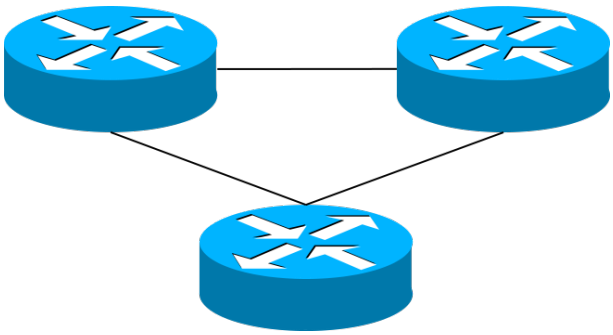


Appunti BGP

BGP e Cisco IOS

a cura di

Stefano Sasso

stefano(at)gnustile.net

Versione 0.9 - 21 Settembre 2009

Indice

Capitolo 1

Peer Groups

Le direttive di *peer group* consentono di mettere in comune tra diversi peer varie opzioni di configurazione. Ecco un esempio:

```
router bgp 100

  neighbor ibgp-peer peer-group
  neighbor ibgp-peer route-map outfilter out
  neighbor ibgp-peer remote-as 100

  neighbor 1.1.1.2 peer-group ibgp-peer
  neighbor 1.1.1.2 route-map ibgp2 in

  neighbor 1.1.1.19 peer-group ibgp-peer
  neighbor 1.1.1.19 route-map ibgp19 in
```

Capitolo 2

Network annunciata

Una *network* si annuncia con

```
router bgp 100
  network 44.12.0.0/16
```

Tuttavia un router BGP può anche non avere una *network* da annunciare direttamente.
Un esempio sono i router BGP di peering (ad esempio presso un NAP). Questo perchè se essi annunciassero la *network*, e il collegamento con il proprio backbone si interrompesse, i pacchetti per la rete continuerebbero ad arrivare al router di peering che non saprebbe come farli giungere a destinazione. Di norma, quindi, solo i core router di una rete annunciano la relativa *network*.

Capitolo 3

Features arcaiche da disabilitare

3.1 Auto Summarisation

Automatically summarises subprefixes to the classful network for prefixes redistributed into BGP.
es:

`61.10.8.0/22` → `61.0.0.0/8`

Must be turned off for any Internet connected site using BGP.

```
router bgp 109
  no auto-summary
```

3.2 Synchronisation

BGP will not advertise a route before all routers in the AS have learned it via an IGP.
Disable synchronisation if:

- AS doesn't pass traffic from one AS to another, or
- All transit routers in AS run BGP, or
- iBGP is used across backbone

```
router bgp 109
  no synchronization
```

Capitolo 4

Soft reconfiguration

Ogni qualvolta in un router si effettuano delle modifiche alle policy è necessario un clear completo delle sessioni BGP, e questo ha un notevole impatto sulla rete. È possibile evitare tutto ciò usando la *soft-reconfiguration*.

La *soft-reconfiguration* viene applicata per neighbor e permette di tenere in memoria, senza applicare, anche i prefissi che sono negati dai filtri.

```
neighbor ... soft-reconfiguration inbound
```

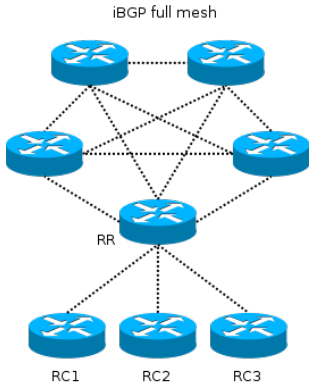
Quindi, dopo le modifiche alle policy di un router è sufficiente dare

```
clear ip bgp 1.1.1.1 soft [in | out]
```

Capitolo 5

Route Reflectors

Le sessioni iBGP devono essere realizzate come full-mesh. Tuttavia un setup del genere risulta da difficile qualora ci siano molti router che devono parlare iBGP.
Un'architettura del genere diventa estremamente più scalabile usando dei route reflectors (*RR*); questi router hanno il compito di replicare ai router client le rotte che apprendono, siano esse eBGP o iBGP.



(la linea tratteggiata corrisponde alla sessione iBGP, non alla connessione fisica)

RR:

```
router bgp 100
  neighbor ibgp-rrc peer-group
  neighbor ibgp-rrc route-reflector-client
  neighbor ibgp-rrc remote-as 100
  neighbor 1.1.2.1 peer-group ibgp-rrc
  neighbor 1.1.2.2 peer-group ibgp-rrc
  neighbor 1.1.2.3 peer-group ibgp-rrc
```

RC1-RC2-RC3:

```
router bgp 100
  neighbor 1.1.2.10 remote-as 10
```

Capitolo 6

Confederations

L'uso di *confederations* è un metodo per ridurre l'iBGP mesh interna ad un AS (come per i *route reflector*).

Il trucco consiste nel dividere un AS in AS multipli, e assegnare l'intero gruppo ad una *confederation*.

Ogni singolo AS si comporterà quindi come un AS a se stante: avrà una full mesh iBGP e connessioni agli altri AS interni alla stessa *confederation*.

- next-hop, metriche e local-preference restano invariate
- al resto del mondo una confederation appare comunque come un singolo unico AS
- gli AS interni ad una confederation utilizzeranno ASn privati
- l'*identifier* della confederation è il numero AS

```
router bgp 65531
  bgp confederation identifier 100
  bgp confederation peers 65532 65533

  neighbor 171.14.3.4 remote-as 65532
  neighbor 171.14.3.7 remote-as 65533
```

Capitolo 7

Uso delle interfacce Loopback in iBGP

È buona cosa assegnare degli indirizzi ip **/32** alle interfacce di loopback: questo per fare in modo che le sessioni iBGP avvengano tra queste interfacce (che sono sempre *up*).
Le informazioni di routing relative agli indirizzi delle loopback verranno quindi trasportate da un protocollo IGP (es: OSPF), mentre le informazioni di transito vengono trasportate da BGP.

```
interface Loopback 0
  ip address 11.22.33.44 255.255.255.255

router ospf 100
  redistribute static
  redistribute connected
  network 11.22.33.44 0.0.0.0 area 0
  network 13.33.22.55 0.0.0.255 area 0

router bgp 100
  neighbor 11.22.33.45 remote-as 100
  neighbor 11.22.33.45 update-source Loopback 0
  neighbor 11.22.33.45 next-hop-self
```

Capitolo 8

route-map/filter in vs route-map/filter out

ATTENZIONE: il verso in/out si riferisce alle rotte comunicate, non al percorso dei pacchetti di dati. Quindi, le rotte comunicate verso l'esterno - out - si riferiscono ai pacchetti in ingresso; viceversa, le rotte comunicate verso l'interno - in - si riferiscono ai pacchetti in uscita.

Capitolo 9

Local Preference

Determina il best path per l'outbound traffic. Percorsi con *local preference* maggiore vincono (default: 100).

Esempio: passare x un determinato router (120.5.1.1) per raggiungere una specifica destinazione (160.10.0.0/16):

```
router bgp 100
  neighbor 120.5.1.1 remote-as 200
  neighbor 120.5.1.1 route-map local-pref in

route-map local-pref permit 10
  match ip address prefix-list MATCH_DEST
  set local-preference 800

ip prefix-list MATCH_DEST permit 160.10.0.0/16
```

Capitolo 10

MED: multi-exit discriminator

Il *MED* è uno dei metodi per determinare il best path per l'inbound traffic. Come vedremo poi, l'altro metodo, generalmente più utilizzato, è l'*AS prepending*. Nel caso del *MED* vince il path con metrica minore.

Nel seguente esempio il link preferito è quello attraverso l'AS 300.

```
router bgp 100
  network 120.68.1.0/24
  neighbor 1.2.3.4 remote-as 200
  neighbor 1.2.3.4 route-map set-med out
  neighbor 1.2.9.3 remote-as 300

route-map set-med permit 10
  match ip address prefix-list MYNETWORK
  set metric 1000

ip prefix-list MYNETWORK permit 120.68.1.0/24
```

Capitolo 11

AS path prepending

L'*AS path prepending* serve a determinare il best path per l'inbound traffic. Sapendo che il percorso utilizzato è sempre quello che attraversa il minor numero di AS, la tecnica consiste nell'iniettare più volte nell'AS path il proprio AS per i percorsi da discriminare. Ipotizziamo che la nostra rete (AS100) sia raggiungibile, da parte dell'AS300, tramite questi percorsi:

```
a) 29 44 109 1384 (100)
b) 18 24 332 2287 332 176 (100)
```

Ipotizziamo ora di voler utilizzare il percorso (b) come preferito (in quanto ha una maggiore ampiezza di banda), dovremo quindi allungare il percorso (a) facendolo diventare (più lungo di b):

```
a) 29 44 109 1384 100 100 100 (100)
```

ecco come:

```
router bgp 100
  neighbor 1.3.5.6 remote-as 1384
  neighbor 1.3.5.6 route-map as-prepend out

route-map as-prepend permit 10
  set as-path prepend 100 100 100
```

Capitolo 12

prefix-list

I *prefix-list* si riferiscono a (classi di) indirizzi ip e possono venire usati da *route-map* o direttamente impiegati come regole di filtraggio delle rotte apprese. Ecco alcuni esempi di *prefix-list* con spiegazione:

```
ip prefix-list EXAMPLE deny 0.0.0.0/0
```

Nega rotta di default.

```
ip prefix-list EXAMPLE permit 35.0.0.0/8
```

Accetta 35.0.0.0/8.

```
ip prefix-list EXAMPLE deny 172.16.0.0/12
```

Nega 172.16.0.0/12.

```
ip prefix-list EXAMPLE1 permit 192.0.0.0/8 le 24
ip prefix-list EXAMPLE2 deny 192.0.0.0/8 ge 25
```

Le due regole hanno lo stesso effetto: accettano tutti i prefissi compresi tra /8 e /24 di quella rete, ma non i /25, /26, ...

```
ip prefix-list EXAMPLE permit 193.0.0.0/8 ge 12 le 20
```

Accetta tutti i prefissi della rete 193/8 compresi tra /12 e /20.

```
ip prefix-list EXAMPLE accept 0.0.0.0/0 le 32
```

Accetta tutto.

Ricordiamoci che *prefix-list* comporta un match preciso: ad esempio

```
ip prefix-list EXAMPLE permit 193.0.0.0/8
```

accetta 193.0.0.0/8, ma non 193.0.0.0/16 (anche se appartiene alla stessa sottorete); mentre

```
ip prefix-list EXAMPLE permit 193.0.0.0/8 le 16
```

accetta sia 193.0.0.0/8 che 193.0.0.0/16 (ma anche 193.0.0.0/12, ma non 193.0.0.0/24).

Ecco come applicarli direttamente come regole di filtraggio per l'apprendimento di rotte:

```
router bgp 100
  network 105.7.0.0/16
  neighbor 102.10.1.1 remote-as 600
  neighbor 102.10.1.1 prefix-list PEER_IN in
  neighbor 102.10.1.1 prefix-list PEER_OUT out

ip prefix-list PEER_OUT permit 105.7.0.0/16
ip prefix-list PEER_OUT deny 0.0.0.0/0 le 32
ip prefix-list PEER_IN permit 0.0.0.0/0
```

Capitolo 13

AS-path filter list

Serve per filtrare le rotte apprese basandosi sull'AS-path.

```
router bgp 100
  neighbor 102.10.1.1 remote-as 600
  neighbor 102.10.1.1 filter-list 5 out
  neighbor 102.10.1.1 filter-list 6 in

ip as-path access-list 5 permit _200$
ip as-path access-list 5 permit _100$
ip as-path access-list 6 permit _150_
```

Vengono utilizzate le classiche regole regex:

- ^ : inizio linea
- $: fine linea
- _ : spazi, inizio o fine

alcuni esempi:

- .* : tutto
- ^$: vuoto (rotte locali)
- _1800$: rotte originate da AS1800
- ^1800_ : rotte ricevute da AS1800
- _1800_ : rotte via 1800
- _790_1800_ : rotte via 1800 e 790

Un altro esempio, per indicare AS300 e suoi neighbor, e neighbor di neighbor, ...

```
ip as-path access-list 17 permit _300$
ip as-path access-list 17 permit _300_[0-9]+$
ip as-path access-list 17 permit _300_[0-9]+_[0-9]+$
```

Capitolo 14

route-map

È una specie di if/else; attenzione: se non soddisfatto if droppa di default, per evitare questo comportamento aggiungere *permit* finale.

```
route-map sample permit 10
  match ip address prefix-list list-one
  set local-preference 120
route-map sample permit 20
  match ip address prefix-list list-two
  set local-preference 80
route-map sample permit 30
  match as-path 5
  set-local-preference 150
route-map sample permit 40
```

Capitolo 15

private AS numbers

I numeri AS privati vanno da **64512** a **65534**, e devono essere rimossi agli EDGE!

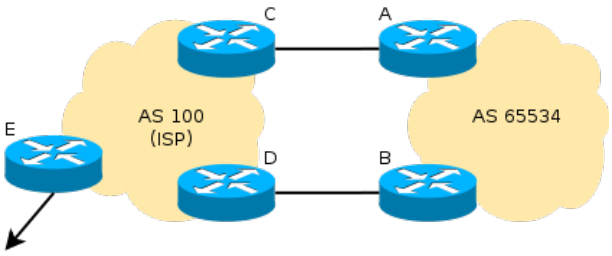
```
router bgp 100
  neighbor 195.12.3.33 remote-as 65530

  neighbor 133.11.113.32 remote-as 198
  neighbor 133.11.113.32 remove-private-AS
```

Capitolo 16

Multihoming scenario

16.1 Two links to the same ISP - simple loadsharing, no redundancy

- uso di ASn privati
- ISP aggrega gli indirizzi - ipotizziamo di avere 121.10.0.0/19
- la /19 viene splittata in due /20, ognuna annunciata in un link (basic inbound loadsharing)



router A:

```
router bgp 65534
  network 121.10.0.0 mask 255.255.240.0
  network 121.10.16.0 mask 255.255.240.0
  neighbor 122.102.10.2 remote-as 100
  neighbor 122.102.10.2 prefix-list routerC out
  neighbor 122.102.10.2 prefix-list default in

ip prefix-list default permit 0.0.0.0/0
ip prefix-list routerC permit 121.10.0.0/20
```

router B:
configurazione sulla falsariga di **A**, cambia solo la prefix-list e l'ip del neighbor

```
...
  neighbor 122.102.11.2 remote-as 100
```

```
  neighbor 122.102.11.2 prefix-list routerD out
...
ip prefix-list routerC permit 121.10.16.0/20
```

router C:

```
router bgp 100
  neighbor 122.102.10.1 remote-as 65534
  neighbor 122.102.10.1 default-originate
  neighbor 122.102.10.1 prefix-list Customer in
  neighbor 122.102.10.1 prefix-list default out

ip prefix-list default permit 0.0.0.0/0
ip prefix-list Customer permit 121.10.0.0/20
```

router D:
configurazione sulla falsariga di **C**, cambia solo la prefix-list e l'ip del neighbor

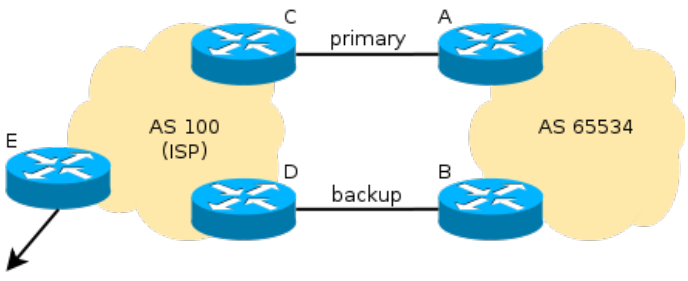
```
...
  neighbor 122.102.11.1 remote-as 65534
...
ip prefix-list Customer permit 121.10.16.0/20
```

router E:

```
router bgp 100
  network 121.10.0.0 mask 255.255.0.0
  ...
  neighbor ebgp-peers peer-group
  neighbor ebgp-peers remove-private-AS
  ...
```

16.2 Two links to the same ISP - one primary, one backup only

- uso di ASn privati
- ISP aggrega gli indirizzi - ipotizziamo di avere 121.10.0.0/19
- annuncia /19 su ogni link, nel backup con una metrica maggiore
- nel router che detiene il collegamento di backup riduce la local-preference per la default route



router A:

```
router bgp 65534
  network 121.10.0.0 mask 255.255.224.0
  neighbor 122.102.10.2 remote-as 100
  neighbor 122.102.10.2 prefix-list aggregate out
  neighbor 122.102.10.2 prefix-list default in

ip prefix-list default permit 0.0.0.0/0
ip prefix-list aggregate permit 121.10.0.0/19
```

router B:
configurazione sulla falsariga di **A**, modificando l'ip del neighbor e aggiungendo

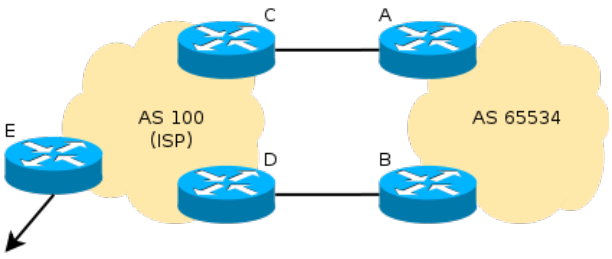
```
...
  neighbor 122.102.11.2 route-map routerD-out out
  neighbor 122.102.11.2 route-map routerD-in in
...
route-map routerD-out permit 10
  match ip address prefix-list aggregate
  set metric 10
route-map routerD-out permit 20

route-map routerD-in permit 10
  set local-preference 90
```

router C-D-E: configurazione sulla falsariga di **C-D-E** dell'esempio precedente con

```
ip prefix-list Customer permit 121.10.0.0/19
```

16.3 Two links to the same ISP - redundancy and loadsharing

- uso di ASn privati
- ISP aggrega gli indirizzi - ipotizziamo di avere 121.10.0.0/19
- annuncia /19 su ogni link, in più divide la /19 in due /20 e ne annuncia uno per link
- è possibile fare as-prepending solo su rotte /19, in modo da dare priorità alle /20 (non affrontato in questo esempio)



router A (sulla falsariga degli esempi precedenti):

```
router bgp 65534
  network 121.10.0.0 mask 255.255.224.0
  network 121.10.0.0 mask 255.255.240.0
  neighbor 122.102.10.2 remote-as 100
  neighbor 122.102.10.2 prefix-list routerC out
  neighbor 122.102.10.2 prefix-list default in

ip prefix-list default permit 0.0.0.0/0
ip prefix-list routerC permit 121.10.0.0/20
ip prefix-list routerC permit 121.10.0.0/19
```

router B:
configurazione sulla falsariga di **A**, modificando l'ip del neighbor e modificando la /20 annunciata

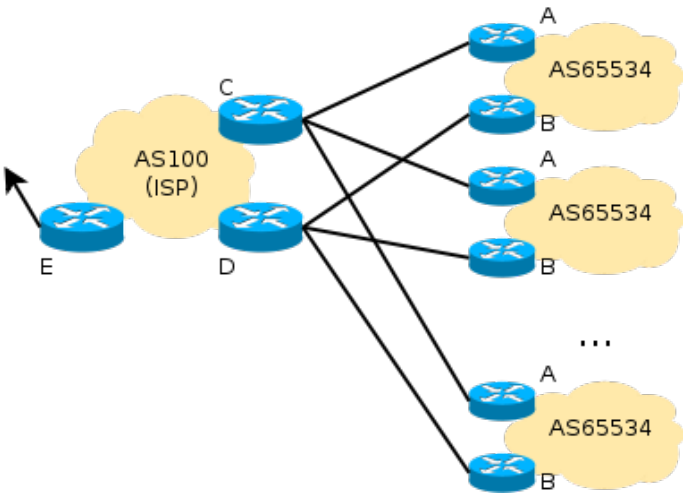
```
...
  network 121.10.16.0 mask 255.255.240.0
...
  neighbor 122.102.11.2 prefix-list routerD out
...
ip prefix-list routerD permit 121.10.16.0/20
ip prefix-list routerD permit 121.10.0.0/19
```

router C-D: configurazione sulla falsariga di **C-D** dell'esempio precedente con

```
ip prefix-list Customer permit 121.10.0.0/19 le 20
```

Il loadsharing viene praticamente eseguito solo lato customer, non dall'ISP. Vedremo in futuro una configurazione più avanzata.

16.4 Punto di vista ISP: multiple dualhomed customers (with private ASn)



Lato customer è identico alle situazioni precedenti.

- si può usare lo stesso ASn privato per tutti i customer
- ogni customer riceve solo default route

router C:

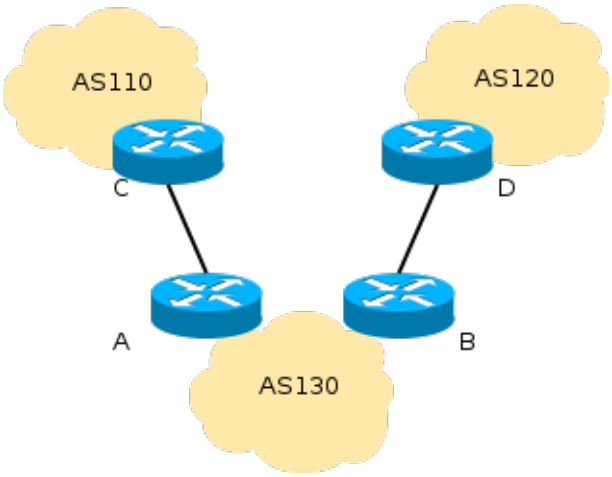
```
router bgp 100
  neighbor bgp-customers peer-group
  neighbor bgp-customers remote-as 65534
  neighbor bgp-customers default-originate
  neighbor bgp-customers prefix-list default out

  neighbor 123.102.10.1 peer-group bgp-customers
  neighbor 123.102.10.1 prefix-list Customer1 in

  neighbor 123.102.11.1 peer-group bgp-customers
  neighbor 123.102.11.1 prefix-list Customer2 in

  neighbor 123.102.12.1 peer-group bgp-customers
  neighbor 123.102.12.1 prefix-list Customer3 in
...
ip prefix-list Customer1 permit 121.10.0.0/19 le 20
ip prefix-list Customer2 permit 121.10.32.0/19 le 20
ip prefix-list Customer3 permit 121.10.64.0/19 le 20
```

router D: sulla falsariga di **C**

router E:

- annuncia tutti i customer; se i customers' prefixes appartengono all'isp address block si annuncia solo l'aggregata

16.5 Two links to different ISPs



- public ASn

- in linea di massima le configurazioni sono molto simili ai precedenti

Nel caso un router sia backup-only (**B**) usiamo l'*AS-prepend*:
router B:

```
router bgp 130
  network 121.10.0.0 mask 255.255.224.0
  neighbor 120.1.5.1 remote-as 120
  neighbor 120.1.5.1 prefix-list aggregate out
  neighbor 120.1.5.1 route-map routerD-out
  neighbor 120.1.5.1 prefix-list default in
  neighbor 120.1.5.1 route-map routerD-in
...
route-map routerD-out permit 10
  set as-path prepend 130 130 130
route-map routerD-in permit 10
  set local-preference 80
```

Invece, se vogliamo avere loadsharing e rindondanza:

- annuncia /19 su ogni link
- nel primo link annuncio normale
- nel secondo link annuncio di /19 con longer AS-path e un /20 normale
- variare la lunghezza del sottoprefisso e dell'AS-path fino a raggiungere la 'perfezione'

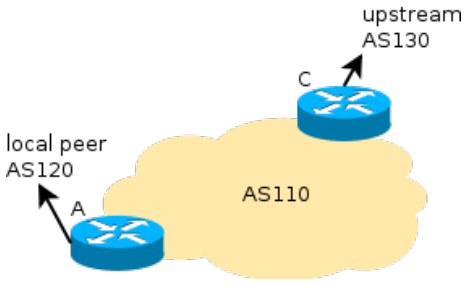
router B:

```
...
  neighbor 1.33.43.2 prefix-list subblocks out
  neighbor 1.33.43.2 route-map routerD-out
...
ip prefix-list subblocks permit 121.10.0.0/19 le 20
ip prefix-list aggregate permit 121.10.0.0/19
...
route-map routerD-out permit 10
  match ip address prefix-list aggregate
  set as-path prepend 130 130
route-map routerD-out permit 20
...
```

Capitolo 17

Service Provider Multihoming

17.1 one upstream, one local peer



- upstream traffic to internet
- local traffic stays local
- annuncia /19 su ogni link
- accetta default solo da upstream (0.0.0.0 o network usabile come default)
- in questo esempio viene dichiarata una *network* anche dal router di peering, sarebbe meglio farlo solo nei core router

Router A (verso local peer):

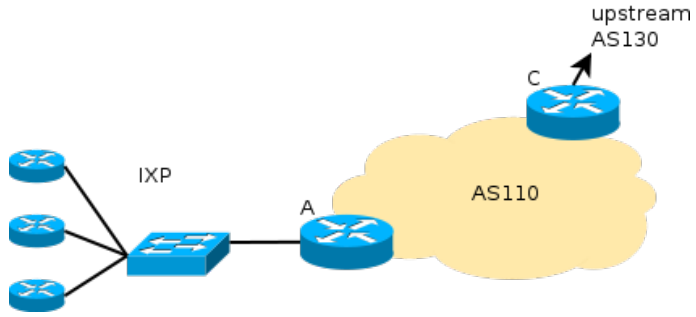
```
router bgp 110
  network 121.10.0.0 mask 255.255.224.0
  neighbor 122.102.10.2 remote-as 120
  neighbor 122.102.10.2 prefix-list my-block out
  neighbor 122.102.10.2 prefix-list AS120-peer in

ip prefix-list AS120-peer permit 122.5.16.0/19
ip prefix-list AS120-peer permit 121.240.0.0/20
ip prefix-list my-block permit 121.10.0.0/19
```

in alternativa alla *prefix-list* si poteva usare:

```
...
  neighbor 122.102.10.2 filter-list 10 in
...
ip as-path access-list 10 permit ^(120_)+$
```

router C (verso upstream): configurazione standard.

17.2 one upstream, local IXP



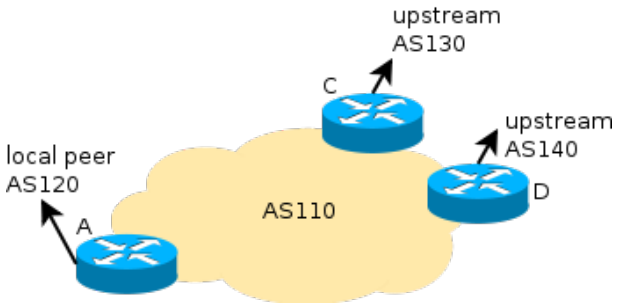
- accetta tutto dai peer IXP
- accetta default da upstream

router A (@ IXP):

```
router bgp 100
  neighbor ixp-peer peer-group
  neighbor ixp-peer prefix-list my-block out
  neighbor ixp-peer remove-private-AS

  neighbor 140.5.10.2 remote-as 100
  neighbor 140.5.10.2 peer-group ixp-peer
  neighbor 140.5.10.2 prefix-list peer100 in
  ...

ip prefix-list peer100 permit 123.0.0.0/19
...
```

NOTA: come già detto, il router **A** non genera l'aggregate (*network*) per AS110 perchè così facendo, se il router venisse disconnesso dalla backbone l'aggregata non verrebbe più annunciata agli IXP peers. In caso contrario i pacchetti provenienti via IXP si perderebbero su **A**.

17.3 two upstreams, one local peer



- multihoming 'standard' verso gli upstream
- traffico locale per il peer

ma... con configurazione di multihoming come quelle viste in precedenza per i pacchetti in uscita verrebbe usato un solo upstream (cosa che non vogliamo!). Per bilanciare il carico anche in uscita abbiamo più metodi:

- accettare una full routing table da entrambi gli upstream => expensive and unnecessary!
- accetta default da un upstream, e alcune rotte selezionate dall'altro => ideale!

configurazione con full routing table

Con la seguente configurazione **C** e **D** hanno una full routing table (e poi si agisce sulla *local-preference*).

router C (accetta tutto, aumenta preferenza per alcune rotte):

```
router bgp 110
  network ...
  neighbor ... remote-as 130
  neighbor ... prefix-list rfc1918-deny in
  neighbor ... prefix-list my-block out
  neighbor ... route-map AS130-loadshare in
...
ip as-path access-list 10 permit ^(130_)+$
ip as-path access-list 10 permit ^(130_)+_[0-9]+$

route-map AS130-loadshare permit 10
  match ip as-path 10
  set local-preference 120
route-map AS130-loadshare permit 20
  set local-preference 80
```

La prima *as-path access-list* seleziona le rotte originate dall'AS130, mentre la seconda seleziona le rotte originate dai neighbors di AS130 (i suoi peers o customers).

router D: accetta tutto tranne rfc1918.

configurazione con rotte parziali

Per ottenere delle rotte parziali bisogna comunque richiedere all'upstream una full-route e poi si filtra in ingresso.

router C (filtra le rotte ricevute):

```
...
  neighbor ... filter-list 10 in
  neighbor ... route-map tag-default-low in

ip prefix-list default permit 0.0.0.0/0

ip as-path access-list 10 permit ^(130_)+$
ip as-path access-list 10 permit ^(130_)+_[0-9]+$

route-map tag-default-low permit 10
  match ip address prefix-list default
  set local-preference 80
route-map tag-default-low permit 20
```

router D (accetta solo default, lasciando la *local-preference* a 100):

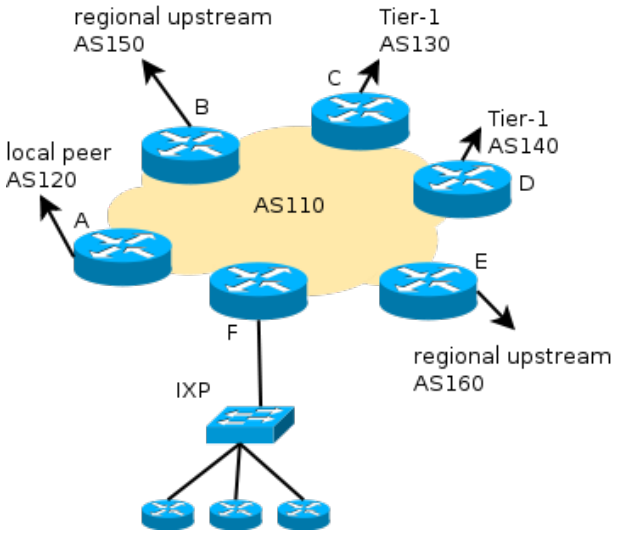
```
...
  neighbor ... prefix-list default in
...
```

NOTA: se gli upstream non generano una default, non generarla su iBGP, ma usare un altro IGP per propagarla.

Ad esempio con OSPF (ricordo che la metrica minore è quella preferenziale):

```
C# router ospf 110
C#   default-information originate metric 30

D# router ospf 110
D#   default-information originate metric 10
```

17.4 two tier-1, two regional upstreams, local peers



- questo esempio mette insieme tutti i precedenti
- genera aggregate (*network*) solo sui core router
- annuncia /19 su ogni link
- accetta parziali/default o full routing table dai Tier-1
- accetta tutte le rotte locali dai peers (sia peer locali che peer@IXP)
- accetta rotte parziali dagli upstream regionali

router A (local peer):

- accetta tutte le rotte locali
- usa filtri basati su *prefix-list* o AS-path
- *local-preference* > 100

router F (IXP peering): come **A**

router B (regional upstream): Fornisce transito per internet, ma di norma con AS-path più lunghi rispetto a un Tier-1.

- accetta le rotte regionali (AS-path filter simile a `^150_[0-9]+$`)
- chiedere rotta di default e settarci una *local-preference* a 60 (in generale minore della pref. della default sui Tier-1).

router E (regional upstream): come **B**. Eventualmente, se si vuole preferire all'altro upstream regionale, settare *local-preference* a 70 (ma minore della pref. della default sui Tier-1).

router C e D (Tier-1 upstream): come già visti (full routing table o parziale). Giocare sulla *local-preference* per la rotta di default.

Inoltre, per un loadsharing dei pacchetti in ingresso, una aggregata totale deve essere annunciata a tutti (sfruttando anche l'AS prepending); possono inoltre essere annunciate rotte più specifiche per sottreti dell'aggregata.

17.5 NOTE generali

Verificare che i router di connessione con peer privati (o presso IXP) non trasportino la rotta di default! Basta semplicemente:

```
ip route 0.0.0.0 0.0.0.0 null0
```

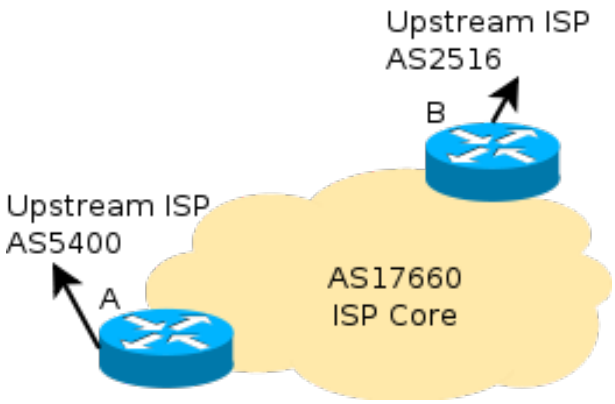
Capitolo 18

Ruolo di iBGP e IGP in reti multihomed

- IGP produce default se non è fornita via eBGP.
- IGP produce le rotte per tutte le /32 loopback.
- IGP produce le rotte per le reti connesse alla backbone ma che non sono di transito (es: rete dei server, ...).
- i router di peering non accettano nessuna default, e forzano default su blackhole (null0).
- in iBGP vengono propagate le rotte parziali (eventualmente filtrando una full table), in modo che vengano direttamente usate. Per il resto si usa la default.
- per evitare loop fare una route statica a null0 per l'aggregato completo.

Capitolo 19

Case study (esempio) di un multihomed ISP



ISP deve fare multihoming:

- Upstream provider AS5400
- Upstream provider AS2516
- /19 assegnata
- AS17660 assegnato
- Connessione equivalente per ogni upstream (stessa banda)

L'ISP vuole:

- routing 'simmetrico' e bilanciamento 'ideale' tra i 2 link
- non ha sufficiente RAM per utilizzare una full routing table sui border router (2 cisco 2600 con 64Mb)

Soluzione:

- Accettare default route da un upstream
- Accettare prefissi parziali dall'altro upstream

Inbound loadsharing: utilizzando looking glasses da varie sorgenti sono stati controllati i percorsi via AS5400 e AS2516: AS2516 era di un HOP più vicino; per questo è stato fatto AS prepending (1) sul link con AS2516.

Outbound loadsharing: in uno dei due router accettate rotte parziali per l'AS, i suoi neighbor e i neighbor dei neighbor. Tuttavia questo link veniva comunque troppo utilizzato, quindi sono state rimosse le rotte transitanti per i primi N (=5 in questo caso) AS.

Configurazione BGP:
router A:

```
router bgp 17660
  no synchronization
  no bgp fast-external-fallover
  bgp log-neighbor-changes
  bgp deterministic-med
  neighbor 166.49.165.13 remote-as 5400
  neighbor 166.49.165.13 description eBGP multihop to AS5400
  neighbor 166.49.165.13 ebgp-multihop 5
  neighbor 166.49.165.13 update-source Loopback0
  neighbor 166.49.165.13 prefix-list in-filter in
  neighbor 166.49.165.13 prefix-list out-filter out
  neighbor 166.49.165.13 filter-list 1 in
  neighbor 166.49.165.13 filter-list 3 out

  prefix-list in-filter  deny rfc1918etc in
  prefix-list out-filter permit 202.144.128.0/19

  ip as-path access-list 1 deny _1_
  ip as-path access-list 1 deny _7018_
  ip as-path access-list 1 deny _1239_
  ip as-path access-list 1 deny _7046_
  ip as-path access-list 1 permit _5400$
  ip as-path access-list 1 permit _5400_[0-9]+$
  ip as-path access-list 1 permit _5400_[0-9]+_[0-9]+$
  ip as-path access-list 1 deny .*
  ip as-path access-list 3 permit ^$
```

router B:

```
router bgp 17660
  no synchronization
  no auto-summary
  no bgp fast-external-fallover
  bgp log-neighbor-changes
  bgp deterministic-med
  neighbor 210.132.92.165 remote-as 2516
  neighbor 210.132.92.165 description eBGP peering
  neighbor 210.132.92.165 soft-reconfiguration inbound
  neighbor 210.132.92.165 prefix-list default-route in
  neighbor 210.132.92.165 prefix-list out-filter out
```

```
  neighbor 210.132.92.165 route-map as2516-out out
  neighbor 210.132.92.165 maximum-prefix 100
  neighbor 210.132.92.165 filter-list 2 in
  neighbor 210.132.92.165 filter-list 3 out

prefix-list default-route permit 0.0.0.0/0
prefix-list out-filter permit 202.144.128.0/19

ip as-path access-list 2 permit _2516$
ip as-path access-list 2 deny .*
ip as-path access-list 3 permit ^$

route-map as2516-out permit 10
  set as-path prepend 17660
```

Capitolo 20

BGP communities

Le *communities* sono dei 'tag' che possono venir propagati ai neighbor che, a seconda dei 'tag' ricevuti potranno intraprendere determinate politiche di routing (in maniera, praticamente, automatica).

L'invio di *communities* ad un peer deve essere esplicitamente indicato:

```
router bgp 100
  neighbor 1.2.3.4 remote-as 200
  neighbor 1.2.3.4 send-community
```

Le *communities* vengono assegnate da delle *route-map*:

```
route-map xyz permit 10
  match ip address ...
  set community 100:300
route-map xyz permt 20
```

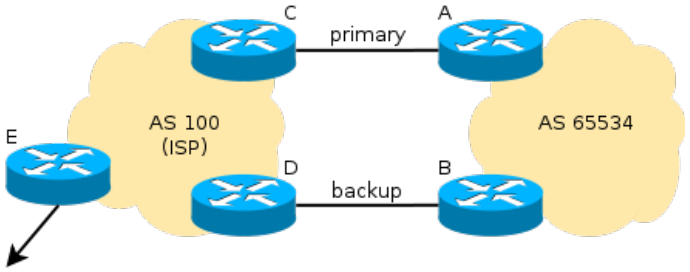
e venire analizzate sempre da parte di *route-map*:

```
ip community-list 1 permit 100:300

route-map xyzk permit 10
  match community 1
  set local-preference 130
route-map xyzk permit 20
```

Ci sono alcune *communities* che hanno dei significati universali:

- AS:100 - setta local-preference a 100 su rotta appresa da me (preferred route)
- AS:90 - setta local-preference a 90 su rotta appresa da me (backup route if dualhomed on AS)
- AS:80 - setta local-preference a 80 su rotta appresa da me (main link is to another ISP with same AS path length)
- AS:70 - setta local-preference a 70 su rotta appresa da me (main link is to another ISP)
- AS:3 (AS:2,AS:1) - effettua as-prepending di 3 (2,1) volte negli annunci dei peer di AS (AS è il backup transit as) - AS_AS_AS (AS_AS/AS)

Vediamo come l'uso di *communities* può semplificare una soluzione di dualhoming vista in precedenza: (two links to the same isp, primary and backup)



router A e C: configurazioni standard

router B:

```
router bgp 65534
  neighbor ... remote-as 100
  neighbor ... send-community
  neighbor ... prefix-list aggregate out
  neighbor ... prefix-list default in
  neighbor ... route-map routerD-out out
  neighbor ... route-map routerD-in in

route-map routerD-in permit 10
  set local-preference 90

route-map routerD-out permit 10
  match ip address prefix-list aggregate
  set community 100:90
route-map routerD-out permit 20
```

router D:

```
router bgp 100
  ...
  neighbor ... route-map bgp-customers in
  ...

ip community-list 90 permit 100:90

route-map bgp-customers permit 30
  match community 90
  set local-preference 90
route-map bgp-customers permit 40
  set local-preference 100
```