per default a tutte le VLAN come egress; questo default può essere "amplificato" in termini di configurazione. Questo ci dice che possiamo per default fare un trunk che ospita tutte le VLAN, ma possiamo anche, con un'opportuna configurazione ospitare in un trunk solo alcune delle VLAN che operano sugli switch. Quando si parla di trunk si parla del collegamento tra due switch. I trunk 1q servono a realizzare le VLAN, ma esistono anche altri meccanismi di trunking, e sono meccanismi in cui si possono prendere varie porte di uno switch, varie porte di un altro switch e collegarle una ad una. A questo punto, se si ha una linea a 100, si fa  $100+100+100\dots$  un'aggregazione di banda. Queste sono tecniche di trunking che servono ad aggregare traffico, ed hanno il loro standard specifico. La cosa si può complicare un po': quando una porta è indicata come egress port per una VLAN si può specificare se i pacchetti in uscita devono essere etichettati come appartenenti alla VLAN. Questo in generale permette di usare un filo che per qualcuno è trunk, per altri no. Quando una porta riceve un pacchetto:

- se è etichettato con una VLAN viene attribuito a quella VLAN;
- se non è etichettato viene attribuito ad una VLAN secondo le regole definite per quella porta.

Lo schema quindi di base è semplice, ma si può complicare a piacere. È tipicamente possibile configurare una porta in modo tale che i pacchetti in ingresso, etichettati oppure no, possano essere scartati. Quindi, ulteriore variante: abbiamo questo trunk, lo vogliamo gestire come trunk, sospettiamo che qualcuno

potrebbe infilarci dei pacchetti che provengono da macchine "normali". Quei pacchetti si prendono e si buttano. Coerentemente con le definizioni precedenti una porta può essere:

- *access*: riceve ed invia solo pacchetti non tagged;
- *trunk*: riceve ed invia solo pacchetti tagged;
- *ibrida*: riceve e/o invia pacchetti tagged e/o non tagged.

### Tag IEEE 802.1Q

La specifica si trova nello standard 802.3ac. Perché? 802.1q non specifica soltanto quello che succede su Ethernet, ma dà delle specifiche che sono vere per tutte le tipologie di rete locale. Ciò che riguarda Ethernet sta in 802.3ac. In questo standard c'è un campo di 2 byte, che, quando assume il valore 81-00 indica che il campo length/type (IEEE 802.3/Ethernet 2.0) è seguito da 2 byte di tag. Il tag contiene anche informazioni di priorità, definite dallo standard 802.1p. Come conseguenza la dimensione massima del pacchetto passa da 1518 byte a 1522 byte.

dst	src	type	payload	crc
(6)	(6)	(2)	(46-1500)	(4)

Questo è il pacchetto Ethernet che siamo abituati a vedere. Quando si ha il tag 802.3ac a bordo c'è un pezzetto in più:

dst	src	type	tag	type	payload	crc
(6)	(6)	81-00	(2)	(2)	(46-1500)	(4)

Se invece ci si vuole riferire a pacchetti più standard, "802.3 doc", con dentro LLC (802.2) la situazione è questa:

dst	src	size	dsap	ssap	ctrl	payload	crc
(6)	(6)	(2)	(1)	(1)	(1)	(43-1497)	(4)

Questo qui è il pacchetto standard con DSAP e SSAP, campo di controllo e così via. A questo punto qui ci si infilano type e tag e si procede normalmente:

dst	src	type	tag	size	dsap	ssap	ctrl	payload	crc
(6)	(6)	81-00	(2)	(2)	(1)	(1)	(1)	(43-1497)	(4)

### La LAN di un edificio

A questo punto siamo pronti per vedere una lan di un edificio abbastanza complessa.

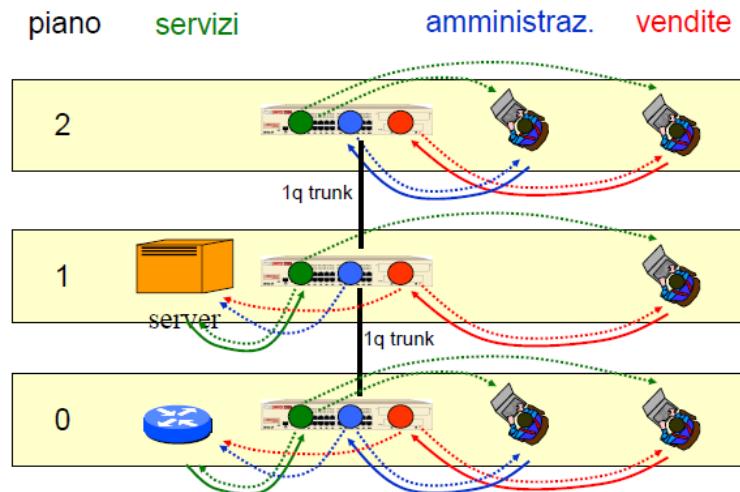


Figura 255 - LAN

Con che idea è realizzata questa LAN? È la stessa struttura vista nell'esempio "semplice": c'è una VLAN che offre servizi, accessibile a tutti, poi ci sono delle VLAN specifiche per ciascun dipartimento. Tutto questo, però invece di stare su un singolo switch sta su tutti gli switch, e ci sono dei trunk 1q che collegano i vari switch, attraverso i quali possono passare pacchetti verdi, blu o rossi. Questo consente di realizzare un'unica infrastruttura complessa attraverso diversi switch.

## MPLS VPN

### Motivazioni

Problema tipico di un customer: una società privata, o una PA, ha varie sedi, vari uffici, e vorrebbe connettere il tutto in una rete IP. Vorrebbe avere dei fili che connettono i suoi site. Abbiamo visto come ci si scambiano dati in internet, come si fa routing, ma poi come è fatta l'infrastruttura interna di una PA o di una società con molte sedi? Questo è il problema che affrontiamo: il customer ha tante sedi e vuole connettere. L'obiettivo del customer è avere una intranet locale. Come prima alternativa il customer può acquistare materiale per mettere su una rete fisica, ma questo atteggiamento rischia di essere poco economico, e rischia di non scalare. Guardiamo le cose dal punto di vista del provider. Un provider possiede un'infrastruttura, che va costruita in questo modo: tipicamente un provider ha un'infrastruttura che corrisponde a dei Point of Presence (PoP), e vorrebbe sfruttare questi pop per offrire connettività IP ai propri customer. Il suo obiettivo è di vendere "fili", ma virtuali, perché i fili virtuali sono più facilmente configurabili, e rispondono in maniera più dinamica alla domanda. Questo provider vorrebbe vendere fili virtuali dove possono fluire pacchetti IP verso i suoi customer. Il provider ha un'infrastruttura IP già dispiegata. Che vuol dire "infrastruttura IP già dispiegata"? Vuol dire che hai PoP, dentro i PoP hai dei router ridondanti, questi router volgono verso un backbone IP, sul backbone c'è routing, che può essere fatto con OSPF o RIP. Come possiamo sfruttare questa infrastruttura?

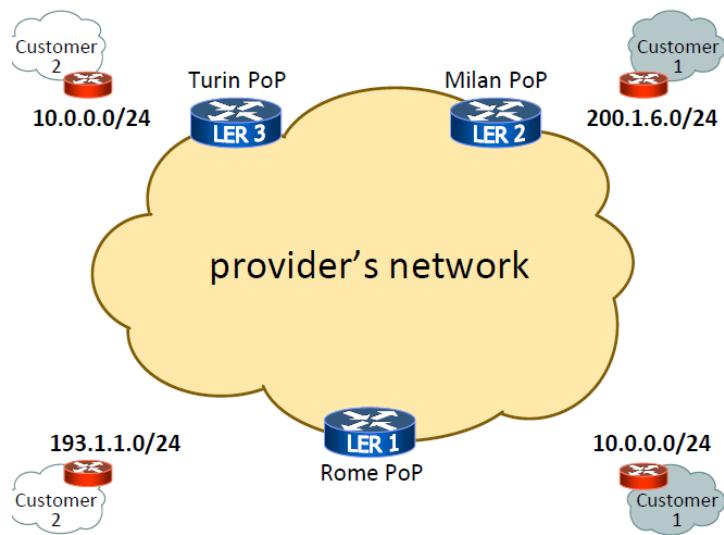


Figura 256 - Provider and customers

Questa è una rete che usiamo in tutti i prossimi esempi. Il provider ha 3 PoP, a Roma, Torino e Milano, e ci sono due customer. C'è un customer numero 1, che ha una sede a Roma e una a Milano, e c'è un customer numero 2 che ha una sede a Roma e una sede a Torino. Il customer vuole avere un piano di indirizzamento IP, e se ne frega dei piani di indirizzamento degli altri customer; non solo: si parla di indirizzi in classe 10, quindi ci si può fare veramente ciò che si vuole, gli indirizzi si possono sovrapporre, perché non

vengono annunciati in internet. Il customer inoltre non vuole avere nessuna interferenza col piano di indirizzamento del provider. I router della rete del provider in figura sono etichettati come LER.

### Vincoli

#### *Vincoli del customer*

Gli indirizzi devono rimanere invariati; tipicamente i customer vogliono avere QoS. C'è un altro aspetto, quello dell'isolamento del traffico dei customer: un customer vuole che il suo traffico non sia mescolato con quello di altri, che il suo traffico non possa essere visto da nessuno. Questo significa che la virtualizzazione deve essere in grado di non mescolare questi traffici.

#### *Vincoli del provider*

Un provider vuole avere costi di configurazione, manutenzione e gestione che siano i più bassi possibile. La configurazione non può essere onerosa, e non può essere onerosa la gestione di questa configurazione. Inoltre non vuoi che questa infrastruttura che stai costruendo pesi, al punto di influenzare le prestazioni dell'intera infrastruttura. Le prestazioni del backbone dovrebbero dipendere soltanto dal traffico, e non dal peso della configurazione. C'è un altro attore in questa storia: i vendor. L'obiettivo del vendor è vendere tanti router. L'idea è che uno possa vendere macchine costose, macchine carrier grade (quelle che hanno prestazioni migliori, e che quindi costano di più).

## MPLS

Guardiamo il disegno di prima, con l'occhio delle vecchie tecnologie di rete. Una volta avremmo fatto queste cose utilizzando tecnologie come ATM. Ma queste sono tecnologie vecchie, che hanno già saturato il mercato, e i vendor lo sanno; di macchine con tecnologie ATM, o ATM-like, i vendor ne hanno già vendute tantissime, e quindi perché non provare con una nuova tecnologia, che sia di moda, che consenta a questo punto di fare un nuovo business? Il punto di incontro tra customer, provider e vendor sono le VPN realizzate con MPLS (Multi Protocol Label Switching). È un punto di incontro perché cioè che questa tecnologia permette di fare sono delle VPN che risolvono i problemi dei customer, i provider riescono a configurarle a costo quasi zero, e i vendor vendono tanti apparati. Questo protocollo è fortemente scalabile, indipendente dal protocollo, data carrying. La prima cosa da fare è collocare MPLS nella pila ISO-OSI:

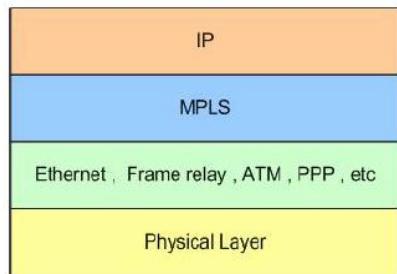


Figura 257 - MPLS vs OSI

Questa cosa mette un po' in difficoltà, alcuni lo definiscono "un protocollo di livello 2 e mezzo", si colloca tra i livello 2 e il livello 3. I pacchetti MPLS encapsulano pacchetti di livello più alto dentro un header MPLS dentro un stack di label. Qui dentro c'è una pila di label, ogni label è un colore, ogni colore contiene 4 campi: valore (il colore specifico), una traffic class per QoS e ECN (Explicit Congestion Notification), un bottom of stack flag e un TTL. Quindi prendi i pacchetti di livello più alto e li metti dentro un header che non è altro che uno stack ambulante, una pila che si sposta nella rete. Per raccontarci meglio le cose dobbiamo tornare al nostro scenario di lavoro.

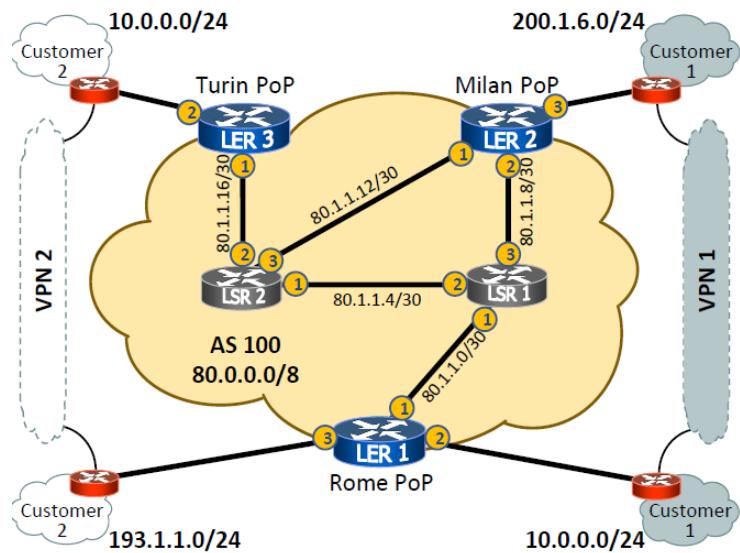


Figura 258 - Interconnection - scenario

Un po' di terminologia: quelli che vediamo disegnati sono sempre router, ma nel gergo delle VPN si chiamano con delle sigle particolari:

- CE, Customer Edge router; sono i router di frontiera dei customer.
- PE, Provider Edge router.
- P, provider router.

La filosofia qui è questa: tu sei un customer che ha una sua sede? La prima cosa che devi fare è avere un router che ti porta verso la rete del provider, Customer Edge. Poi se ti sposti verso la rete del provider, il

router che ti ci fa accedere è il Provider Edge. I router del provider, quelli che non si interfacciano con il resto del mondo, sono i provider P. Questo però non giustifica i nomi LER e LSR.

#### How-to

L'obiettivo adesso è dare scacco alle VPN in 3 mosse, capire come funzionano in 3 mosse. Il nostro scacchomatto in 3 mosse è fatto di una prima mossa, in cui rendiamo i PE raggiungibili uno con l'altro tramite IP. In questa prima mossa ignoriamo la sigla MPLS. La seconda mossa è utilizzare BGP per annunciare i prefissi dei customer. Ma BGP non è fatto per il routing interdominio? Lì in AS100 non c'è traccia di internet, eppure usa BGP. La tecnologia BGP è fatta per un'altra cosa, eppure viene sfruttata in questo frangente; l'uso di BGP nelle VPN MPLS è molto più importante in questo momento del BGP sfruttato nel routing interdominio. Se prendiamo la tabella di instradamento dei grandi provider che utilizzano BPG in questo modo, e ci guardiamo dentro, scopriamo che la tabella è molto più grande di quella di Potaroo. Qui si passa dalle centinaia di migliaia di entry di Potaroo ai milioni di entry di provider come British Telecom. Terza mossa: utilizziamo MPLS per realizzare tunnel all'interno del backbone del provider. Qui scopriamo che MPLS viene utilizzato per realizzare gli stessi tunnel visti nell'AS di transito. La differenza è che cui la configurazione scala, è molto facile. Scopo del gioco è realizzare delle VPN per i customer da parte del provider, e queste sono le tre mosse con cui questo ci riesce.

#### Prima mossa

Ignoriamo tutti gli altri aspetti. In questo momento abbiamo una rete IP, su ciascuno dei PE prendiamo la loopback, le diamo un indirizzo IP e vogliamo anche assicurare la raggiungibilità IP tra i PE nel backbone. Come faccio a realizzare la raggiungibilità tra le loopback? Utilizzo un IGP a scelta, ad esempio OSPF.

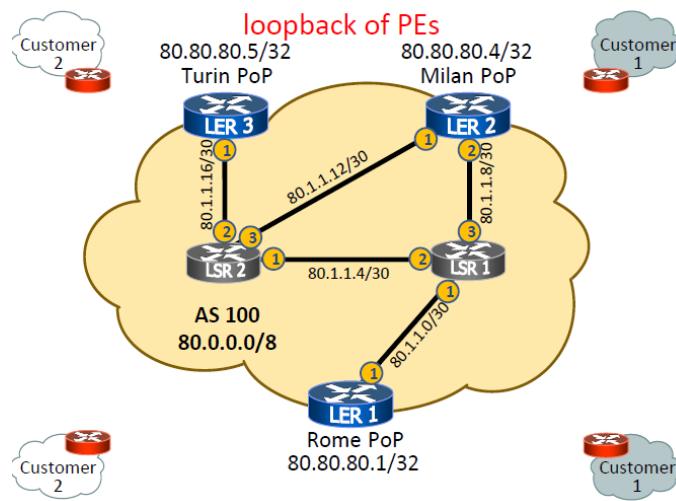


Figura 259 - Loopback of PEs

Quando ho messo in piedi questa cosa mi viene in mente una soluzione allettante, mi viene la tentazione di mettere in piedi, per il customer 1, dei tunnel IP-IP tra i PE su cui il customer è attestato, e poi quando ho i router li uso come se fossero dei fili, e a questo punto ho realizzato l'infrastruttura. L'idea è

ragionevole, ma ha degli svantaggi: è difficile da configurare, e ha un numero di configurazioni quadratico, quindi la soluzione viene scartata. Non soltanto: fai tunnel IPIP, ma quando ci sono vari tunnel attestati sullo stesso PE, come fai a distinguere il customer su cui il PE è attestato? Questo può creare dei problemi. Invece di fare dei tunnel IP-IP si fanno dei tunnel IP in UDP, ma la configurazione diventa ancora più complessa. La soluzione viene scartata. Il risultato della prima mossa quindi è definire un indirizzo di loopback dei PE e fare routing IP all'interno dell'AS.

#### *Seconda mossa*

Dobbiamo raccontare ai vari PE dove sono attestate le varie VPN dei customer. Mi piacerebbe che un PE dicesse "da me c'è il customer numero 4 con questo prefisso e quest'altro prefisso". Utilizziamo BGP per annunciare i prefissi dei customer. L'annuncio BGP, però, che semantica ha? Io dico a te che ho questo prefisso, perciò se hai quel traffico me lo puoi mandare. Si usa in realtà una variante di BGP, che si chiama MP-BGP. Questa non è una vera variante, è ancora BGP, in cui si usano delle communities fatte apposta (Extended Communities) per fare questo giochetto. Mettiamo in piedi un peering iBGP con tutti gli altri PE, usando per scalare dei route reflector. Le network dei customer sono annunciate all'interno dei peerings. Concretamente:

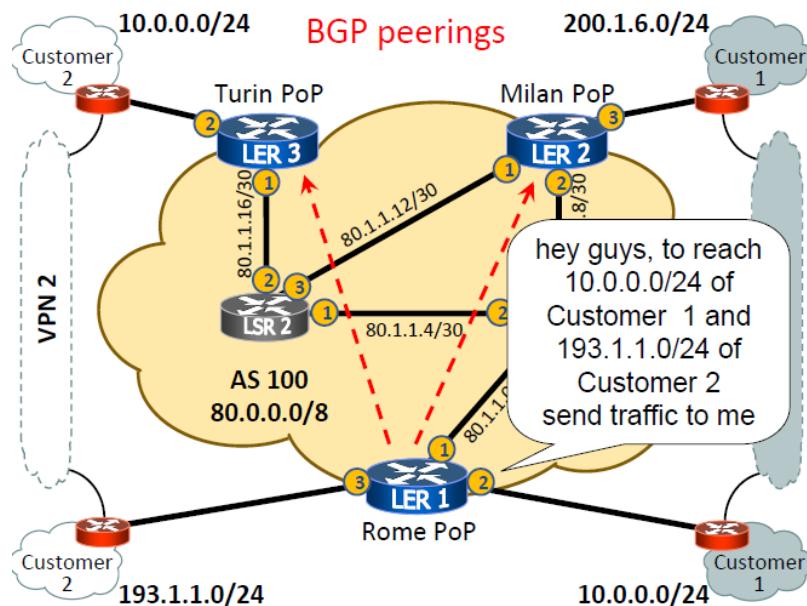


Figura 260 - BGP peerings

Questi sono annunci assolutamente semplici per BGP, quindi BGP questa cosa la fa benissimo. Siccome questi piani di indirizzamento sono sovrapponibili, non basta dire "per la 212 manda traffico a me", ma "per la 212, relativa alla VPN di customer 1, manda traffico a me". All'interno del pacchetto BGP però c'è spazio per metterci dentro il nome di una VPN? Non c'è. Ci si mette allora una community, che è un campo aggiornale all'interno del pacchetto e dentro la community ci mettiamo la VPN, e il gioco è fatto. Questo annuncio perché può muoversi all'interno dell'AS perché c'è connettività dentro l'AS, e la connettività è

garantita dai peering iBGP. Io adesso so che dal router di Torino, per mandare traffico alla 193.192.172.0/24 di customer 2 devo mandare traffico a LER1. Come glielo mando? Con un tunnel IP-IP? Troppo difficile da configurare. Vediamo se c'è un'idea più furba. L'idea più furba è utilizzare MPLS per fare questi tunnel, terza mossa.

#### *Terza mossa*

Un pacchetto IP di un customer, proveniente da un CE, viene incapsulato dal PE vicino alla sorgente in un pacchetto MPLS. Il PE vicino alla sorgente invia il pacchetto alla loopback del PE vicino alla destinazione, e il pacchetto IP attraversa il backbone dentro un pacchetto MPLS. Dentro al pacchetto il PE ci mette due label. La label più in basso (quella che non esce mai, perché rimane dentro al pacchetto per tutto il viaggio) denota la VPN. La semantica è la stessa delle VLAN in rete locale, IEEE 802.1q. Qui per colorare i pacchetti si usano le label MPLS. Nel pacchetto ci si mette un'altra label immediatamente sopra. Su questa label si fa label swapping, e viene fatto sempre sulla label affiorante. La label utilizzata per il label swapping viene cambiata per ogni hop del viaggio dal PE di partenza al PE di arrivo. La label più alta, per convenzione di MPLS, viene buttata al penultimo hop, quando si è praticamente arrivati a destinazione.

#### LDP

Come si costruiscono questi cammini di label swapping? Ci pensa LDP, Label Distribution Protocol. Non è l'unica strada possibile, ma è molto diffusa. LDP è utilizzato per costruire gli LSP (Label Switched Paths), e lo fa importando le informazioni dal data plane IP. Può farlo perché ogni router sa come raggiungere ogni loopback (prima mossa), e se hai un cammino IP puoi prendere quel cammino IP e farlo diventare un cammino MPLS. Facendo così inoltre, se cambia il routing IP automaticamente puoi far cambiare il routing MPLS, semplicemente ricalcando il cammino che avrebbe fatto IP. Cosa vuol dire "label swapping data plane di MPLS"? Ci sarà un data plane per i colori. Quando si studia un algoritmo di routing si vedono sempre due cose: il data plane e il control plane, la tabella di routing e come si costruisce. Il tassello mancante è proprio questo. LDP lo fa, utilizzando le informazioni che già ci sono. Nel data plane IP c'è già scritto come si raggiungono le label, quindi tanto vale costruire dei cammini che ricalcano la raggiungibilità di quelle loopback. Questo lo possiamo fare perché abbiamo fatto bene la prima mossa. Ricapitolando: i router parlano IP, e hanno quindi un data plane IP; hanno contemporaneamente un data plane MPLS. Come si fa a sapere come va instradato un pacchetto? Nei protocolli di livello più basso c'è scritto qual è la pila protocolare che deve gestire il pacchetto. LDP si appoggia sul routing IP, è un protocollo che nasce per essere utilizzato in modo del tutto indipendente da IP, nasce per costruire tabelle di routing in contesti molto diversi da questo, in cui magari non c'è un data plane da parassitare.

#### Riferimenti tecnologici

Per capire bene come funzioni questo meccanismo rivediamo tutte e tre le mosse dello scacco, su un livello tecnologico un po' più sofisticato. Abbiamo visto le tre mosse; tutto ciò deve anche scalare, non devo

incorrere nel vecchio problema della configurazione manuale dei tunnel. Per convincerci che la prima mossa sia stata fatta bene andiamo a fare uno show ip route su LER1:

---

```
LER1>show ip route
Codes: C - connected, O - OSPF
[.]
C 80.80.80.1/32 Loo
O 80.80.80.4/32 via 80.1.1.2, GE1/o
O 80.80.80.5/32 via 80.1.1.2, GE1/o
```

---

E scopriamo che c'è effettivamente la raggiungibilità tra le diverse loopback, realizzata attraverso OSPF. Facciamo adesso uno show IP route su LSR2:

---

```
LSR2>show ip route
Codes: C - connected, O - OSPF
[.]
O 80.80.80.1/32 via 80.1.1.5, GE1/o
O 80.80.80.4/32 via 80.1.1.14, GE3/o
O 80.80.80.5/32 via 80.1.1.18, GE2/o
[.]
```

---

e scopriamo la raggiungibilità delle diverse /32. Naturalmente il data plane IP deve assicurare la raggiungibilità tra le interfacce di loopback dei PE, nel senso che altri prefissi sono inutili per MPLS, però dal punto di vista del networking possono essere interessanti perché magari voglio entrarci con telnet, o via SNMP. Questo lavoro potrebbero farlo anche le rotte statiche, ma in caso di link failure ci sarebbero dei problemi di raggiungibilità. Mossa numero 2: control plane BGP: andiamo a guardare dentro LER1 e facciamo il solito show ip bgp, ma con qualche opzione in più: show ip bgp vpnv4 all. È un normale show ip bgp però fatto apposta per mostrare le VPN di IPv4. Guardiamo quello che esce fuori:

---

```
LER1>show ip bgp vpnv4 all
Network Next Hop ... ... LocPrf
Route Distinguisher: 100:11 (VPN1)
 *>i212.102.68.0 80.80.80.4 100
Route Distinguisher: 100:22 (VPN2)
 *>i193.193.173.0 80.80.80.5 100
```

---

Network, NextHop,...,LocPrf, relativamente ad una certa VPN. Nell'annuncio va identificata anche la VPN, e questo è il motivo per cui all'indirizzo IP viene aggiunta un'altra cosa, che si chiama route distinguisher. Sono nati degli indirizzi IPv4 che però non sono spediti attraverso la rete. Il route distinguisher serve solo a dire, negli annunci che vanno da un PE ad un altro PE, a dire: "questo prefisso è di questa VPN", quindi non va confuso con altri prefissi. Su ogni PE, visto che sono disponibili diverse VPN, ho bisogno di avere

una tabella di instradamento per ciascuna VPN, devo cioè avere tabelle di instradamento virtuali multiple, e ogni customer port di un PE viene associata con una specifica tabella di instradamento virtuale. Questo viene fatto a tempo di provisioning, nel momento in cui l'infrastruttura viene realizzata. Quando si dice "porta" si dice una cosa che può essere una porta fisica, oppure più probabilmente una porta logica su un router, che potrebbe essere per esempio una porta relativa ad una certa VLAN. Questa struttura di instradamento virtuale si chiama VRF, Virtual Routing and Forwarding. Questo consente ad un router di avere diverse tabelle di instradamento, e ogni tabella di instradamento si chiama VRF instance. Ogni istanza di VRF contiene le rotte che sono ricevute dai CE che sono direttamente connessi e che sono associati con quell'istanza di VRF, e contiene inoltre le rotte che sono ricevute via BGP da altri PE. LER1 ha due VRF: una relativa a customer 1 e una relativa a customer 2. Sono tabelle di instradamento IP o MPLS? IP. Abbiamo detto che c'è bisogno di distinguere i diversi prefissi IP, perché possono essere sovrapposti, e devono essere sovrapposti negli annunci BGP. Per farlo si aggiunge il route distinguisher all'indirizzo IP. Un route distinguisher è fatto così:

$$RD = \text{Type} + \text{Provider's Autonomous System Number} + \text{Assigned Number}$$

Questo fa sì che non ci siano VPN che hanno lo stesso route distinguisher, e consente di convertire indirizzi IP non univoci in indirizzi VPN-IP non univoci. Mossa numero 3: il data plane MPLS. Per capirlo bene guardiamo dentro le tabelle di instradamento MPLS dei router.

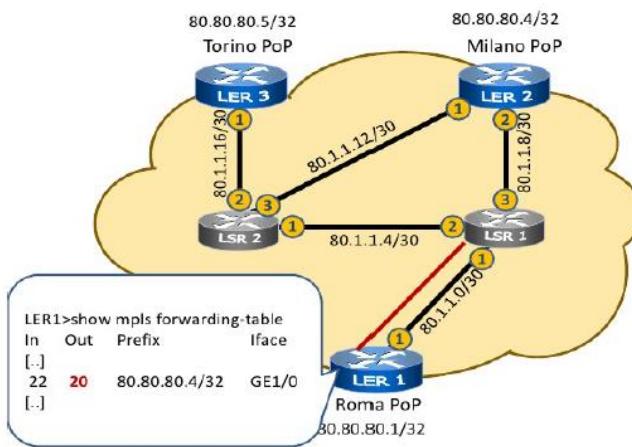
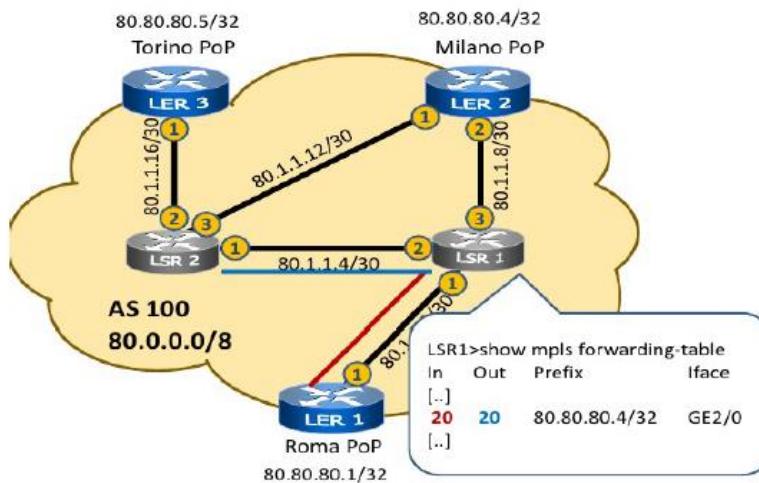
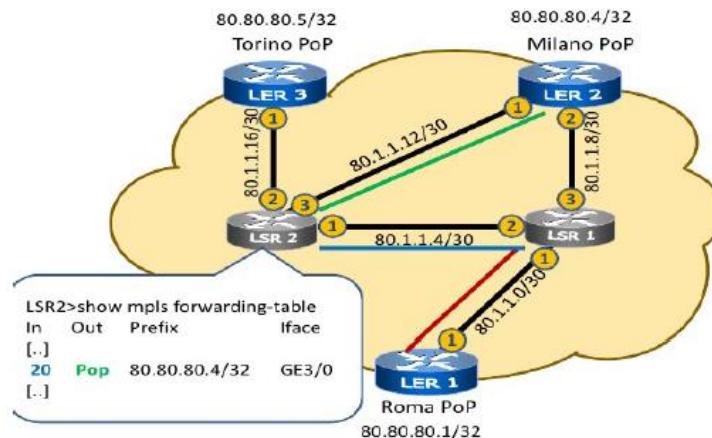


Figura 261 - tabelle di instradamento MPLS dei router

Se andiamo a guardare dentro la tabella di instradamento di LSR1 ci troviamo una cosa più semplice e diretta:



Ci troviamo che se arriva un'etichetta "rosso 20" la dobbiamo sostituire con un'etichetta "azzurro 20". E poi ancora:



se arriva un'etichetta "azzurro 20" fai un POP sullo stack e manda il pacchetto a LER2 (Milano PoP). A questo punto siamo in grado di dare un significato alle sigle LER e LSR. Un LER è un router che per primo incapsula un pacchetto dentro MPLS, un LSR è un pacchetto che fa label switching all'interno di una rete. Mettiamo tutto assieme:

- un host di customer1 a Roma invia un pacchetto destinato a 212.102.68.2 (a Milano);
- LER1 riceve il pacchetto dal CE: aggiunge un'etichetta MPLS per indicare che appartiene a VPN1;
- LER1 cerca VPN1 nella sua tabella di instradamento: capisce che il next-hop è 80.80.80.4 (la loopback di LER2);
- LER1 cerca 80.80.80.4 nella LFIB: trova "label 20, interfaccia GE1/0";
- LER1 aggiunge un'altra etichetta MPLS (20) e invia il pacchetto a GE1/0.

## VPN

Il data plane MPLS deve mettere in piedi degli LSP (Label Switching Path) tra le interfacce di loopback dei PE. Ciascun LSR usa la sua tabella per scambiare la label in cima allo stack e inviare il pacchetto al next-hop corretto. Il penultimo router semplicemente toglie la label in cima allo stack. In questo modo il next-hop vedrà solo la label sottostante, e la userà per distinguere tra le diverse VPN. Il route distinguisher è sufficiente a distinguere gli spazi di indirizzamento delle varie VPN? Il route distinguisher aiuta a sovrapporre gli spazi di indirizzamento: rende unico il prefisso di un customer. Finora abbiamo pensato in questo modo: una VPN è un insieme di prefissi attestati su un PE tramite un CE. Qualche volta le VPN non sono propriamente una mesh, quindi non abbiamo la possibilità di distinguere le VPN attraverso il route distinguisher. Quando si realizza una VPN si pensa che si hanno dei router che comunicano fra loro in full mesh (tutti con tutti), ma nella realtà non è così. La tecnologia va resa un po' più complicata. Il customer può dire: ho diverse VPN, quelli non si devono vedere, ma in qualche specifico caso, questo prefisso, questa macchina, devono essere viste dalla VPN X. Non esiste un customer che dice "la mia VPN è questa, questo è il confine, basta". Nascono topologie complicate come questa:

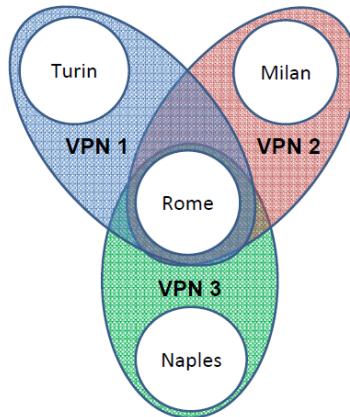


Figura 262 - VPN complessa

Si hanno 3 VPN che si vogliono tenere separate, ma la parte romana deve interagire con tutte le altre (quindi in realtà è una sede condivisa). Per fare questo abbiamo bisogno di introdurre l'idea di route target.

## Route target

Route target è un'extended community, e può essere utilizzato per indicare quali rotte devono essere importate ed esportate in un VRF. VRF: è una tabella di instradamento locale che fa riferimento ad una certa VPN; nella VRF ci si mette roba locale e roba che viene da remoto, presa da annunci BGP. Aggiungiamo a questi annunci BGP una label. A questo punto quando arrivano annunci relativi a certe label posso stabilire in modo critico se annunci relativi a certe label li voglio oppure no. In questo modo

chi importa può fare le sue scelte; queste label si chiamano route target, e possono supportare topologie complesse. Si assegna un route distinguisher per ogni site, un route target per ogni VPN, e configuri i PE per importare ed esportare rotte con route target specifici. Tornando all'esempio di prima, possiamo mettere un route distinguisher per ogni sede.

- Route Distinguishers:
  - Torino RD 100:1
  - Milano RD 100:2
  - Roma RD 100:3
  - Bari RD 100:4
- Route Targets:
  - VPN1 → 100:1000
  - VPN2 → 100:2000
  - VPN3 → 100:3000

e scriviamo sui PE di Roma e Torino:

- PE a Torino
  - ip vrf siteTorino
  - rd 100:1
  - route-target import 100:1000
  - route target export 100:1000
- PE a Roma
  - ip vrf siteRoma
  - rd 100:3
  - route-target import 100:1000
  - route target export 100:1000
  - route-target import 100:2000
  - route target export 100:2000
  - route-target import 100:3000
  - route target export 100:3000

Guardiamo la configurazione:

---

```
hostname LSR1
mpls label protocol ldp
interface Loopback0
ip address 80.80.80.2 255.255.255.255
interface GigabitEthernet1/0
ip address 80.1.1.2 255.255.255.252
```

---

```
mpls ip
interface GigabitEthernet2/0
ip address 80.1.1.5 255.255.255.252
mpls ip
.
.
.
router ospf 10
network 80.0.0.0 0.255.255.255 area 10
```

---

e vediamo sia come è fatto un LSR, sia come è fatto un router di quelli che l'etichettatura la devono effettivamente fare. Configurare un PE è una cosa un po' più complessa. Quattro mosse:

1. configura l'interfaccia di loopback e configura un IGP sulle porte non-customer
2. parla MPLS e LDP sulle porte non-customer;
3. configura i peering iBGP in full mesh;
4. fai un mapping delle customer port sulle istanze di VRF.

## NAT (Network Address Translation)

### Private address allocation

Ci sono molte reti in cui si ha l'esigenza di utilizzare la tecnologia IP di livello 3, ma non si ha l'esigenza di fruire di (o fornire) servizi a internet. Per moltissime reti internet è soltanto un servizio esterno, e il rapporto con internet può non esserci o può essere soltanto occasionale. Queste reti in cui non c'è l'esigenza di internet possono essere configurate come reti IP pur non essendo fisicamente connesse a internet. Gli indirizzi IP assegnati alle macchine hanno la necessità di essere univoci solo all'interno della rete privata (possono coincidere con altri indirizzi assegnati ad altre macchine in internet o in altre reti private). Vantaggi di questa configurazione:

sicurezza (impenetrabilità da attacchi esterni);

ampia disponibilità di indirizzi;

risparmio di indirizzi globalmente unici.

Lo svantaggio ovvio di questa configurazione è l'isolamento totale, non si interagisce con internet. La possibilità di costituire reti private ha portato alla RFC 1918, per l'allocazione degli indirizzi specifici per le reti private, relativa alla private address allocation. Questi indirizzi sono sottratti agli indirizzi "globalmente unici" di internet:

nessun pacchetto in internet può averli a bordo;

sono utilizzati esclusivamente all'interno delle reti private.

Gli indirizzi privati sono i seguenti:

10.0.0.0/8 (da 10.0.0.0 a 10.255.255.255): equivalente ad una rete in classe A;

172.16.0.0/12 (da 172.16.0.0 a 172.31.255.255): equivalente a 16 reti in classe B;

192.168.0.0/16 (da 192.168.0.0 a 192.168.255.255): equivalente a 256 reti in classe C.

Come si mettono assieme quelle due affermazioni che sembrano essere in contrasto? Queste reti sono praticamente ovunque, e sono tipicamente un core business per i provider più significativo di quanto si pensi. Noi pensiamo ad un provider come un'organizzazione che ci fornisce un accesso a internet, oppure come un'organizzazione che fa hosting e housing. I provider in realtà fanno la maggior parte del loro business su un piano completamente diverso, che è quello della realizzazione di reti private. Le reti private si realizzano quando ci sono aziende che devono mettere la propria infrastruttura su una rete separata. Alcuni provider fanno questo quasi per vocazione esclusiva (British Telecom Italia, ex Albacom). Quando si realizza una rete privata quello che si fa è chiedersi: queste macchine dentro l'ente hanno bisogno di un IP pubblico, da mostrare in internet? Non è necessario, queste macchine devono parlare tra di loro. Gli si

danno quindi gli indirizzi privati che non sono visibili all'esterno. Se la rete è grande il 10... ci consente di avere un indirizzamento piuttosto ampio. Talvolta è necessario comunque avere sì un piano di indirizzamento privato, ma è opportuno anche fare in modo che le varie reti private possano dialogare fra loro, e quindi dargli indirizzi consistenti.

## NAT (Network Address Translation)

Un po' di terminologia essenziale:

- *stub domain*: una rete in cui viaggiano pacchetti provenienti o destinati alla rete stessa ma nessun pacchetto in transito;
- *address realm*: una rete in cui gli indirizzi identificano univocamente gli host;
- st all'interno di essa:
  - realm diversi potrebbero avere indirizzi sovrapposti;
  - internet è un realm;
  - ogni rete che utilizza indirizzi privati è un realm a sé stante;
- *public (o global) address*: indirizzo ottenuto dallo IANA (Internet Assigned Number Authority), cioè indirizzo del realm internet;
- *private (o local) address*: indirizzo non ottenuto dallo IANA, il cui uso privato è consentito dall'RFC 1918.

Quando si hanno dei realm in linea di principio questi realm sono completamente separati, ma potremmo aver voglia di farli comunicare. Lo strumento per farlo si chiama NAT, e ha il compito di tradurre indirizzi IP tra due realm. Per farlo il pacchetto in transito viene modificato: il numero IP che non è legittimo nel realm destinazione viene sostituito con un altro numero IP opportuno. Il routing deve essere completamente trasparente per quanto riguarda gli end system: mittente e ricevente del pacchetto del pacchetto IP non sono consapevoli della traduzione intermedia.

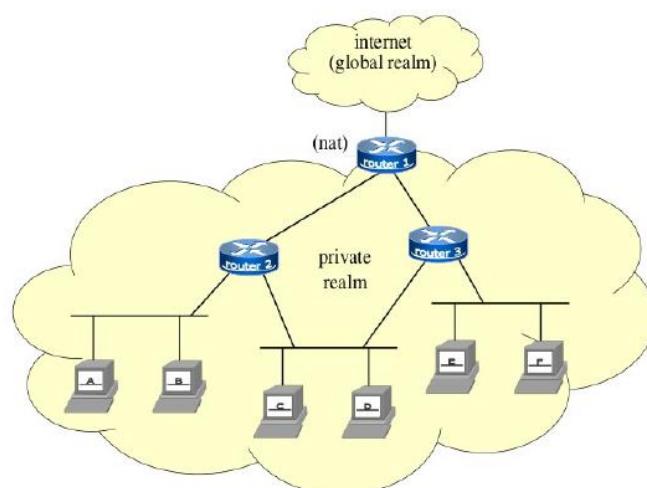
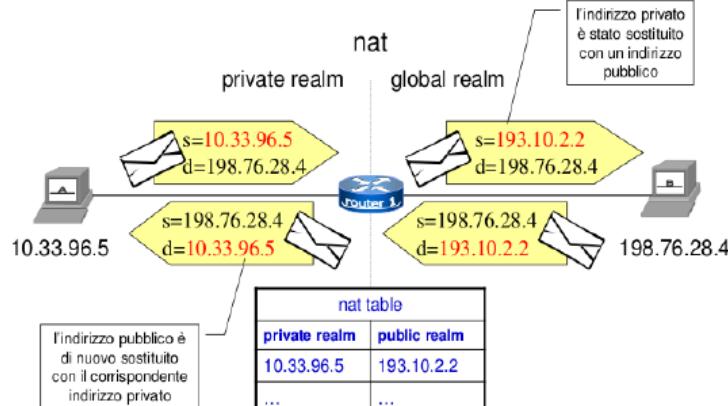


Figura 263 – Esempio

In questo esempio ipotizziamo che il router1 abbia a disposizione gli indirizzi 192.10.2.0/24



Il routing è fatto in modo tale che i pacchetti che vanno verso il realm esterno passino per quel router, che si occupa di sostituire gli indirizzi.

Vantaggi:

- consente ad una rete che fa uso di indirizzi privati di afferire ad internet;
- consente a molte macchine di afferire ad internet con pochi indirizzi globalmente unici:
  - le macchine vengono configurate con indirizzi privati;
  - quando una macchina invia un pacchetto al suo default gateway, questo gli assegna temporaneamente un indirizzo pubblico preso da un pool (pooling of IP addresses);
  - la dimensione del pool determina il numero massimo di macchine che possono afferire ad internet contemporaneamente;
- consente una facile migrazione da un provider ad un altro provider:
  - le macchine non devono essere riconfigurate sostituendo i numeri IP assegnati dal primo provider con quelli assegnati dal secondo provider.

#### Binding degli indirizzi

La traduzione (binding) tra indirizzi può essere statica o dinamica.

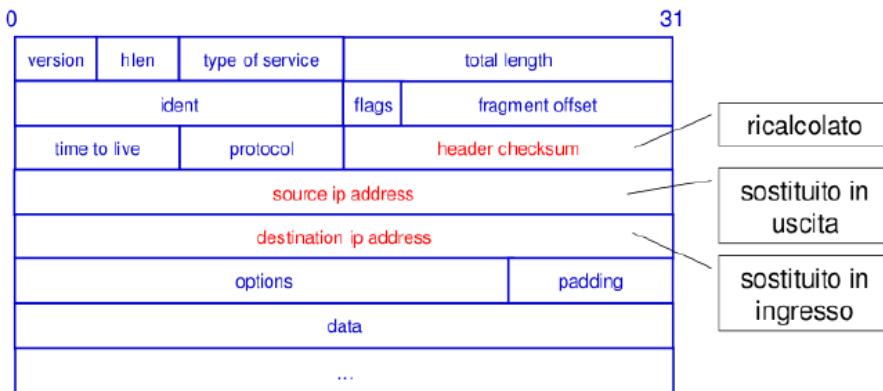
- binding statico: la tabella viene configurata manualmente;
- binding dinamico:
  - la tabella viene calcolata dinamicamente (cambia nel tempo a seconda del traffico);
  - gli indirizzi pubblici vengono assegnati alle macchine che ne hanno bisogno.

La tipica situazione è mista: ad alcune macchine viene assegnato un indirizzo fisso, ad altre un indirizzo calcolato dinamicamente. Quando si deve binding statico e quando dinamico? All'interno di una rete con indirizzamento privato gli apparecchi sono classificati in base a diverse esigenze di connettività:

- nessun accesso: apparecchi che non necessitano di essere connessi in internet;
  - display degli arrivi e delle partenze di stazioni e aeroporti sportelli, registratori di cassa, conta soldi, leggi-badge, risorse locali: stampanti, plotter, interfacce dei router di una rete interna;

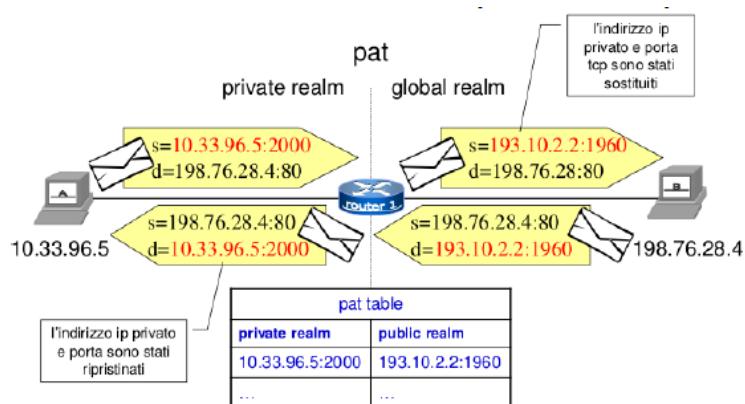
- tanto vale dargli degli indirizzi che iniziano con 10, se non devono essere visti dall'esterno.
- accesso parziale: apparecchi che afferiscono ad internet esclusivamente come client; non ospitano servizi pubblici, non devono avere un indirizzo IP associato staticamente.
- accesso completo: apparecchi che ospitano servizi pubblici; devono avere sempre lo stesso indirizzo IP (quello registrato nel dns)

Con che cosa vengono modificati i pacchetti IP?

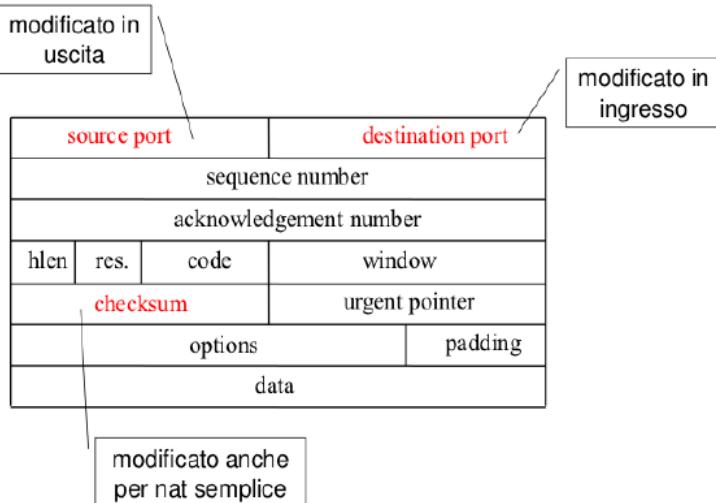


### PAT (Port Address Translation)

È una variante di NAT, chiamato anche IP masquerading oppure NAPT (network address and port translation). Consente a molte macchine con indirizzamento privato di utilizzare lo stesso indirizzo pubblico. Quando gli indirizzi IP pubblici sono più d'uno viene chiamato anche multi-PAT. Permette di scalare in questo modo: nel pacchetto IP in uscita viene inserito il solo indirizzo pubblico disponibile, e per riconoscere le macchine interne vengono usati i numeri delle porte TCP e UDP. I numeri delle porte nei pacchetti di livello 4 vengono sostituiti quando il pacchetto transita attraverso al router. Potremmo avere una tecnica mista, in parte NAT e in parte PAT.



Questo ha un effetto sui pacchetti TCP:



Nat e pat violano il principio end-to-end, secondo cui lo stato della connessione tra due host deve essere conservato e gestito solamente nei due end systems (e non negli intermediate systems). Il fatto che c'è una tabella (per la traduzione degli indirizzi) significa che c'è uno stato in una parte intermedia della rete. Questa circostanza ha effetti sulla connettività end-to-end:

- una macchina in una rete privata non può scambiare messaggi con un'altra macchina in una seconda rete privata;
- le applicazioni peer-to-peer, per esempio, sono ostacolate.

Un esempio tipico della seconda situazione: la rete Skype. Il suo lavoro è fortemente ostacolato dalla presenza di NAT, quindi nasce una serie di stratagemmi chiamate tecniche di NAT traversal per risolvere questi problemi. Un modo per risolverlo può essere avere una terza macchina che tiene traccia degli indirizzi.

#### Problemi di nat e pat

- Protocolli di livello di applicazione: è compromesso il funzionamento dei protocolli di livello di applicazione che menzionano gli indirizzi IP, come per esempio il protocollo FTP. Dentro FTP ci sono tracce evidenti degli IP utilizzati, e questo richiede, se si passa per NAT, di modificare il payload, non soltanto gli header.
- Frammentazione: la sostituzione degli indirizzi deve essere coerente in ogni frammento.
- DNS: per le macchine interne: i nomi interni devono corrispondere agli indirizzi del private realm mentre i nomi esterni devono corrispondere ad indirizzi del public realm; per le macchine esterne: tutti i nomi devono corrispondere ad indirizzi del public realm.
- Prestazioni degli apparati di rete: le prestazioni dei router (pacchetti per secondo instradati) si deteriorano notevolmente quando le funzionalità di NAT o PAT sono attive.

## Il protocollo ipv6

### Ipv6

Mettiamo a confronto gli indirizzi IPv4 e gli indirizzi IPv6. Si passa da indirizzi lunghi 32 bit, suddivisi in gruppi da 8 bit ed espressi in forma decimale (es. `192.168.0.1/27`) a indirizzi di 128 bit, divisi in 8 blocchi da 16 bit ciascuno, ed espressi tipicamente in notazione esadecimale (es. `2001:760:0002:0000:5bff:febc:5943/64`). La scrittura di questi indirizzi può essere semplificata, dicendo che c'è una sottostruttura fatta di tutti 0, e non è il caso di starli a scrivere tutti:

---

`2001:760:2:0:0:5bff:febc:5943/64`  
`2001:760:2::5bff:febc:5943/64`

---

Quali miglioramenti ci sono rispetto ad IPv4? In primo luogo c'è stata una semplificazione dell'header, con l'obiettivo di ridurre il carico computazionale dei router e se possibile contenere l'overhead, e quindi in generale contenere l'occupazione di banda. Come risultato si ha che l'header IPv6 è lungo solo il doppio dell'header IPv4, e questo non è male, visto che gli indirizzi, che sono una gran parte dell'header, passano da 32 a 128 bit. C'è una nuova gestione delle opzioni (che sono una delle questioni più faticose per un router ipv4), per ottenere un forwarding più efficiente e garantire una maggiore estensibilità del protocollo, e anche qui si ottiene un incremento delle prestazioni.

### Header ipv4

4Bytes	Ver	IHL	TOS	Total length						
4Bytes	Identification		Flags	Fragment offset						
4Bytes	TTL	Protocol	Checksum							
4Bytes	32 bits Source Address									
4Bytes	32 bits Destination Address									
	IP Options			Padding						

Figura 264 - Header ipv4

È il pacchetto ipv4 che siamo abituati a vedere. Il campo IHL serve perché gli header sono di lunghezza variabile, e possono essere più lunghi o più corti in presenza o in assenza del campo opzioni, e questo è faticoso per un router, perché un router, quando ha un pacchetto in transito deve verificare se ci sono delle opzioni, e processare l'header del pacchetto con lunghezza variabile. Anche la "seconda striscia" è faticosissima per i router, perché i router hanno bisogno talvolta di frammentare i pacchetti, quindi si passa da una MTU più grande ad una MTU più piccola. Il padding serve a far sì che si arrivi ad un numero intero di long word da 4 byte. I campi in giallo non sono più implementati in IPv6. Questo che vuol dire?

Se non hanno più implementato IHL significa che l'intestazione è di lunghezza fissa. Questo vuol dire che i router per i pacchetti in transito non si devono porre il problema di capire quanto è lungo il pacchetto. Posso ancora implementare delle opzioni, ma mi serve qualche trucco. La striscia gialla della frammentazione è stata tolta perché non si frammenta più. Un'altra cosa: la checksum; questa è la checksum di IP. Quella checksum nei pacchetti IPv4 che ruolo aveva? Il pacchetto IPv4 sta dentro un pacchetto di livello 2 e contiene a bordo un pacchetto di livello 4. Quei pacchetti hanno delle checksum, per verificare che non ci sia un errore nel pacchetto. Anche il pacchetto di livello 4 ce l'ha. Se c'è un errore quindi già quelle due checksum se ne possono accorgere, e allora, la checksum di IPv4 che tipi di errori rileva? Immaginiamo il viaggio del pacchetto all'interno della rete. La checksum di livello 2 rileva errori in uno specifico hop, quindi il pacchetto viene scartato nel passaggio da un router al router successivo. L'errore rilevato dalla checksum di IPv4 che errore è allora? È un errore terribilmente subdolo: il pacchetto arriva su un router, e ci arriva correttamente sul router (sennò si sarebbe arrabbiato il livello 2). Il pacchetto arriva sul livello 3, viene processato dal router, il quale se fa casini col payload di livello 4 allora il livello 4 se ne accorge. Il router manipola il pacchetto quando gli arriva: decrementa il ttl. In questa operazione ci potrebbe essere un errore nel momento in cui l'intestazione del pacchetto IPv4 viene manipolato dal router. Di questo non se ne accorge né il livello 2, né il livello 4. Questa checksum è fatta per questo. Si può buttare tranquillamente, perché questi sono errori che hanno una probabilità bassissima di accadere. Se pure ci dovesse essere un errore in questa zona del pacchetto, il pacchetto rischia di essere recapitato dalla parte sbagliata, e sarà buttato da qualche altro punto di vista, perché per esempio su TCP non c'è nessuna connessione in piedi relativa a quello specifico indirizzo IP. Ecco perché questa checksum è sostanzialmente ridondante nell'attuale gerarchia dei protocolli e può essere eliminata.

#### Header ipv6

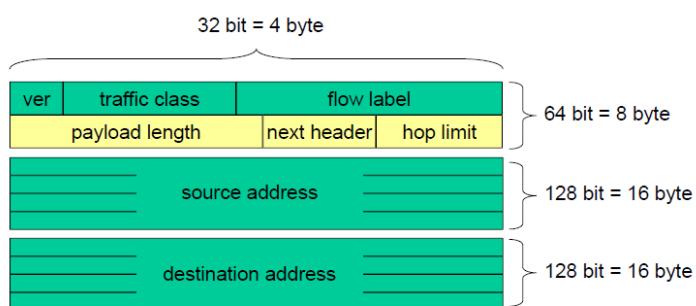


Figura 265 - Header ipv6

C'è poco dentro questo header. Ci troviamo:

- Ver (versione, per versioni successive del protocollo);
- Traffic class: è il Type of service di IPv4;
- Flow label;

- 128 bit di indirizzo mittente;
- 128 bit di indirizzo destinatario.

Ci sono poi i campi che sono quelli di IPv4 ma che sono stati rinominati:

- Payload length: la lunghezza del pacchetto che viaggia a bordo di questo pacchetto (è il Total Length di IPv4);
- Hop Limit: è esattamente il TTL di IPv4 (ha anche lo stesso numero di bit);
- Next header: nel campo Protocol del pacchetto IPv4 c'è scritto qual è il protocollo che viaggia all'interno del pacchetto, qui si chiama Next Header, è la prossima intestazione che devi cercare nel pacchetto.

È un pacchetto più lungo, ma molto più leggero e semplice. Dove trovano posto le opzioni? Non c'è traccia di un posto per le opzioni, perché l'header ha sempre la stessa lunghezza. Inoltre, cosa ci si fa col campo flow label? Non lo sa ancora nessuno; chi ha progettato IPv4 non ha detto bene bene che cosa ci si poteva fare con tutti i campi del pacchetto, quindi la storia si può leggere in questo modo: quello è un canovaccio per una commedia che poi sarà recitata da chi effettivamente utilizzerà il protocollo. Tanto per fare un esempio, quando è nato IP nessuno avrebbe immaginato che poi sarebbe nato NAT. Per ora Flow Label dovrebbe essere usato per mettere lo stesso numero a tutti i pacchetti che sono componenti dello stesso flusso, in modo tale che siano trattati tutti allo stesso modo sui router più attraversati (i router mettono in code diverse i pacchetti che fanno parte di flussi diversi). Una cosa simile la fa già TCP, ma a questo punto il router dovrebbe salire al livello 4, e non gli compete. Qualche volta il router il numero di porta del destinatario non ce l'ha, e non ce l'ha in tutte quelle applicazioni in cui il pacchetto di livello 3 è cifrato. Questo comunque è quello che si dice che si farà di Flow Label. Una cosa interessante è Hop Limit. Il TTL diventa HOP limit, il numero di bit era 8 e rimane 8. Se la rete diventa più grande (perché ci sono tanti apparati), forse 8 bit non bastano più, perché con 8 bit ci si fanno 256 hop, anche meno. C'è una filosofia dietro questo: se è vero che il numero di apparati cresce nel tempo, è anche vero che nel tempo cresce anche il numero di connessioni, e quindi la rete tende paradossalmente ad essere più corta, non più lunga. Con questo spirito i progettisti di IPv6 hanno lasciato come numero di bit 8.

#### Gestione delle opzioni

Le opzioni ci sono ancora, e vengono gestite attraverso gli extension header. La storia normale è: tu hai un pacchetto IPv6, che contiene un pacchetto TCP, che contiene dei dati. Questo è quello che avviene normalmente. Che cosa c'è scritto nel pacchetto IPv4 se dentro c'è un pacchetto TCP? Nel campo protocol c'è scritto che il protocollo che viaggia a bordo di IP è TCP. In IPv6 nel campo next header ci sarà scritto che ciò che c'è dentro il pacchetto è TCP. Devi mettere un'opzione? Mettila tra l'intestazione IPv6 e l'intestazione TCP. Due opzioni? Poco cambia.

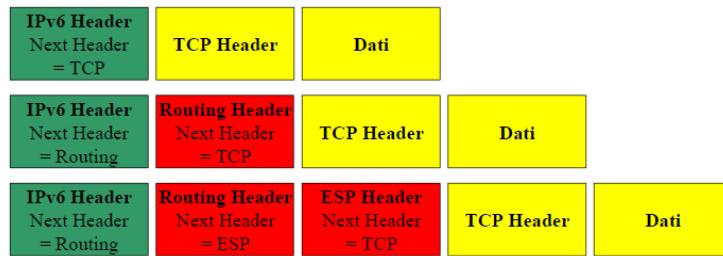


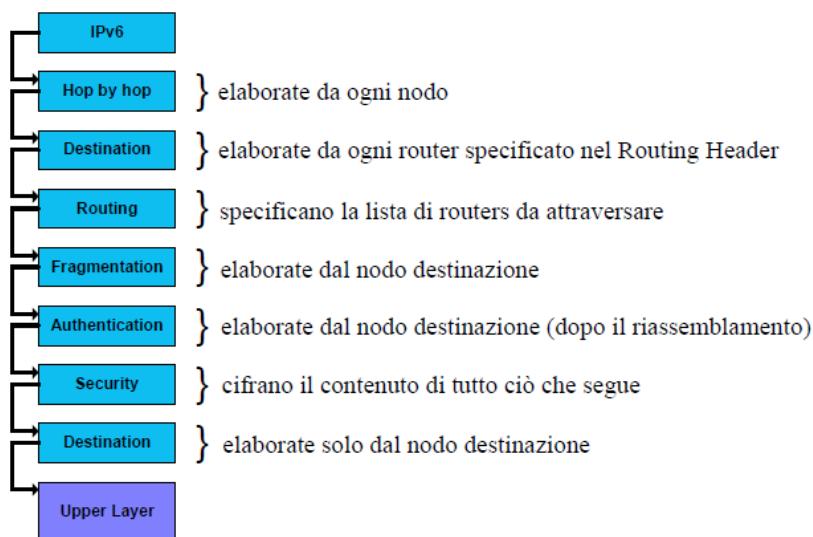
Figura 266 - Extention header

In sostanza, se ti serve un'opzione, questa opzione la fai diventare un'intestazione intermedia. Con questo meccanismo puoi avere una montagna di opzioni; questo consente di avere un pacchetto che di base ha sempre le stesse caratteristiche, se ci sono poi delle opzioni intermedie queste si gestiscono col Next Header. Come sono incastriati tra loro i tipi di header? Ce ne sono parecchi:

- 00: Hop-by-Hop Options. Sono quelle che devono essere esaminate da ogni nodo lungo il percorso, e interessano effettivamente i nodi di transito, le altre interessano solamente i nodi destinatari. Alcune opzioni possibili:
  - Router Alert: serve ad allertare i router intermedi;
  - Jumbo Payload: serve ad avvertire il router attraversato che il pacchetto è di dimensione superiore a quello tipico per un pacchetto IPv6;
- 43: Routing. Si può fare in IPv6, come in IPv4, source routing. L'opzione source routing specifica la lista dei router attraversati. In IPv4 si può fare o source routing o loose source routing; nel loose source routing nel pacchetto c'è una lista dei router attraversati, ma ce ne potrebbero essere altri. In IPv6 rimane solo l'opzione loose source routing, che viene realizzata con questo header. Dentro l'intestazione ci metti la lista esplicita dei router che devono essere attraversati.
- 44: Fragment. Dicevamo che non si fa frammentazione, ma questa cosa in realtà è parzialmente vera, perché si può ancora frammentare, ma si frammenta end to end: chi frammenta è solamente il mittente. Nessun'altra frammentazione è possibile. Questa cosa può sembrare stravagante, perché se IP deve frammentare vuol dire che c'è un mismatch tra quello che fa il livello più alto e quello che fa IP sull'MTU. In ogni caso, per non frammentare mai IPv6 prevede che tutte le macchine debbano supportare una MTU minima di 1280 byte (contro i 68 byte di IPv4). Questo vuol dire che tu macchina mittente se fai tutti pacchetti da al più 1280 byte questi arrivano sicuramente a destinazione. In ogni caso lo standard prevede che tutte le macchine debbano supportare la procedura di path MTU discovery, cioè di scoperta delle MTU che vengono attraversate dai pacchetti
- 51: Authentication Header. Fornisce supporto nativo all'autenticazione, sei sicuro del fatto che il pacchetto ha viaggiato inalterato in tutta la rete e che l'indirizzo IP del mittente non è stato manomesso.

- 60: Destination Options. Usato per trasportare informazioni opzionali che saranno valutate soltanto dal destinatario, e nella Daisy Chain (cioè nella catena delle opzioni) si può trovare in due punti:
  - nel routing header: implica che tutti i nodi intermedi specificati nel Routing Header devono guardare queste opzioni;
  - alla fine della Daisy Chain: questo vuol dire chiaramente che solo la macchina destinataria guarderà queste opzioni.
- 50: Encapsulating Security Payload. È l'opzione di confidenzialità: ciò che viaggia all'interno del pacchetto viene opportunamente criptato in maniera tale che sia intellegibile soltanto al mittente e al destinatario.
- xx: Protocolli di livello piu' alto come per IPv4.
- 58: Internet Control Message Protocol (ICMPv6).
- 59: nessun next header

Le due opzioni, Authentication Header ed Encapsulating Security Payload possono essere usate assieme o separatamente.



*Figura 267 - extension headers: ordine nel pacchetto*

Questa è la Daisy Chain delle opzioni. È la sequenza delle intestazioni che possono essere utilizzate dopo le intestazioni IPv6 e prima delle intestazioni dell'upper layer (cioè di TCP), in questa sequenza.

## Indirizzi ipv6

Si passa da 32 a 128 bit, e questo non vuol dire che si hanno 4 volte il numero di indirizzi di prima, ma 4 volte il numero di bit, quindi si ha a disposizione una montagna di indirizzi (circa  $3,4 \times 10^{38}$ ); in realtà, se si avesse la stessa efficienza di assegnazione di IPv4 questo numero scenderebbe a 1030. In IPv6 questi 128 bit sono fatti in questo modo: 64 bit per l'host e 64 bit per la rete. Questo vuol dire che se hai una punto-punto prendi 64 bit per indirizzare le interfacce che stanno su quella punto punto. Questo è un utilizzo decisamente non ottimale. In questo momento i registri regionali, per invogliare i provider ad utilizzare IPv6 distribuiscono dei grossi banchi di indirizzi (fai un po' di fatica, però hai uno spazio di indirizzamento che andrà bene praticamente per sempre). Questo:

---

X:X:X:X:X:X:X:X

---

è il formato con il quale gli indirizzi vengono tipicamente rappresentati, dove ciascuna X è una sequenza di 16 bit espressa in notazione esadecimale. Per esempio, un indirizzo IPv6 è:

---

2001:0000:1234:0000:0000:0000:ABCD:0532

---

Ci sono alcuni trucchi stenografici: gli zeri a sinistra di un campo sono opzionali, e questo significa che l'indirizzo di prima può essere scritto come 2001:0:1234:0:0:Do:ABCD:532; ancora: i campi successivi di zero sono rappresentati da ::, cioè:

---

2001:0000:1234:0000:0000:0000:ABCD:0532 → 2001:0:1234::Do:ABCD:532

---

Questo però può essere fatto solo una volta in un indirizzo (la notazione 2001::1234::C1Co:ABCD:876 ) non è valida, perché non si saprebbe dove mettere più zeri da una parte o dall'altra. In una URL gli indirizzi si scrivono tra parentesi quadre: [http://\[2001:1:4F3A::206:AE14\]:8888/index.html](http://[2001:1:4F3A::206:AE14]:8888/index.html)

Naturalmente i programmi che usano URL vanno modificati, perché c'è bisogno di un parser diverso, e queste URL sono scomode da usare.

### Tipi di indirizzi

IPv6 suddivide gli indirizzi in:

- unicast (indirizzi dei nodi)
- multicast (indirizzi di gruppi di nodi)
- anycast (indirizzi di servizio)

Non c'è più l'indirizzo broadcast, per paura di attacchi DoS (Denial of Service), ma anche per un altro motivo: si cerca di utilizzare nelle reti un indirizzamento che coinvolga un numero di macchine abbastanza limitato, cercando di non scomodare chi è effettivamente interessato alla comunicazione. Questo fa sì che si usi il più possibile multicast anziché broadcast. Gli indirizzi anycast non sono una novità, sono normali indirizzi unicast, e vale il solito gioco anycast visto altre volte: tu annunci un indirizzo da più parti, e l'annuncio che arriva prima, con la metrica migliore, viene accettato. Questo è una presa di coscienza del fatto che alcuni indirizzi possono essere usati per arrivare a certi servizi in modalità anycast piuttosto che unicast.

primi 16 bit	utilizzo	prefisso	fraz. sp. ind.
0000-01FF	Reserved	0000::/7	1/128
0200-03FF	Reserved for NSAP allocation	0200::/7	1/128
0400-05FF	Reserved for IPX allocation	0400::/7	1/128
0600-1FFF	Reserved	Various	
2000-3FFF	Global Unicast	2000::/3	1/8
4000-FBFF	Reserved	Various	
FC00-FDFF	Unique Local Unicast	FC00::/7	1/8
FE00-FE7F	Reserved	FE00::/9	1/512
FE80-FEBF	Link Local Unicast	FE80::/10	1/1024
FEC0-FEFF	Site Local Unicast	FEC0::/10	1/1024
FF00-FFFF	Multicast	FF00::/8	1/256

(in grigio i tipi deprecati)

Figura 268 -Architettura degli indirizzi ipv6

### Indirizzi Unicast

In IPv4 c'era un tipo di indirizzo unicast, qui ce ne sono un bel po', ma non è tutto: una certa interfaccia non si attribuisce un solo indirizzo unicast, ma se ne attribuisce tanti, che possono essere anche di diversi tipi, in base a questa classificazione.

#### Unspecified

È l'indirizzo formato da tutti zeri:

---

o:o:o:o:o:o:o:o (::)

---

Questo indirizzo indica l'assenza di indirizzo. Che vuol dire? Questo è l'indirizzo che puoi utilizzare quando sei una macchina che non ha ancora un indirizzo (per esempio lo puoi utilizzare nella richiesta DHCP), e lo utilizzerai in una procedura che si chiama duplicate address detection, per verificare che l'indirizzo che stai cercando di usare non sia già usato da qualcuno. Questo fa pensare che IPv6 abbia una procedura standard che, quando vuoi usare un indirizzo, verifica se quell'indirizzo non è già usato. Questa cosa si faceva già in IPv4, e si faceva mandando un ARP sull'indirizzo che si cercava di utilizzare, e se qualcuno rispondeva allora l'indirizzo era già in uso. Quando guardiamo cosa c'è in IPv6 troviamo delle cose che in

IPv4 sono nate un po' come dei trucchetti, e che qui sono state implementate sistematicamente. Anche qui esiste la o/o, la rottazdi default:

---

::/0

---

che è comunque diversa dall'assenza di indirizzo.

#### *Loopback*

La loopback, invece che 127.0.0.1, in IPv6 è ::1 (0:0:0:0:0:0:0:1). Su questo c'è poco da dire. Una cosa che si può fare, per verificare se lo stack IPv6 è attivo sulla macchina è fare:

---

ping6 ::1

---

#### *IPv4 compatibile*

L'idea è quella di inserire indirizzi IPv4 in indirizzi IPv6. I primi 96 bit dell'indirizzo sono posti a 0, e i restanti 32 vengono dall'indirizzo IPv4:

- 0:0:0:0:0:0:192.168.0.1
- ::192.168.0.1
- ::CoA8:1E01

Questi indirizzi sono utilizzati in alcune transizioni IPv4 - IPv6, ora sono deprecati. Qual è l'idea? Tu sei una macchina che ha un indirizzo IPv4 che vuole entrare nel gioco IPv6, non hai voglia di andare a prenderti un indirizzo, vuoi farlo in maniera molto rapida, quello che puoi fare è costruire a partire dal tuo indirizzo IPv4 un indirizzo IPv6 per iniziare a comunicare sulla rete.

#### *IPv6 mapped*

L'idea è analoga a quella degli indirizzi IPv4 compatibile, e qui la sintassi è leggermente diversa: i primi 80 bit sono posti a 0, i successivi 16 sono posti a 1 (FFFF) e gli ultimi 32 bit specificano l'indirizzo IPv4. È la stessa cosa di prima, ma con la struttura dell'indirizzo leggermente diversa, e non sono deprecati.

- 0:0:0:0:0:FFFF:192.168.0.1
- ::FFFF:192.168.0.1
- ::FFFF:CoA8:1E01

#### *Subnet Prefix e Host Identifier*

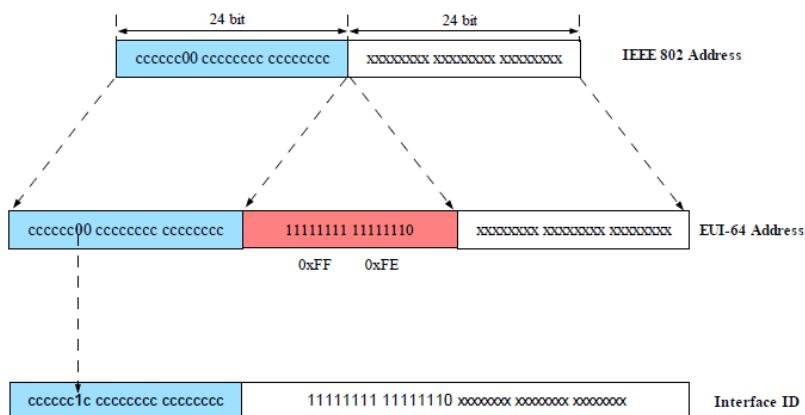
Entriamo un pochino meglio nella struttura degli indirizzi IPv6. Tipicamente un indirizzo IPv6 unicast si compone di due parti:

- il prefisso (primi 64 bit);
- host identifier (restanti 64)



Figura 269 - subnet prefix e host identifier

Questo vuol dire che tu su una rete (intanto continua a valere la regola che ogni LAN ha un suo prefisso) hai delle LAN a cui sono attribuiti 64 bit di indirizzamento (è senz'altro uno spreco, ma questa è la convenzione attuale). Un host può essere identificato manualmente (puoi attribuire un indirizzo IP manualmente) oppure puoi utilizzare l'identificativo di interfaccia, utilizzando il MAC address. Si può costruire un indirizzo IPv6 prendendo il MAC address dell'interfaccia e convertendolo nella sintassi EUI64. Com'è fatto questo interface id? Identifica univocamente un'interfaccia, deve essere univoco su un link e può essere ricavato a partire dall'identificatore EUI64. EUI64 è un'evoluzione del MAC, e identifica un produttore e il numero di serie dell'apparecchiatura; c'è una procedura standard che consente di passare dal formato del MAC address, EUI48, ad EUI64. Per ricavare l'indirizzo EUI64 si prende il bit universal local (il settimo) e lo si inverte. A questo punto si taglia a metà l'indirizzo MAC e si inserisce in mezzo la sequenza oxFFFE.



mac address:	00-AA-00-3F-2A-1C
EUI-64 address:	00-AA-00- <b>FF-FE</b> -3F-2A-1C
complementando U/L:	02-AA-00- <b>FF-FE</b> -3F-2A-1C
in notazione IPV6:	<b>2AA:<b>FF:FE</b>3F:2A1C</b>

Figura 270 - interface id da mac-address

Quel bit dice semplicemente se l'indirizzo ha una valenza locale oppure globale. Non è che sia una cosa bellissima, per certi versi: stiamo utilizzando a livello 3 un'informazione specifica del livello 2, stiamo legando tra loro due livelli della pila ISO-OSI che non dovrebbero essere legati. Questo effettivamente è un meccanismo molto comodo, perché il MAC address è uno standard universalmente adottato, e con uno sforzo di configurazione nullo si riesce a dare un identificatore a tutte le macchine della rete. In realtà, anche qui non si sono inventati niente, perché il vecchio protocollo IPX (utilizzato nella LAN per anni)

aveva esattamente lo stesso trucco nella creazione dell'indirizzo IP. Questo può creare problemi significativi di privacy, perché a questo punto se vai in giro hai un identificatore della macchina che si porta dentro il tuo MAC address. Il MAC address è un'informazione più riservata dell'indirizzo IP, quindi questo atteggiamento può portare a problemi di privacy, perché si può essere tracciati ovunque si è nel mondo: ti prendi diversi prefissi, ma la parte host identifier dell'indirizzo è sempre quella. C'è un RFC che specifica un modo alternativo di generare l'interface ID con una randomizzazione. ARP e RARP ci sono ancora: questo è solo un modo di lavorare, ma non si è obbligati ad utilizzare questo meccanismo. In realtà in IPv6 ARP sparisce, ma viene costruito un oggetto che è molto simile.

#### *Link e site*

Un link per IPv6 è una rete fisica, una LAN, e può essere anche un collegamento punto-punto; nodi dello stesso link sono detti neighbor. Un site è un gruppo di link gestiti da un'unica autorità, per esempio il campus di una università. Il concetto di site è un po' nebuloso.

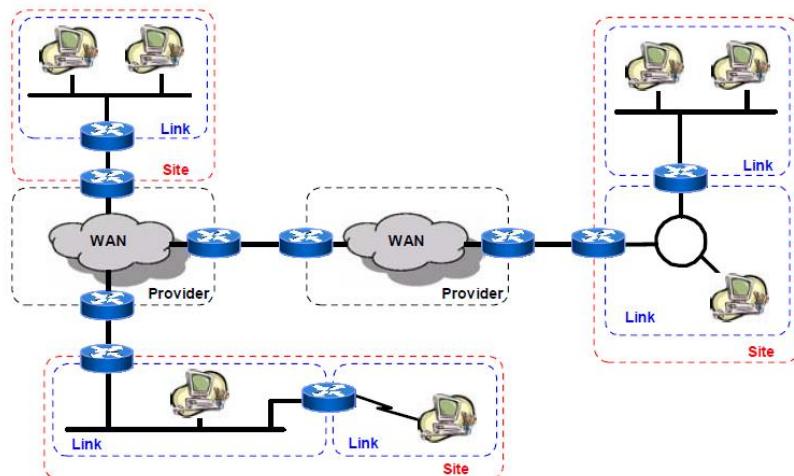


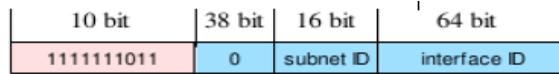
Figura 271 - link e site

In IPv6 esistono degli indirizzi link local; questi indirizzi sono scoped e il loro scope è il link locale. Questi indirizzi possono essere usati per la comunicazione soltanto tra nodi dello stesso link, e non possono passare attraverso un router. Questo è un meccanismo molto comodo perché fornisce ad un nodo un indirizzo IPv6 per iniziare la comunicazione anche se quel nodo l'indirizzo non ce l'ha (quindi tipicamente questo indirizzo nasce all'accensione della macchina). L'indirizzo link local può essere automaticamente configurato sull'interfaccia utilizzando l'interface id che costruiamo a partire dal MAC address. Il formato è questo:



Figura 272 - Link local

Gli indirizzi site local (già deprecati) sono anch'essi indirizzi scoped e sono utilizzati all'interno di un site. Hanno una struttura che è molto simile a quella degli indirizzi link local, con un pattern standard iniziale, però qui ci sono anche 16 bit di subnet id.



*Figura 273 - Site local*

Il problema è che il concetto di site è un po' vago: qual è il router che delimita il site? Non era troppo chiaro, quindi ci è stata messa una pietra sopra, e sono stati introdotti gli indirizzi unique local, che sono quelli che effettivamente assomigliano agli indirizzi privati IPv4. Sono simili agli indirizzi site local, ma sono quasi globalmente univoci. Sono fatti in questo modo.



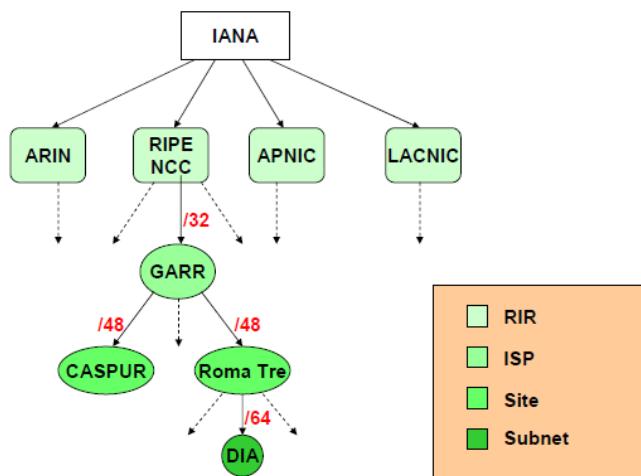
*Figura 274 - Unique local*

Il global id è un numero generato casualmente. Questi sono indirizzi che sono fatti per essere utilizzati come gli indirizzi privati IPv4. Con gli indirizzi privati IPv4 però succedeva ogni tanto questo: tu sei un'azienda che ha una sua rete e uno suo piano di indirizzamento in classe 10. Un'altra azienda ha un suo piano di indirizzamento, anch'essa in classe 10. Ad un certo punto queste due aziende si fondono, e hanno bisogno di un piano di indirizzamento in classe 10 unico. Naturalmente i due piani di indirizzamento si sovrappongono. Questi indirizzi unique local nascono con l'idea di alleviare questo problema, in questo modo: le due aziende si fanno il proprio piano di indirizzamento utilizzando indirizzi unique local. Con 40 bit randomizzati la probabilità che i due piani di indirizzamento si sovrappongano è molto bassa. I 16 bit successivi li utilizzano per le subnet, e gli ultimi 64 per l'host. Questo fa sì che tu abbia una buona possibilità di fare fusioni tra aziende conservando i loro piani di indirizzamento di partenza. L'idea è quella di site, però specificata un po' meglio. Questi hanno una marea di indirizzi, 128 bit per l'indirizzamento, che ci fanno con una classe 10? Una classe 10 deve essere NATtata, per poter uscire su internet, e tipicamente è il trucco che fai quando non hai indirizzi a sufficienza. Tu usi un indirizzamento privato quando non vuoi che lo spazio di indirizzamento interno sia visibile all'esterno, per esempio per motivi di sicurezza, quindi l'idea che ci siano comunque in internet gli indirizzi privati, anche in un mondo in cui lo spazio di indirizzamento è molto ampio è un'idea sensata, e questo è il motivo per cui questi indirizzi sono significativi. Questi indirizzi non sono globalmente instradabili, perché l'unicità non è garantita, e non possono essere né annunciati né aggregati, per via del global ID.

#### *Global unicast*

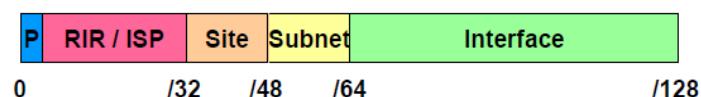
Sono analoghi agli indirizzi IPv4 pubblici, iniziano con i bit 001 secondo il format prefix (la struttura dell'indirizzo) 2000-3FFF, e nascono con una forte gerarchizzazione, con un forte meccanismo di

assegnazione gerarchica, ma con l'idea che, a differenza di IPv4, la gerarchia si rifletta anche nei diversi campi dell'indirizzo. Questo vuol dire che in questo momento se prendi un indirizzo a caso nella rete non riesci a dire se è stato attribuito da un registro europeo o asiatico. Qui si potrebbe avere una strutturazione dell'indirizzo in modo che tutto questo sia chiaro.



*Figura 275 - global unicast: esempio di allocazione*

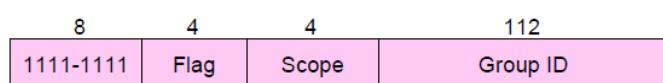
Questo è un esempio di allocazione di questi indirizzi unici, è la gerarchizzazione attuale. Quello che uno pensa di avere in questi casi è un effetto benefico sulle tabelle di instradamento, perché si pensa: se questa rete è di Roma Tre, e siccome questo è CASPUR, che stanno tutti e due dentro GARR, che sta in Italia, magari quella che esce fuori è un'aggregazione di un sacco di roba che sta in Italia e che può essere messa sotto un prefisso unico. Ormai però la struttura dei provider è talmente transcontinentale che il fatto che questo beneficio in futuro ci sia effettivamente non è completamente chiaro.



*Figura 276 - global unicast: politiche di assegnazione*

### Indirizzi multicast

Multicast vuol dire “uno a tanti”, in IPv6 non esiste broadcast, e l’idea di scoped address sostituisce il TTL di IPv4. In IPv4 si fa uso di multicast con dei check sul TTL, per accertarsi che il pacchetto non vada troppo in là nella rete. Qui c’è uno scoped address, che tu per definizione usi all’interno di un ambito prefissato. Il formato multicast è questo:



*Figura 277 - Multicast*

Sono identificati dal format prefix oxFF. Di indirizzi multicast ce ne sono di due tipi: permanenti e temporanei. Quelli permanenti sono del tipo “tutti i router della rete”, “tutti i bridge della rete” e così via. Quelli temporanei sono gli indirizzi che prendi per una trasmissione multicast e che ti serve soltanto per tre ore in un giorno. Lo scope può essere:

- node (1);
- link (2);
- site (5);
- organization (8);
- global (E)

e il group ID identifica il gruppo multicast in un dato scope. Ad esempio se prendiamo il group ID All-Nodes avremo che:

- All’indirizzo FF02::1 partecipano tutte le interfacce sullo stesso link;
- All’indirizzo FF05::1 partecipano tutte le interfacce sullo stesso site;
- All’indirizzo FF0E::1 partecipano tutte le interfacce su internet;

Alcuni indirizzi multicast interessanti:

INDIRIZZO	SCOPE	TIPO
FF02::1	Link	All nodes
FF02::2	Link	All Routers
FF05::2	Site	All Routers
FF02::1:FFxx:xxxx	Link	Solicited-node

Figura 278 - Indirizzi multicast

È particolarmente interessante l’indirizzo multicast solicited node.

### Indirizzi anycast

Non sono specifici di un format prefix, sono semplicemente indirizzi unicast che sono assegnati ad un insieme di interfacce, e servono ad identificare il server più vicino ad un mittente. Alcuni indirizzi anycast nascono già come utilizzati per scopi specifici.

#### Indirizzi per ogni host

Di tutta questa selva di indirizzi, quanti ne arrivano a ciascun host? Ogni host IPv6 deve riconoscere come propri i seguenti indirizzi:

- Un indirizzo link-local per ogni interfaccia;
- Gli indirizzi unicast/anycast assegnati (manualmente o automaticamente);
- L’indirizzo di Loopback;
- L’indirizzo del gruppo All-Nodes multicast;
- Gli indirizzi Solicited-node multicast per ogni indirizzo unicast/anycast assegnato;

- Gli indirizzi multicast di tutti gli altri gruppi di cui l'host fa parte.

## Icmpv6

Protocollo e tipi di pacchetto

È la versione di ICMP per IPv6, ha le stesse funzionalità di base (segnalazione errori, controllo, diagnostica) e aggiunge nuove funzionalità:

- neighbor discovery: Neighbor Solicitation, Unreachability, Autoconfigurazione;
- gestione dei gruppi multicast.

ICMPv6 nasce per accorpare in un unico protocollo le funzioni svolte in IPv4 da ICMP, ARP e IGMP (un protocollo per la gestione dei gruppi multicast in IPv4). Come formato del pacchetto ci dobbiamo aspettare un normale pacchetto IPv6 che contiene imbustato un pacchetto ICMP, con un next header che è il 58.

Nell'header ICMPv6:

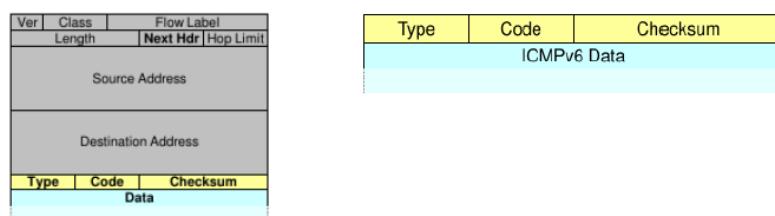


Figura 279 - Formato dei pacchetti

ci sono:

- type;
- code;
- checksum;
- ICMPv6 data.

Il primo bit del campo type distingue tra due classi di messaggi: i messaggi d'errore e i messaggi informativi. I messaggi d'errore vengono tutti da IPv4:

- Destination Unreachable (1);
- Packet Too Big (2);
- Time Exceeded (3);
- Parameter Problem (4).

Packet Too Big è particolarmente interessante in IPv6. I messaggi informativi sono:

- Diagnostica:
  - Echo request/Echo reply (128/129);
- Controllo:
  - Gestione dei gruppi multicast;

- Multicast Listener Query/Report/Done (132/133/134);
- Neighbor discovery;
  - Router Solicitation/Advertisement (133/134);
  - Neighbor Solicitation/Advertisement (135/136);
  - Redirect (137);
  - Inverse Neighbor Discovery (141/142);
- Richiesta di informazioni:
  - Node Information Query/Response (139/140)

#### Path MTU Discovery

In IPv6 non c'è frammentazione, e se proprio si vuole frammentare si può farlo solo end to end (i router non frammentano pacchetti, se ne deve occupare l'host mittente). Per non frammentare l'host deve sapere l'MTU dell'intero collegamento, e per conoscere questa MTU usa una procedura che si chiama Path MTU Discovery, basata su messaggi ICMPv6 Packet Too Big. Un messaggio di questo genere viene generato da un router quando ha un pacchetto troppo grande per immetterlo sulla linea successiva. Questi pacchetti di risposta riportano nel campo dati l'MTU che deve essere utilizzata per passar al link successivo. La procedura funziona in questo modo:

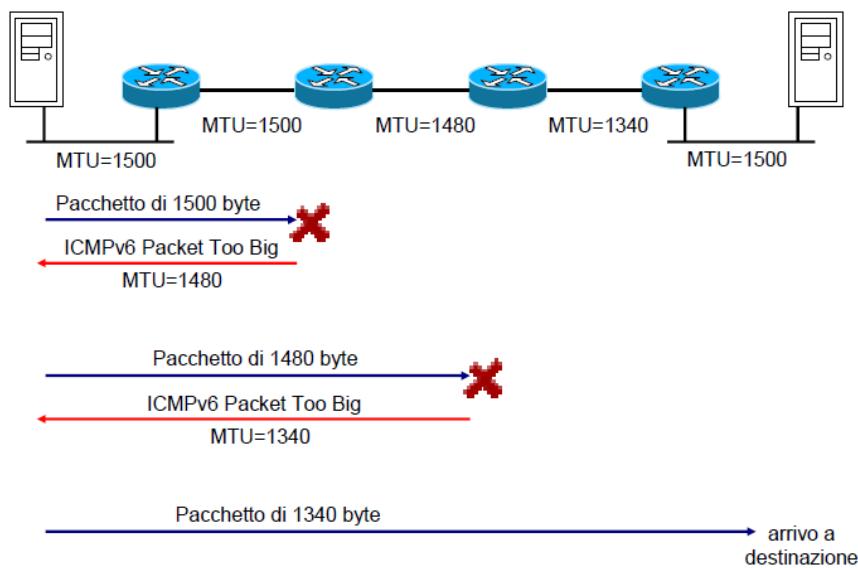


Figura 280 - path mtu discovery: esempio

Un nodo che vuole fare discovery manda un primo pacchetto con le dimensioni standard. Se riceve un messaggio d'errore packet too big manda un nuovo pacchetto con le dimensioni specificate le messaggio di risposta e riprova. Questo fa sì che ad un certo punto passi. Periodicamente un nodo rifà l'operazione, perché se non rinnovi periodicamente la stima impari solo peggioramenti, e non miglioramenti. Non è detto che la si debba fare, perché IPv6 specifica una MTU minima di 1280 byte. Ragionando un po' su questa cosa si capisce che la situazione non è proprio così banale. Questi pacchetti che vengono inviati,

sono pacchetti che nascono da un flusso applicativo o da un flusso TCP? Se nascono da un flusso TCP vuol dire che TCP si è messo già a lavorare per fare frammenti e spedire pacchetti. Quando hai fatto la discovery devi segnalare al tuo TCP che intendi usare una modalità di frammentazione diversa da quella da cui eri partito. Nei primi pacchetti che spedisce TCP tenta di calcolarsi il più rapidamente la CWND. Eventuali failure nella procedura di discovery devono essere considerate nella crescita di CWND oppure devono essere considerate un po' extra, e la CWND la fai partire un po' dopo? Tutte queste cose non sono molto chiare in IPv6. Una cosa è chiara: questo trucco lo puoi utilizzare per scoprire tunnel. In un mondo che è ancora fortemente IPv4, una parte dell'infrastruttura IPv6 già dispiegata è un'infrastruttura a tunnel fatta in questo modo: isola IPv6, altra isola IPv6, in mezzo c'è un'infrastruttura IPv4. Per passare da un'isola ad un'altra faccio un tunnel IPv4 nell'ambito del quale faccio passare dei pacchetti IPv6. Quando spedisce un pacchetto IPv6, per capire che quel pacchetto passa attraverso dei router, puoi fare una procedura di path MTU discovery. Dove sta al mondo una tecnologia con una MTU da 1480 byte? La devono ancora inventare. Se ti arriva una MTU da 1480 vuol dire che hai un passaggio per un tunnel in cui sconti sulla MTU i 20 byte di IPv4 che servono ad encapsulare il pacchetto del tunnel, quindi tutta questa storia della path MTU discovery viene utilizzata adesso come un trucco per scoprire l'eventuale presenza di tunnel, in modo indiretto, nell'attraversamento della rete.

### Neighbor discovery

Già l'espressione non è chiarissima. Nelle procedure di neighbor discovery scopriamo un meccanismo di:

- Address resolution: Neighbor Solicitation, Neighbor Advertisement, Neighbor Unreachability Detection;
- Autoconfigurazione: Router Solicitation e Router Advertisement;
- Redirect.

In generale questo si dice che serve a gestire le informazioni di controllo all'interno di un link. Tutto ciò usa pacchetti di ICMPv6. I messaggi non possono uscire dal link, e sono validi se hanno unHOP limit pari a 255 (questo consente di verificare che il messaggio sia partito dal link).

### Redirection

È simile alla redirezione in IPv4: tu spedisce un pacchetto ad un router, ma quello sa che tu puoi raggiungere quella destinazione più comodamente attraverso un altro router oppure addirittura puoi fare una consegna diretta.

### Neighbor solicitation

Questo somiglia moltissimo ad ARP. Si usano pacchetti icmpv6 anziché ARP. Può utilizzare i meccanismi di autenticazione e cifratura previsti da IPSEC. Usa indirizzi multicast anziché broadcast, e questo dovrebbe portare ad una maggiore efficienza. Noi sappiamo che ARP è un problema nelle reti locali: l'ARP

request è broadcast, quindi quel pacchetto passa attraverso tutti gli switch, finisce su tutte le interfacce, tutte le interfacce lo riconoscono come proprio, e siccome lo riconoscono come proprio generano un'interruzione hardware, il pacchetto viene passato alla pila protocollare e viene processato magari per rendersi poi conto del fatto che quel pacchetto non era importante per la macchina. Questo è molto inefficiente, ed ha problemi di sicurezza. L'idea è quella di mettere una patch a questo, utilizzando questi indirizzi multicast anziché broadcast. Neighbor solicitation si fa per ottenere un indirizzo fisico di un altro nodo. Però, se io un indirizzo fisico già ce l'ho, perché ho costruito l'indirizzo IPv6 a partire dall'indirizzo MAC, che senso ha andare a chiedere quale sia l'indirizzo fisico di una macchina? Non è detto che si usi quel meccanismo, quindi l'idea che tu abbia a disposizione ARP, o una cosa che gli assomiglia, anche in IPv6 è ragionevole. Il nodo calcola l'indirizzo solicited node corrispondente all'indirizzo IPv6 del destinatario, e a questo punto invia a questo indirizzo un pacchetto di neighbor solicitation specificando l'indirizzo IPv6 del destinatario nel campo dati. La richiesta viene fatta multicast, non broadcast. Il destinatario, se presente, risponde con un pacchetto di neighbor advertisement in cui chiaramente inserisce il proprio indirizzo fisico. Dobbiamo capire come è fatto questo indirizzo solicited node multicast. Ad ogni indirizzo IPv6 unicast corrisponde un indirizzo multicast solicited node, ma non è vero il viceversa: ad un indirizzo solicited node multicast possono corrispondere diversi indirizzi IPv6. L'indirizzo solicited node multicast si forma aggiungendo gli ultimi 24 bit dell'indirizzo al prefisso `ff02::1:ff00:0/104`. Riduce le collisioni in caso di indirizzi formati da Interface ID hardware e riduce il numero di gruppi multicast a cui partecipare in caso di indirizzi multipli con lo stesso Interface ID.

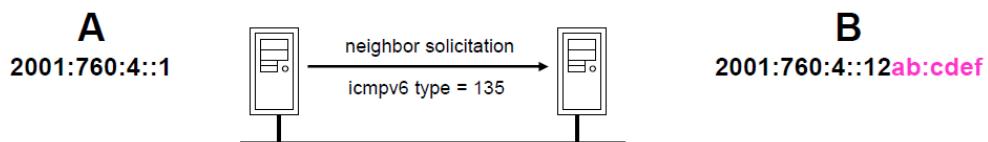


Figura 281 - neighbor solicitation: esempio

Quando prendi un indirizzo IPv6 fai parte automaticamente di un certo gruppo multicast, quello che si ottiene prendendo il proprio indirizzo IPv6, staccandoci gli ultimi 24 bit e appiccicandoli a questo prefisso. A questo punto, se io voglio sapere qual è il MAC address di una certa macchina di cui conosco l'indirizzo IPv6 chiedo al gruppo multicast che corrisponde all'indirizzo IPv6 "Qualcuno sa a quale MAC address corrisponde?". Adesso, questo gruppo multicast, quanto è popoloso? Quante macchine vado ad interpellare? Prima con IPv4 andavo a chiedere a tutta la LAN, adesso mando la richiesta a poche macchine: appartengono a quel gruppo multicast tutte le macchine che finiscono con un indirizzo con gli ultimi 24 bit uguali. La probabilità che ce ne siano già 2 è molto bassa. Si evita di mettere in difficoltà tutta la LAN ogni volta che serve un MAC. Questa cosa scala ancora di più, non solo dal punto di vista del numero di macchine, ma anche dal punto di vista del numero di indirizzi che può assumere una macchina: un'interfaccia IPv6, contrariamente a quanto accade per le macchine IPv4, ha tanti indirizzi IPv6, perché

hai almeno un indirizzo link local, ma poi tipicamente puoi avere tanti indirizzi globali. Questo implica che la macchina sia iscritta a tanti gruppi multicast solicited node? No. Tipicamente una macchina anche se ha più indirizzi IPv6, negli ultimi 64 bit per tutti questi indirizzi ha lo stesso interface ID, e se per tutti questi indirizzi gli ultimi 64 bit sono uguali a maggior ragione sono uguali gli ultimi 24 bit. Quindi, anche se una macchina in linea di principio ha tanti indirizzi IP, si iscrive a pochissimi gruppi multicast (se si usa l'indirizzo MAC per costruire l'indirizzo si iscrive l'interfaccia ad un solo gruppo). Dal punto di vista tecnologico, che cosa succede veramente nelle LAN attuali? Io sono una macchina che deve conoscere il MAC address corrispondente ad un certo indirizzo IPv6. Eseguo l'algoritmo. Qual è il gruppo multicast che dovrà corrispondere a questo indirizzo? Lo so facilmente. Adesso prendo un pacchetto IPv6 e lo spedisco nella rete al gruppo multicast che ho trovato. A chi va, sulla LAN, questo pacchetto? Dal punto di vista del MAC address, che MAC address si prende come MAC address destinatario? Gli switch come lo trattano? Le interfacce a cui è destinato il pacchetto, come lo trattano?

Prima possibilità: quando il pacchetto viene dato al livello 2 perché sia spedito sulla rete, quel pacchetto deve finire in un pacchetto di livello MAC, perché bisogna specificare l'indirizzo del destinatario. Se la mia scheda, dal punto di vista multicast, è un po' stupidotta prende quel multicast e lo spedisce broadcast. In realtà le schede possono essere un po' più furbe. C'è una corrispondenza tra i gruppi multicast che abbiamo costruito e i gruppi multicast IEEE 802: ad ogni indirizzo solicited node multicast corrisponde un indirizzo multicast MAC 802. Gli switch che fanno? L'ideale sarebbe che uno switch, ricevuto un pacchetto diretto ad un certo gruppo multicast MAC spedisce questo pacchetto soltanto sulle porte su cui è presente una qualche macchina di quel gruppo multicast. Gli switch attuali questo non lo sanno fare, tipicamente uno switch quando riceve un pacchetto multicast lo prende e lo spedisce su tutte le porte (tranne quella di ricezione), quindi l'idea di diminuire il traffico sulla rete non funziona. Il pacchetto genera un interrupt hardware per tutte le macchine della rete oppure no? In realtà se le cose sono fatte bene almeno qui si risparmia, perché la macchina del destinatario ha detto al proprio MAC layer "Mettiti in ascolto su questo specifico gruppo multicast", questo vuol dire che il pacchetto viene visto dalla NIC, e osservato come proprio o non proprio. Se viene osservato come non proprio, a questo punto il pacchetto non viene passato all'hardware della macchina, quindi almeno da questo punto di vista ci abbiamo guadagnato. Su uno switch comunque si possono fare delle configurazioni manuali, e dire su quali gruppi multicast sta una porta, ma non va troppo bene, perché stai inchiodando le macchine; stai facendo una cosa che somiglia molto a costruire una VLAN che corrisponde a quel gruppo multicast. Esistono comunque dei protocolli per alleviare questo problema.

#### Neighbor unreachability detection

Tu sei un livello 3, che per tua natura puoi osservare tutte le conversazioni che sono stabilite dal livello applicativo (riscontri compresi). IP può tenere traccia di queste conversazioni, può utilizzare informazioni che provengono da protocolli di livello superiore, e può specificare alla macchina che fa uso di icmpv6 che

un certo vicino non è più raggiungibile, magari osservando che partono pacchetti TCP e non arrivano riscontri in direzione opposta. Anche se non si usa TCP si può mettere in parallelo al normale traffico sostanzialmente dei ping, anche se nessuno te l'ha chiesto. Questo fa sì che tu come macchina possa tenere sotto controllo un insieme di vicini continuamente, e che tu possa segnalare ai livelli più alti se un vicino non né più raggiungibile in tempi molto rapidi. Quindi, in definitiva, Se un nodo non ha informazioni sulla raggiungibilità di un vicino, gli invia pacchetti unicast di Neighbor Solicitation in parallelo al traffico normale. Se non ottiene risposta, cancella il vicino dalla Neighbor Cache e ripete il procedimento di Neighbor Solicitation, perché il nodo potrebbe aver cambiato indirizzo fisico. Se questo fallisce, il vicino è irraggiungibile, le conseguenze dipendono dal tipo di vicino:

- Host: viene notificato un errore ai protocolli di strato superiore;
- Router: il nodo seleziona un altro router.

### Neighbor discovery

Attraverso l'autoconfigurazione stateless i nodi IPv6 possono connettersi alla rete senza dover configurare manualmente l'indirizzo, senza bisogno di un server DHCP. Possiamo costruire gli indirizzi partendo dall'interface ID, e fin dall'inizio i nodi possono comunicare tra loro utilizzando indirizzi link local. Per capire bene questa storia c'è da scoprire un'altra funzionalità interessante, che è quella di router advertisement. I router che stanno su link IPv6 sono obbligati a spedire periodicamente dei messaggio di router advertisement, inviati a tutti i nodi sul link (gruppo multicast FF02::1). Questi pacchetti annunciano la presenza del router, forniscono alcuni parametri importanti (MTU, reachable time, hop limit da utilizzare all'interno della LAN) e contengono una lista dei prefissi assegnati al link; il router quindi dice cose del tipo "su questo link si possono utilizzare questi cinque prefissi", e le macchine ne prendono coscienza. Questi messaggi inoltre annunciano la presenza del router:

- Specificano il suo indirizzo IPv6 link-local (perché il router ha un indirizzo link local);
- Specificano il suo indirizzo di livello 2;
- Indicano se è disponibile ad essere il default router.

Quando un host riceve un pacchetto di router advertisement costruisce un pacchetto IPv6 mettendo di fronte all'interface ID di cui già dispone uno dei prefissi ottenuti dal router advertisement. Per ciò che riguarda il default gateway si può utilizzare l'indirizzo link local del router.

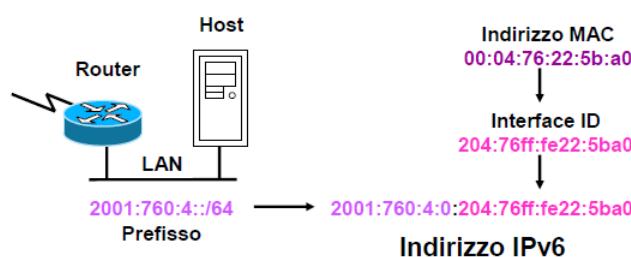


Figura 282 - indirizzo ipv6 dal router advertisement

## Router solicitation

È vero che un router manda continuamente i pacchetti di router advertisement, però magari un host che è impaziente vuole subito avere informazioni sul link, quindi manda un messaggio di router solicitation, che viene mandato al gruppo multicast che corrisponde a tutti i router sul link (FF02::2). Ogni router che fa parte di questa LAN risponde con un pacchetto unicast di router advertisement indirizzato al nodo che ha emesso la richiesta. È un po' lo stesso meccanismo di scambio informazioni degli access point wi-fi.

## Duplicate address detection

Prima di assegnare un indirizzo ad un'interfaccia un nodo deve comunque verificare se l'indirizzo è stato già assegnato ad un nodo dello stesso link. Se si utilizza l'autoconfigurazione stateless questo non dovrebbe mai accadere, ma non si sa se tutti i nodi sul link utilizzano effettivamente l'autoconfigurazione stateless. Il nodo che vuole prendersi l'indirizzo invia una serie di pacchetti di Neighbor Solicitation all'indirizzo solicited node corrispondente all'indirizzo che vorrebbe assegnare (tentative address). L'indirizzo sorgente è l'unspecified address (::). Se non c'è risposta a questo pacchetto di richiesta l'indirizzo è valido e può essere assegnato.

## Procedura di autoconfigurazione

All'avvio un host configura gli indirizzi link local come specificato: uno per ciascuna interfaccia, e con il processo di duplicate address detection specifica se l'indirizzo sia unico sul link. Se è unico lo assegna a quell'interfaccia. Che cosa vuol dire verifico se l'indirizzo è in uso? Dopo quanto tempo devo aspettare, senza ottenere una risposta, per stabilire che quell'indirizzo non lo usa nessuno? C'è un timer configurato sulla macchina che me lo dice. A questo punto ha un indirizzo link local su tutte le interfacce, e puoi comunicare con tutti i nodi sul link a cui sei connesso. A questo punto posso fare ulteriori operazioni: per ogni router advertisement che ricevo (per esempio a fronte di una router solicitation) io aggiungo il router alla lista dei router disponibili, configuro un indirizzo per ogni prefisso nell'annuncio. Che ci faccio con tanti indirizzi? Cosa farci con link local lo sappiamo, ma non basta un solo indirizzo globale? Qual è la killer application? Ho maggiore raggiungibilità, posso per esempio annunciare in giro più prefissi, aumento la mia tolleranza ai guasti. Posso fare però un'altra cosa più sofisticata: posso orientare una parte del mio traffico su un provider e una parte su un altro, per esempio per ottimizzare. Questo in IPv4 non lo posso fare, posso fare solo del multihoming molto semplice.

## Esempio di autoconfigurazione

- L'host ha indirizzo MAC 00:04:76:22:5b:ao
  - Il suo Interface ID è 204:76ff:fe22:5bao
- I prefissi associati alla LAN sono

- 2001:760:4::/64
- 2002:c1cc:a102::/64

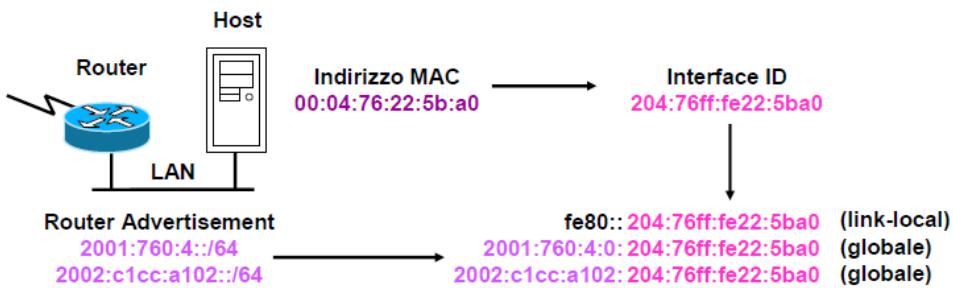


Figura 283 - autoconfigurazione: esempio

L'autoconfigurazione stateless ha un buco: non prevede l'autoconfigurazione del name server di default, quello va specificato manualmente.

#### Tempo di vita degli indirizzi

È uno strumento che può andare nella direzione multiprovider; gli indirizzi hanno un certo tempo di vita.

In particolare, nel router advertisement, ad ogni prefisso sono associati due tempi di vita:

- valid lifetime;
- preferred lifetime.

Se si porta il preferred lifetime a zero l'indirizzo è deprecato. Se valid lifetime vale zero allora non è più valido, e non può più rimanere assegnato all'interfaccia. Io ti do un prefisso, e tu lo puoi utilizzare per farti i tuoi indirizzi, ad un certo punto io vorrei che tu quel prefisso non lo utilizzassi più, magari perché sta scadendo il contratto con un certo provider e vorrei che utilizzassi un altro prefisso. Che cosa faccio? Ti porto a zero il preferred lifetime, in uno dei pacchetti in cui lo manifesto. Questo vuol dire che le connessioni che hai già in piedi le puoi continuare ad utilizzare, però nuove connessioni, con quell'indirizzo lì come indirizzo mittente, non le puoi più fare. Dopodiché aspetti un po' e anche il valid lifetime scende a zero, e questo vuol dire che anche le connessioni attive sono tagliate e non puoi più utilizzare quell'indirizzo. I due tempi sono fatti in modo tale che il meccanismo di taglio non sia troppo drastico. Gli host a questo punto devono sempre rimanere in ascolto dei messaggi di router advertisement, perché tu ti sei preso un indirizzo, ma quell'indirizzo può sempre venire deprecato.

## Renumbering

Con questo tempo di vita dei prefissi si può fare il renumbering automatico degli host, cioè: vuoi cambiare provider ad un certo istante, rinumerando tutte le interfacce? Configuri un nuovo prefisso che hai a disposizione su un router porre a zero il preferred lifetime di quello vecchio. Naturalmente questo funziona perché il router è identificato dall'indirizzo link local, quindi non ci sono problemi di raggiungibilità del router. A questo punto gli host utilizzano entrambi i prefissi per un po', e alla fine del processo di renumbering il vecchio prefisso viene rimosso dal router, e gli indirizzi vengono rimossi alla fine del valid lifetime.

## Gestione avanzata degli indirizzi

### Router renumbering

Qui si possono anche, attraverso icmpv6, cambiare i prefissi che sono associati ai router: puoi avere una stazione di gestione da qualche parte nella rete, e questa stazione distribuisce ai router i prefissi che i router possono distribuire alle macchine. Questo rende il renumbering più efficace. Naturalmente per fare una cosa del genere devi utilizzare obbligatoriamente IPSEC: se ti impossessi di questo protocollo e riesci a modificarne i pacchetti appositamente, puoi indurre un renumbering sbagliato su tutte le macchine della rete e non far funzionare più nulla.

### Configurazione stateful

Come in IPv4 anche in IPv6 c'è un DHCP, perciò la configurazione la si può fare in modo stateful anziché stateless, le macchine che non hanno un indirizzo si rivolgono ad un server, il server gli dà un indirizzo e tutto funziona sostanzialmente come in IPv4.

## Transizione ipv4-ipv6 e meccanismi di compatibilità

### Il processo di transizione

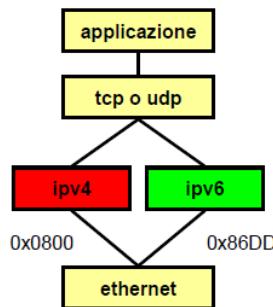
Problema: IPv4 e IPv6 sono due protocolli di livello 3 che non sono compatibili. Svolgono le stesse funzioni, e per potersi affermare IPv6 deve in qualche modo immaginare una coesistenza, una qualche forma di compatibilità con i dispositivi IPv4 esistenti. L'esistenza di meccanismi di transizione semplici ed efficaci è essenziale per il successo di IPv6. Il principio è: la transizione da IPv4 a IPv6 non è istantanea, ma c'è comunque un lungo periodo di coesistenza (che potrebbe durare decenni) tra i due protocolli. Lo scenario è questo: c'è già una fase iniziale in cui una rete IPv6 inizia ad esistere ma si appoggia largamente all'infrastruttura IPv4. I nodi IPv6 utilizzano prefissi IPv4 esistenti. C'è poi (non in questo momento) una fase intermedia in cui i due protocolli coesistono e si può immaginare una fase finale in cui sarà l'infrastruttura IPv4 ad appoggiarsi all'infrastruttura IPv6, che nel frattempo avrà preso il sopravvento.

In generale le applicazioni devono comunque essere modificate per poter utilizzare IPv6. Ci sono tre categorie fondamentali di meccanismi di transizione:

- meccanismi implementati sugli host;
  - Host Dual Stack;
  - BIS, BIA, ...;
- Meccanismi implementati a livello di rete;
  - Tunnel:
    - Manuali, 6to4, automatici;
    - Altri: ISATAP, Teredo, ...;
  - Rete Dual Stack:
- traduttori di protocollo;
  - SIIT e NAT-PT;
  - altri TRT

#### *Host dual stack*

La situazione è molto semplice: invece che avere una pila IPv4 da sola hai una pila IPv4 e una pila IPv6. A questo punto un nodo dual stack, oltre a implementare i due protocolli ha anche due indirizzi, sulla stessa interfaccia. Le applicazioni IPv4 usano IPv4, per le applicazioni che usano IPv6 il DNS risolve sia indirizzi IPv4 che indirizzi IPv6. A questo punto: se hai un indirizzo IPv6 per una certa destinazione utilizzi IPv6, se la destinazione ha soltanto l'indirizzo IPv4 usi IPv4.



*Figura 284 - host dual stack: funzionamento*

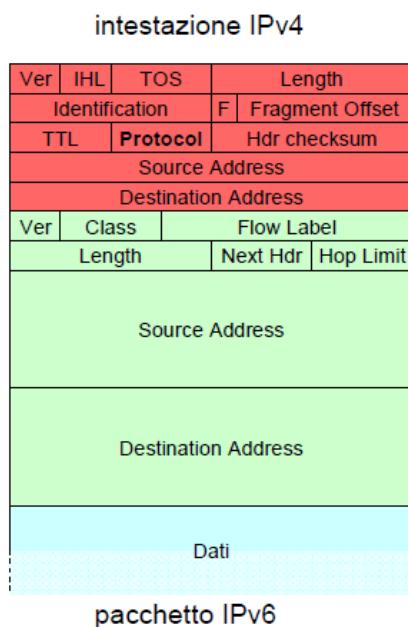
La figura evidenzia un aspetto interessante: ad IPv4 e ad IPv6 è associato un diverso EtherType. Questo implica che se tu vuoi distinguere un pacchetto IPv4 da un pacchetto IPv6 non hai bisogno di accedere al campo version del pacchetto di livello 3, è già il livello 2 che smista il pacchetto. Questo implica che il campo version nei pacchetti è del tutto inutile. Gli host dual stack sono semplici, non richiedono supporto, ma non riducono il numero di indirizzi IPv4 necessari. Hai anche bisogno di gestire una doppia infrastruttura di rete. Cosa vuol dire “richiede la gestione di una doppia infrastruttura di rete”? Se una macchina ha due indirizzi su una certa scheda, uno IPv4 e uno IPv6, quegli indirizzi devono essere entrambi annunciati. Questo vuol dire che tu hai un'infrastruttura di instradamento globale nella quale si

sovrappongono gli annunci di indirizzi IPv4 e di indirizzi IPv6. Questo non semplifica la situazione del routing, anzi, tendenzialmente la complica.

## Meccanismi basati su tunnel

## Tunnel Ipv6-in-IPv4

Qual è l'idea? Hai delle zone IPv6, che sono tante isolette sulla rete, e le colleghi tra loro utilizzando dei tunnel. Questo permette di fare IPv6 disponendo soltanto in parte di un'infrastruttura IPv6 nativa. Quando attraversi il tunnel i pacchetti IPv6 sono incapsulati in pacchetti IPv4, che vengono decapsulati all'uscita dal tunnel.



*Figura 285 - tunnel ipv6-in-ipv4*

I pacchetti vengono a questo punto instradati normalmente sulla rete IPv4. Naturalmente gli estremi devono essere nodi dual stack, e se lo guardi al di fuori, dal punto di vista di IPv6, hai che il tunnel è un singolo hop IPv6. Attenzione all'MTU, perché l'MTU del tunnel è inferiore di 20 byte a quella normale per via della presenza di IPv4. Posso fare un tunnel:

- Da router verso router;
  - Da host verso router;
  - Da host verso host.

I tunnel più facili da utilizzare sono quelli configurati staticamente. Si configurano configurando normalmente gli estremi. Un'altra possibilità è il tunnel broker, in una situazione tutt'altro che statica: ho un'applicazione raggiungibile tramite IPv4, e questa applicazione mi serve per creare dinamicamente dei router configurati. Sto in una zona in cui c'è solo IPv4, e mi serve un tunnel per andare nella zona IPv6. Il broker identifica l'utente che fa questa richiesta e in funzione della richiesta il broker configura

automaticamente il tunnel verso un router che utilizza IPv6. Questo si presta ad un utilizzo occasionale dell'accesso alla rete IPv6.

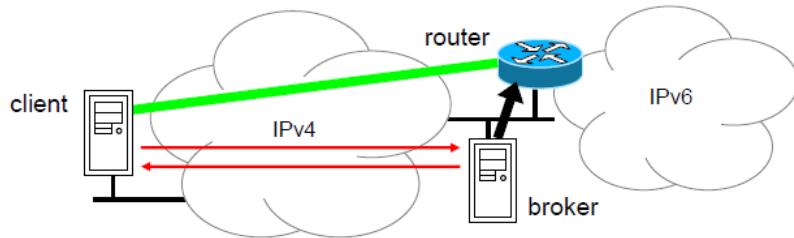


Figura 286 - tunnel broker

Nei tunnel automatici l'indirizzo IPv4 dell'estremo del tunnel è determinato automaticamente dall'indirizzo del destinatario. Questo vuol dire che quando il pacchetto arriva sul router, che non sa come fare per instradarlo, perché non ha fisicamente strade verso la destinazione utilizza un tunnel che viene identificato automaticamente dall'indirizzo del destinatario. La cosa più interessante che si può fare con questi tunnel è la scelta dei tunnel 6to4.

#### Tunnel 6to4

È l'applicazione più importante dei tunnel automatici. Consentono di entrare nella rete IPv6 ad un costo molto basso, dal punto di vista amministrativo. Hai un po' di macchine di un site (una struttura, un edificio, un'organizzazione) nella quale vuoi affermare IPv6. Si configurano le macchine per utilizzare IPv6; è immaginabile che ci siano altri site che fanno questa scelta. C'è una sorta di arcipelago di site che sono IPv6-compliant e che sono tra loro connessi perché vivono in un mondo che è IPv4.

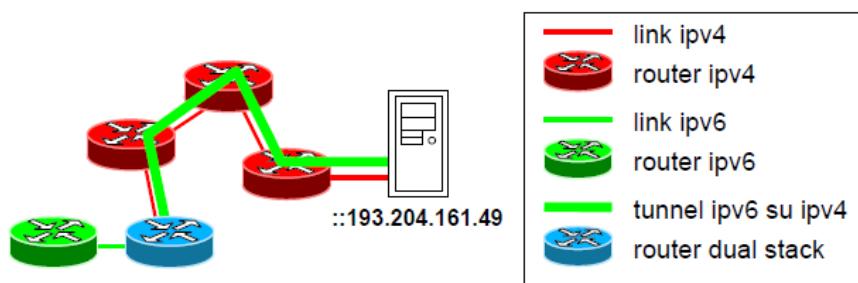


Figura 287 - Tunnel configurati

Come si fa per connetterli tra loro? La rete interna a ciascun site la si realizza con un piano di indirizzamento basato sul prefisso 2002::/16 (il prefisso 6to4). Gli indirizzi delle macchine che stanno dentro site 1 sono fatti così:

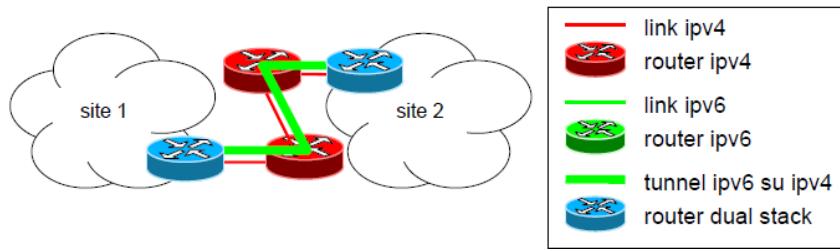


Figura 288 - tunnel 6to4

L'indirizzo IPv4 è l'indirizzo del router che consente di accedere al site 1. SLA è semplicemente la subnet. L'interface ID possiamo farlo come vogliamo (autoconfigurato, random...).

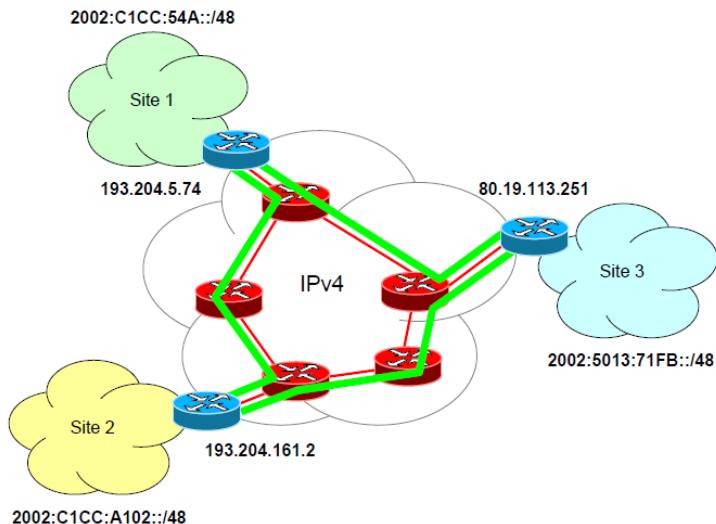


Figura 289 - tunnel 6to4: esempio

Con gli indirizzi fatti così, se una macchina che sta dentro site 1 vuole parlare con una macchina che sta dentro site 2 il pacchetto viene instradato su un router che è il router di uscita dal site, perché quel prefisso lì dentro non c'è: se quella macchina sta dentro site 2 il suo indirizzo sarà fatto così: 2002 (6to4), e poi ci sarà l'indirizzo IPv4 pubblico del router di accesso a site 2. Il router di uscita da che quell'indirizzo non è suo, ma di un altro site; siccome sa qual è l'indirizzo del router che sta all'ingresso di site 2, usa questa informazione per mettere in piedi, al volo, un tunnel da site 1 a site 2. Non hai bisogno di costruire un piano di indirizzamento completo rispetto agli indirizzi aggregatable local di IPv6, bastano i prefissi 2002, poi ci si fa il proprio piano di indirizzamento. Questo fa sì che si possano avere quanti site si vogliono, perché dietro un indirizzo IPv4 ci si può mettere un intero site, e questo consente effettivamente di risparmiare indirizzi IPv4.

### Relay router

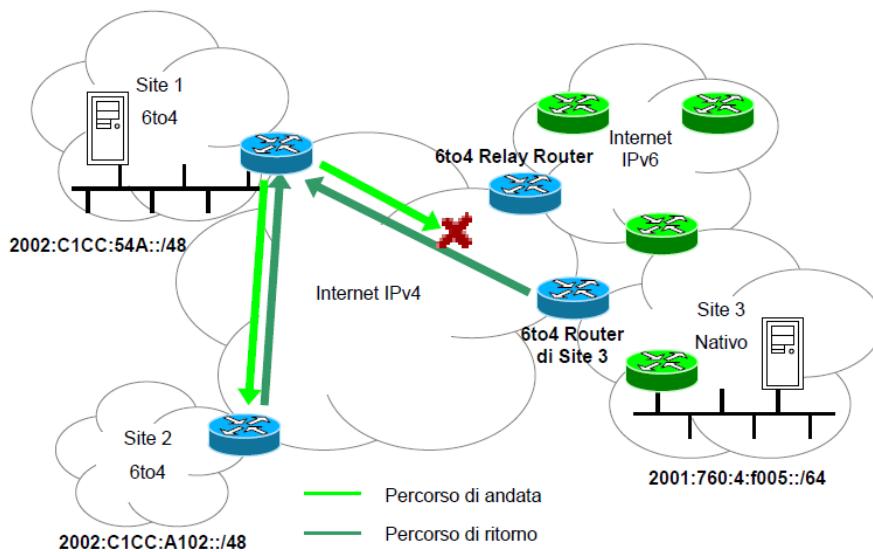


Figura 290 - 6to4: senza relay router

In questa figura c'è un altro attore: oltre ai site 6to4, ciascuno dei quali col proprio piano di indirizzamento, c'è una rete IPv6 nativa, con indirizzi aggregatable global perfettamente distribuiti. La questione è questa: se la macchina che sta in site 1 vuole comunicare con la rete IPv6 nativa, come se la cava? Non c'è un tunnel da mettere in piedi automaticamente. Quello che si fa è utilizzare dei router particolare, i relay router: servono ad accedere da siti 6to4 alla rete IPv6 nativa. Io sono un relay router, e lo dico in tutta la rete IPv4. A questo punto arriva un pacchetto da site 1 che ha come indirizzo del destinatario un indirizzo IPv6 vero, non come quelli che stanno sui siti 6to4. Il pacchetto viene instradato verso il router di site 1, che adesso lo deve instradare verso la rete IPv6 nativa; non sa come fare, ma sa che su internet IPv4 sono annunciati questi relay router. Riceve questo annuncio, e mette in piedi un tunnel verso il relay router; il relay router sa che può essere contattato da router che sono router di accesso dei siti 6to4, e quando riceve pacchetti che sono tunnelati attraverso rete IPv4, che vengono da router 6to4, decapsula i pacchetti e li spedisce sulla rete IPv6 nativa.

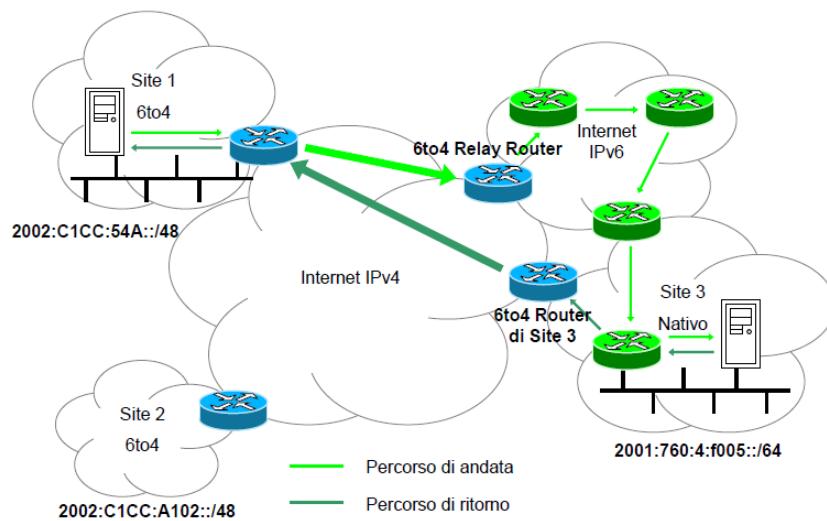


Figura 291 - 6to4: con relay router

La cosa è più complicata quando la rete IPv6 nativa diventa una rete gigante, e a questo punto non ha senso che ci sia soltanto un relay router, sarebbe il destinatario di una montagna di traffico. Come si fa a risolvere il problema? Abbiamo siti 6to4 in giro per il mondo (un po' negli Stati Uniti, un po' in Europa e un po' in Asia); questi tra loro si tunnelano benissimo. Poi c'è la rete IPv6 nativa, che pian piano si spande per il mondo, ed è parte in Europa, parte negli Stati Uniti e parte in Asia. A questo punto si ha un sito 6to4 che sta negli Stati Uniti, e si deve mandare un pacchetto verso la rete IPv6 nativa. Questo pacchetto viene inviato ad una macchina che nella rete IPv6 nativa è negli Stati Uniti. Se c'è soltanto un relay router, che sta a Milano, costringi il pacchetto a viaggiare nella rete IPv4 fino a Milano, dagli Stati Uniti, poi dentro la rete IPv6 nativa a ritornare da Milano negli Stati Uniti. Il problema si affronta con tecniche già discusse: invece di utilizzare un relay router ne metto diversi.

Adesso ci mettiamo nei panni del router 6to4 che ha un pacchetto per la rete IPv6 nativa; si trova davanti 5 relay router che può utilizzare. Sulla base di cosa sceglie un router piuttosto che un altro? Segue l'andamento di BGP: se c'è una cosa che funziona bene nel backbone è il routing interdominio. La cosa più semplice fa fare è annunciare i relay router in modo anycast: ai 5 ipotetici router si associa la stessa /24, e poi annunciamo quella /24 in maniera assolutamente libera nel routing interdominio. A questo punto siamo anche al riparo dalle fluttuazioni delle rotte: passi prima per un relay router, poi per un altro, tanto il server a cui devi arrivare è sempre lo stesso, e sta in un dominio IPv6 nativo.

- Vantaggi:
  - Semplice da configurare;
  - Permette di utilizzare IPv6 senza disporre di indirizzi e senza avere un provider IPv6 nativo;
  - Non richiede alcun supporto particolare ai nodi: i prefissi 6to4 possono essere appresi tramite autoconfigurazione stateless come qualsiasi altro prefisso.

- Svantaggi:
  - Gli indirizzi IPv6 sono legati all'indirizzo IPv4 del router di bordo: se cambia indirizzo IPv4, l'intero site deve essere rinumerato;
  - Il relay router può essere molto lontano sia dal nodo sorgente (in IPv4) sia dal nodo destinazione (in IPv6).

#### *Rete dual stack*

Puoi pensare di utilizzare tunnel per un certo tempo, ma se vuoi che tutto funzioni devi avere una rete dual stack. Una rete dual stack è una rete in cui non soltanto c'è un doppio stack sugli end system, ma anche sugli intermediate system. Quindi puoi avere a questo punto un backbone dual stack in cui hai la stessa topologia per IPv4 e IPv6 e anche le due pile protocollari per IPv4 e IPv6. Questo implica che tu abbia a disposizione tabelle (e protocolli) di instradamento IPv4 e IPv6 su tutti i router.

Meccanismi basati su traduzione di protocollo

#### *Traduttori di protocollo*

Trasformano pacchetti IPv4 in pacchetti IPv6 e viceversa. Questa può essere un'alternativa ai nodi dual stack, e può effettivamente consentire un risparmio di indirizzi. Il fatto che consentano di passare da un protocollo all'altro implica che sia importante dove posizionare questi traduttori, giacché tutto il traffico tradotto passa per il nodo traduttore, e questo naturalmente pone problemi di robustezza, sicurezza, e può rendere inutile una parte della rete dual stack. Possono esserci vari tipi di traduttori di protocollo:

- A livello IP: SIIT, NAT-PT;
- A livello di trasporto: TRT;
- Modificando la pila protocollare dell'host: BIS, BIA.

In molti di questi meccanismi si fa l'ipotesi di rappresentare indirizzi IPv4 come particolari indirizzi IPv6.

#### *Traduttori a livello IP*

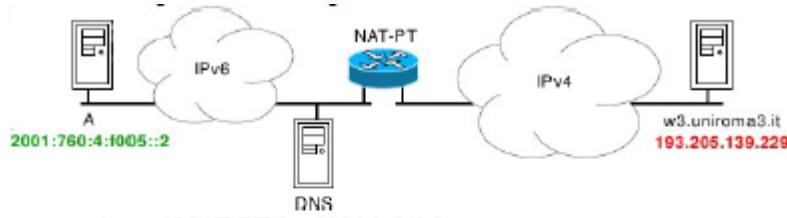
##### *SIIT*

SIIT sta per Stateless IP/ICMP Translation Algorithm. È un meccanismo del tutto generale che permette a nodi IPv6 di comunicare con nodi IPv4 attraverso il traduttore. Gli indirizzi IPv4 sono mappati su indirizzi IPv6; in particolare, le destinazioni IPv4 sono indicate con indirizzi IPv6 IPv4-mapped (`::ffff:a.b.c.d`), che devono essere instradati verso il traduttore. I nodi ottengono indirizzi IPv4 temporanei che vengono mappati in indirizzi IPv6 "IPv4-translated" (`::ffff:o:a.b.c.d`) e usati come indirizzo sorgente. A questo punto il traduttore traduce i pacchetti in transito, e la traduzione è immediata, perché sia l'indirizzo del mittente che del destinatario sono scritti negli indirizzi IPv6 che sono gli indirizzi del pacchetto tradotto, gli indirizzi sono desunti direttamente dagli indirizzi IPv6. Io ho una macchina che sta in un mondo IPv6, e ciò che vede intorno è solo IPv6. Questa macchina deve mandare un pacchetto ad una macchina che è solo IPv4. A questo punto, i contenuti molto spesso ce li hai solo nel mondo IPv4, allora che cosa fai? Tu

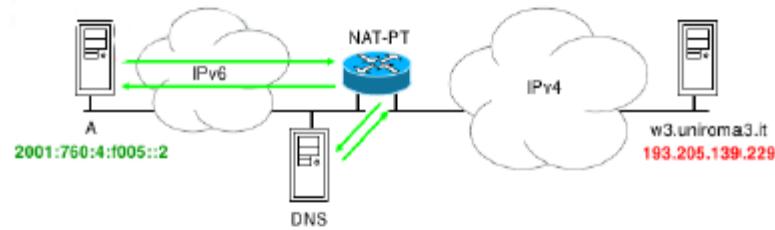
macchina IPv6 conosci l'indirizzo IPv4 del destinatario del pacchetto (te lo dà il DNS). Prendo questo indirizzo e lo infilo in un indirizzo IPv6 che è l'indirizzo IPv4-mapped. A questo punto mando fuori un pacchetto che ha come indirizzo del destinatario l'indirizzo IPv4-mapped corrispondente all'indirizzo IPv4. Questo pacchetto deve essere instradato dentro la rete IPv6 in modo che arrivi sicuramente al traduttore di protocollo, che quando lo riceve butta via il prefisso (la parte convenzionale, FFFF), dentro ci trova l'indirizzo IPv4 vero del destinatario, costruisce l'indirizzo IPv4 vero e lo spedisce al destinatario. C'è anche l'altro verso. Dentro un pacchetto che la macchina IPv6 spedisce ci deve essere anche un indirizzo IPv6 mittente, come me lo scelgo? Non posso mettere uno dei miei indirizzi IPv6? Quando arriverà la risposta dal traduttore di protocollo, che indirizzo mittente userà per me nella rete IPv4 nativa? È un problema. Il nodo ottiene un indirizzo IPv4 temporaneo che viene mappato nell'indirizzo IPv6, e questo indirizzo viene utilizzato come indirizzo mittente. Tu ti approvvigioni in pratica di un indirizzo IPv4 host e usi anche questo per costruirti un indirizzo IPv6, che viene utilizzato come indirizzo IPv6 mittente. Quando il pacchetto arriva sul traduttore di protocollo anche per ciò che riguarda l'indirizzo IPv4 mittente il traduttore di protocollo fa il giochetto che fa sull'indirizzo IPv4 del destinatario. Il passaggio sulla rete IPv4 è completamente stateless, perché sia all'andata che al ritorno il traduttore di protocollo non dovrà nient'altro che ricavare all'andata l'indirizzo IPv4 dall'indirizzo IPv6, e al ritorno l'indirizzo IPv6 dall'indirizzo IPv4. Quali sono i punti critici di questa storia? Continuano a servire indirizzi IPv4, nel nodo di partenza (in modo tale che il traduttore non debba farsi una tabella di conversione). Questo vuol dire che gli indirizzi IPv4 devono stare nei pool di indirizzi che stanno a disposizione dei server che le macchine IPv6 possono interpellare al volo per farsi un indirizzo IPv4, quindi non è che si risparmia tanto. Ci sono altri svantaggi: uno è sicuramente che continui ad aver bisogno di certi pool di indirizzi IPv4 da usare come indirizzo mittente, quali sono gli altri? Nella rete IPv4 quei prefissi IPv4 che sono utilizzati come indirizzi mittente per SIIT devono essere comunque annunciati, e devono essere annunciati dalle reti che sono reti dell'area IPv6, e questo è complicato perché tu aggiungi al normale routing IPv4 la complessità di annunciare quei prefissi. C'è un analogo svantaggio in termini di routing nella rete IPv6? Nel routing IPv6 devo annunciare anche i prefissi speciali che ho utilizzato per fare routing fino al destinatario.

#### NAT-PT

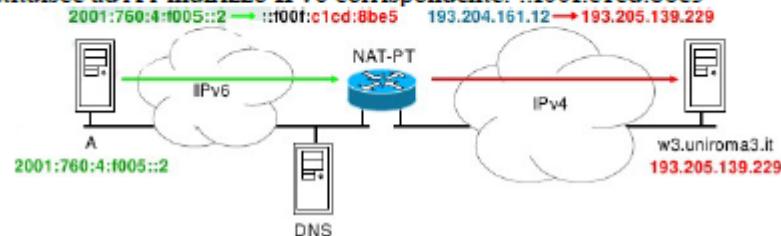
Combina le tecniche di NAT con SIIT, ed è simile al NAT IPv4. Il nodo traduttore dispone di un pool di indirizzi IPv4 che vengono assegnati ai nodi che lo utilizzano. Questo traduttore a questo punto deve mantenere lo stato delle associazioni. Gli indirizzi IPv4 sono rappresentati mediante indirizzi IPv6 aggiungendo i 32 bit dell'indirizzo ad un prefisso arbitrario di 96 bit instradato verso il traduttore, quindi si ha una mappatura IPv6 → IPv4 dinamica, e IPv4 → IPv6 deterministica. La traduzione di pacchetto avviene come in SIIT, e se si vogliono fare le cose per bene ci si accosta un meccanismo di traduzione delle risposte DNS, e questo consente di implementare questo meccanismo in maniera completamente trasparente. In che modo?



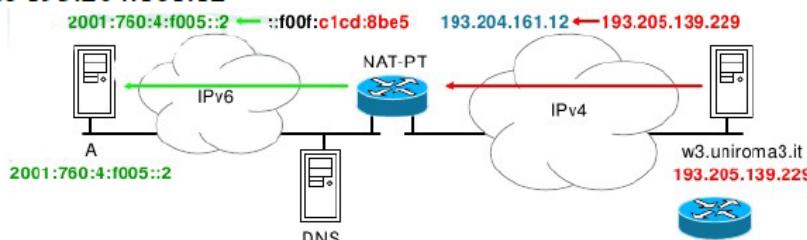
- Il prefisso associato al NAT-PT è ::f00f:0:96
  - Sulla rete IPv6 esso è instradato verso il NAT-PT
- Il pool di indirizzi IPv4 è 193.204.161.0/26
  - Sulla rete IPv4 esso è instradato verso il NAT-PT
- A è un nodo IPv6-only e vuole connettersi al web server w3.uniroma3.it, che ha solo un indirizzo IPv4



- A fa una query sull'indirizzo IPv6 di w3.uniroma3.it
- Il DNS ALG nel NAT-PT intercetta la query:
  - Interroga il server DNS ed ottiene l'indirizzo IPv4: 193.205.139.229
  - Restituisce ad A l'indirizzo IPv6 corrispondente: ::f00f:c1cd:8be5



- A si connette a ::f00f:c1cd:8be5
- Il NAT-PT intercetta la connessione
  - Associa ad A un indirizzo IPv4 temporaneo dal pool:  
2001:760:4:f005::2 → 193.204.161.12
  - Mantiene traccia dell'associazione in una tabella di stato
  - Inoltra i pacchetti sulla rete IPv4 verso 193.205.139.229 utilizzando come indirizzo sorgente 193.204.161.12



- I pacchetti di risposta verso A vengono instradati verso il NAT-PT, tradotti in IPv6 ed inoltrati ad A
- A ha la percezione di essere connesso all'host IPv6
- ::f00f:c1cd:8be5 (mappatura deterministica)
- Il web server ha la percezione di essere connesso all'host IPv4 193.204.161.12 (mappatura dinamica)

Questo somiglia molto a NAT. Il DNS “taroccato” poi fa sì che contrariamente a quanto avviene in SIIT la macchina A non si accorge proprio di niente. Tutto questo ha gli stessi svantaggi del NAT IPv4:

- Fragilità;
- Necessità di algoritmi specifici per gestire protocolli che non siano semplici client/server a una sola connessione;
- Non permette connettività diretta da estremo a estremo.

Però di NAT in giro ce n'è tanto e quindi questa potrebbe essere la strada. Non è ancora largamente implementato, e il motivo di fondo è che noi usciamo da IPv4 per entrare in IPv6 anche per non aver niente a che fare con NAT, se dentro IPv6 poi abbiamo bisogno ancora di NAT c'è qualcosa che non torna.

#### BIS e BIA

Sono semplicemente dei meccanismi di traduzione implementati via software, e quello che puoi fare è in BIS effettuare la traduzione SIIT internamente al nodo, invece che passare per un traduttore di protocollo. BIA è una cosa molto più semplice: Permette di utilizzare IPv6 con applicazioni che non lo supportano (su un host dual stack), e traduce le chiamate alla socket API IPv4 a chiamate alla socket API IPv6.

### Source address selection multihoming

#### Source address selection

Come lo selezioni l'indirizzo mittente per la comunicazione? Tu hai una macchina IPv6, con tanti indirizzi associati ad una interfaccia, e si può anche immaginare che ci siano tanti indirizzi IPv6 associati ad una certa destinazione. La macchina, a livello IP, quale sceglie come indirizzo per connettersi ad un'altra macchina? Quale sarà usato? Gli algoritmi di source address selection funzionano essenzialmente in questo modo: si fa una query ad un DNS, e questa query tira fuori un certo numero di record; questi possono essere indirizzi IPv4, IPv6 e possono essere vari indirizzi IPv6. Come ci si regola? Dipende da quali indirizzi sono disponibili:

Se un host ha un indirizzo IPv6 link-local e un indirizzo IPv4 globale, ha senso preferire IPv4;

Se un host ha un indirizzo IPv6 globale e un indirizzo IPv4 autoconfigurato (169.254.x.x/16), ha senso preferire IPv6.

Con regole di questo genere si ordinano tutti gli indirizzi della destinazione in ordine di preferenza, e si provano gli indirizzi destinazione uno per volta scegliendo, per ciascun indirizzo destinazione, come indirizzo sorgente l'indirizzo più appropriato. L'indirizzo sorgente è scelto in base a una serie di regole:

- Usare il giusto scope;
- Evitare indirizzi deprecati;
- ...;

- Preferire l'indirizzo “più simile” (longest match);

L'ordine di preferenza degli indirizzi destinazione è stabilito da regole simili:

- Evitare destinazioni irraggiungibili;
- Usare il giusto scope;
- Evitare indirizzi sorgente deprecati;
- ...;
- Preferire l'indirizzo “più simile” (longest match).

### Multihoming

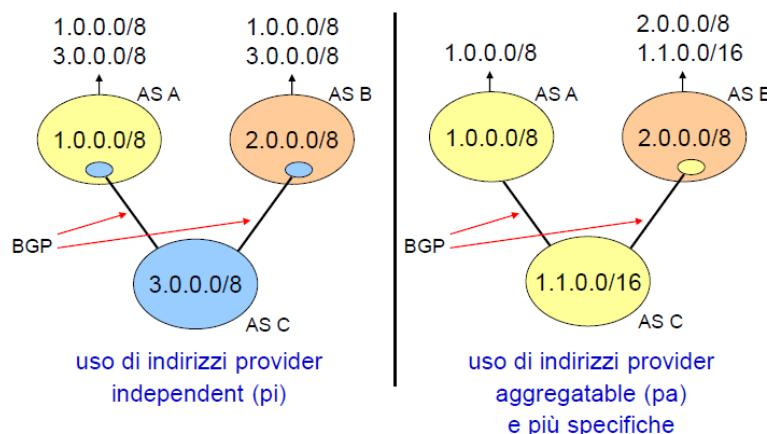


Figura 292 - multihoming IPv4

In IPv4 il multihoming si fa con BGP. Queste sono le due situazioni tipiche: quella di chi ha a disposizione indirizzi PI (provider independent) che vengono annunciati a diversi provider con BGP e di chi ha a disposizione indirizzi standard (qui si annunciano le più specifiche su diversi provider). Questo gioco sappiamo che porta ad una crescita della tabella di instradamento. In IPv6 si è cercato di mettere una patch anche da questo punto di vista, e la soluzione banale che si è cercata è quella di un customer che non fa peering BGP. L'idea è questa: andiamo a spostare il gioco di BGP più verso l'alto nella gerarchia di internet, immaginando che un certo numero di customer non parli BGP. Chiaramente non parlare BGP può essere un problema, perché se non parli BGP c'è il rischio che tu non abbia ridondanza, che non possa fare bilanciamento di traffico, e non si capisce per cosa si faccia il multihoming. D'altra parte se il customer non parla BGP la tabella di routing è più snella, e puoi pensare che le uniche entry nella tabella della default free zone siano le /32 di tantissimi provider. Si cerca di fare questo: se la gente che sta in basso non parla BGP, tutto il traffico delle rotte più specifiche e degli annunci “strani” che abbiamo visto non c'è più, e quindi vengono annunciati soltanto i grandissimi prefissi dei grandi provider, ma questa cosa la puoi fare a patto di continuare ad avere ridondanza, bilanciamento di carico e così via. L'approccio scelto per affrontare il problema è quello di separare il locator dall'identifier in un indirizzo IP. Qui si va proprio alla radice dell'indirizzamento IP, riscoprendo le funzioni fondamentali di un indirizzo IP: un indirizzo IP, anche se normalmente non ci si pensa, di funzioni ne svolge due:

- identificazione della macchina (dal punto di vista applicativo);
- identificazione della posizione del nodo nella topologia della rete (io so che una certa /15 sta là, una certa /24 sta là...).

Queste due sono due funzioni completamente diverse, e l'idea in IPv6 è di separare queste funzioni. Si definisce un Upper-Layer ID per identificare il nodo ai protocolli di strato superiore, e un locator che identifica la posizione del nodo nella rete. Puoi pensare di avere un ULID per ogni nodo, o per ogni connessione di livello superiore. Il locator lo usi come indirizzo sorgente e destinatario del pacchetto, con l'idea che un nodo abbia un locator per ogni indirizzo IPv6, e quindi, se hai multihoming, uno per ogni prefisso di multihoming.

### shim6

Per fare questa cosa si fa una variazione sulla pila protocollare introducendo uno strato che si chiama shim layer (ed in particolare shim6) che viene inserito tra IP e TCP/UDP. Guardiamo la storia attraverso un esempio:

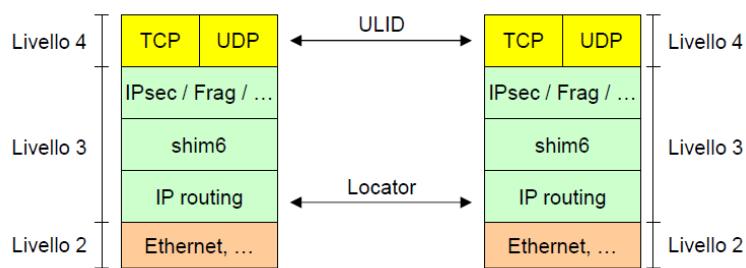


Figura 293 - shim6

Il TCP su una macchina decide di stabilire una connessione con il TCP che sta sull'altra macchina. Per mettere in piedi la connessione si sceglie un indirizzo IP mittente e un indirizzo IP destinatario (li sceglie con il meccanismo di address detection). Parte la connessione. Si scambiano un po' di pacchetti. Ad un certo punto ci si rende conto, da parte di IP, che i pacchetti non passano più, c'è un problema sulla rete. Stante il fatto che io ho diversi indirizzi IP posso provare ad utilizzare un diverso indirizzo IP mittente, chissà che le cose non vadano meglio. Utilizzo l'indirizzo dell'altro provider. Se questo lo dico a TCP lo faccio diventare matto, perché TCP ha tutta l'identificazione della connessione che è basata su <IP mittente, IP destinatario, porta mittente, porta destinatario>, e io non voglio abbattere quella connessione per metterne in piedi un'altra. Allora cosa faccio? Non glielo dico al mio TCP, e gli modifichiamo gli indirizzi utilizzati sotto il naso, e lo fa questo shim layer. A questo punto l'identificatore mio, e anche quello del destinatario, non cambiano per tutta la connessione, ma il locator mio e del destinatario possono cambiare. Di questo, i due TCP non se ne accorgono, perché quando arrivano i pacchetti prima di darli a TCP li cambio, e quando mi arrivano pacchetti che devono essere spediti li cambio anch'essi. Questo fa sì che la macchina possa scegliere, anche durante la connessione, un provider diverso da quello utilizzato

durante il setup della connessione. Questa è l'idea di shim layer. In IPv4 chi è che svolge questo ruolo? È BGP. BGP non modifica la struttura delle connessioni, anche se quando cambi percorso cambia il RTT, e questo può avere effetti sulla CWND, quindi si potrebbero avere delle fluttuazioni nella qualità del servizio, ma la connessione è inchiodata. Tutta questa cosa, non la potevo realizzare con un meccanismo di renumbering? Queste macchine il renumbering lo fanno facilmente, ma se faccio renumbering utilizzo un altro indirizzo IP, e la connessione me la perdo. Non soltanto: i tempi di renumbering non sono istantanei, tu comunque rimani isolato dal mondo per un po' di tempo. La connessione si apre usando l'ULID, e questo è l'indirizzo scelto da Source Address Selection. Se la connessione è persistente, si inizializza shim6. Questo evita l'overhead di shim6 per le connessioni di breve durata. Se il peer non risponde, o non supporta shim6 (questa cosa va negoziata) si continua senza. La negoziazione avviene attraverso Extension Headers. I due peer si scambiano “context tags” shim6 (una context tag identifica un “contesto” shim6, un contesto identifica gli ULID della connessione e i locator attualmente in uso). Se c'è un problema, i peer provano gli altri locator nel tentativo di ristabilire la comunicazione. Una volta trovata una coppia di locator funzionante, shim6 modifica tutti i pacchetti in transito:

- Cambia gli indirizzi dagli ULID ai locator in uso;
- Inserisce in un Extension Header la Context Tag in modo che il destinatario sappia a che connessione corrisponda il pacchetto.

Il protocollo di strato superiore continua a vedere una connessione tra gli ULID. Tutto ciò è simile a NAT, ma eseguito negli host. Però “Doesn't break end-to-end”, che vuol dire? Quando hanno progettato internet l'hanno basata sull'end-to-end principle: tutto sta sugli end system, all'interno della rete non c'è niente. NAT rompe questo principio, perché tu passi per una macchina che conserva lo stato.