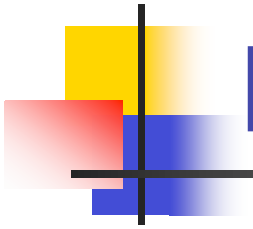
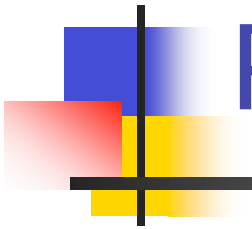
Implementando BGP

José A. Domínguez
<jad@ns.uoregon.edu>
University of Oregon



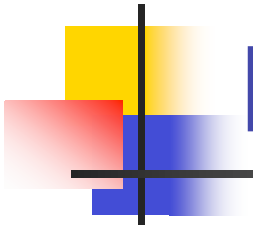
Programa

- Uso de atributos de BGP
- Implementando IBGP
- Implementando EBGP
- Enfasis en Estabilidad, Escalabilidad y Ejemplos de Configuraciones



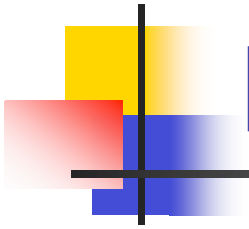
Repaso de BGP

Por qué utilizarlo?



Resultado Deseado?

- Implementación de políticas de enrutamiento que sean:
  - *Escalable*
  - *Estable*
  - *Simple (o eso esperamos!)*



Más Detalles...

- Necesitas escalar su IGP
- Eres un cliente con dos conexiones a ISPs
- Necesitas transitar todas las rutas en Internet
- Necesitas implementar una politica de enrutamiento, o expandir las políticas de QoS

Actualizaciones de BGP

Retiros

Atributos

Prefijos

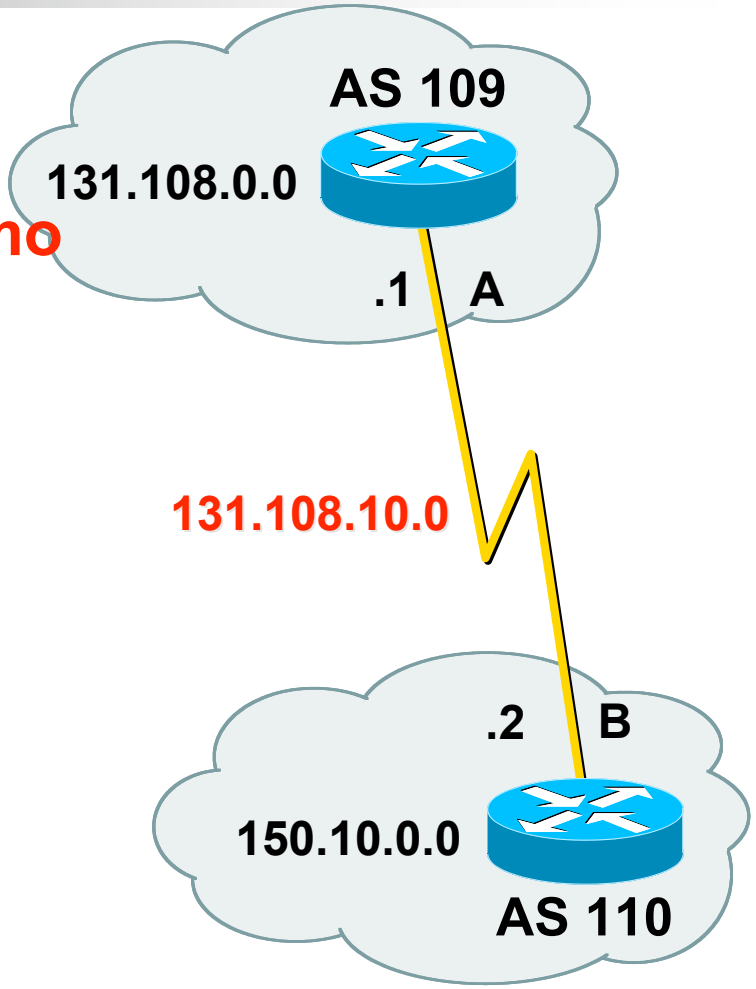
(NLRI - Network-Layer Reachability Information)

Atributos de BGP Utilizados para Definir la Política de Enrutamiento

- 1: ORIGIN
- 2: AS-PATH
- 3: NEXT-HOP
- 4: MED
- 5: LOCAL_PREF
- 6: ATOMIC_AGGREGATE
- 7: AGGREGATOR
- 8: COMMUNITY
- 9: ORIGINATOR_ID
- 10: CLUSTER_LIST
- 14: MP_REACH_NLRI
- 15: MP_UNREACH_NLRI

BGP Externo (eBGP)

- **Entre router en AS diferentes**
- **Usualmente con conexióm directa**
- **Con next-hop apuntando a si mismo**



- Router B

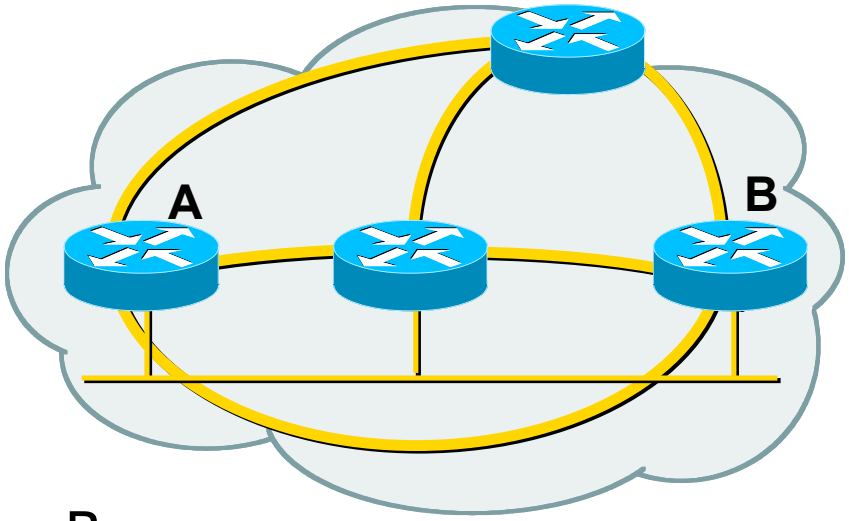
  router bgp 110
  neighbor 131.108.10.1 remote-as 109

- Router A

  router bgp 109
  neighbor 131.108.10.2 remote-as 110

BGP Interno

- **Vecinos en el mismo AS**
- **No se modifica el Next-hop**
- **No necesariamente con conexión directa**
- **No anuncia otras rutas aprendidas por iBGP**



- Router B:

  router bgp 109

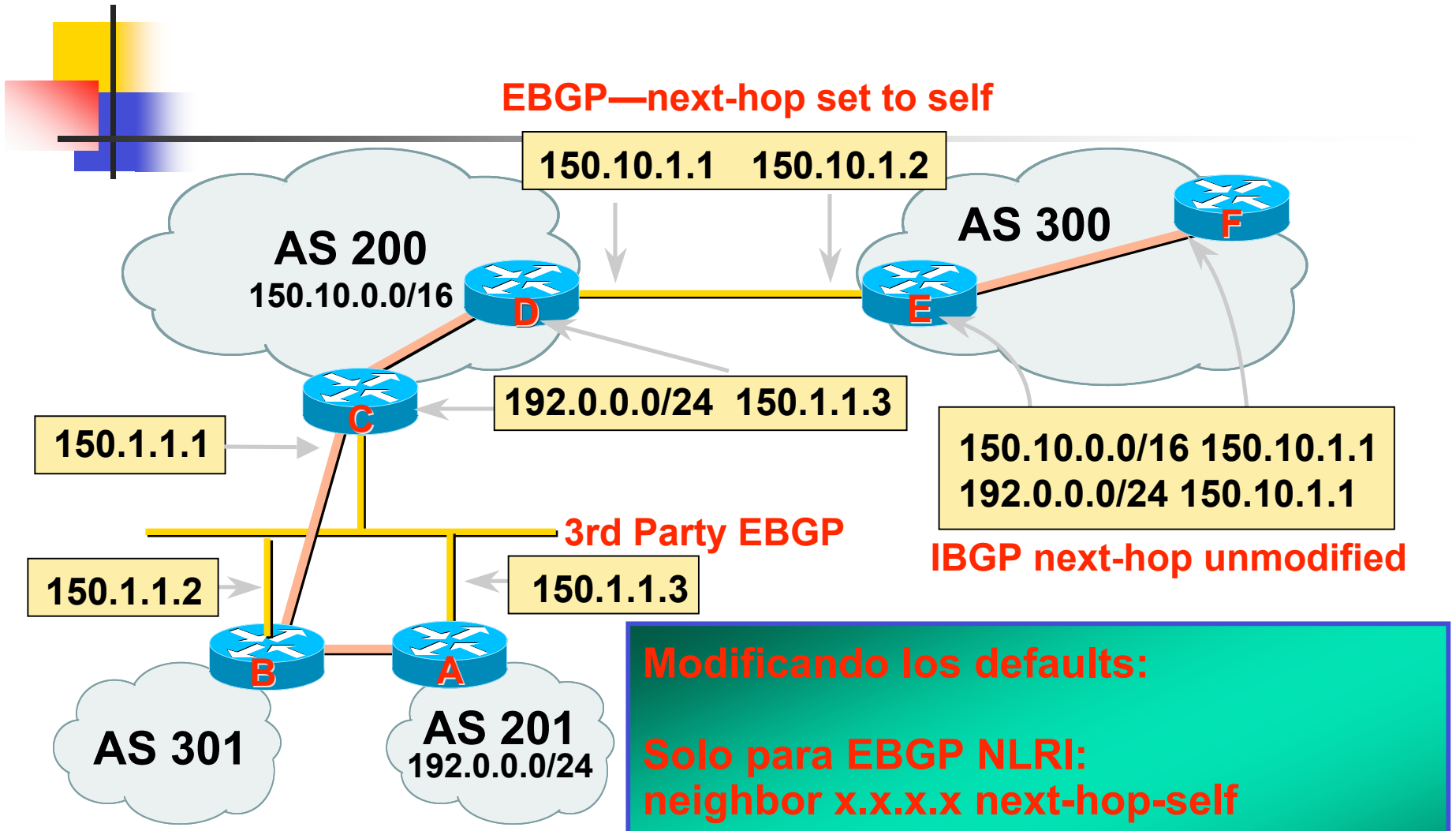
  neighbor 131.108.30.2 remote-as 109

- Router A:

  router bgp 109

  neighbor 131.108.20.1 remote-as 109

Atributos de BGP: NEXT_HOP



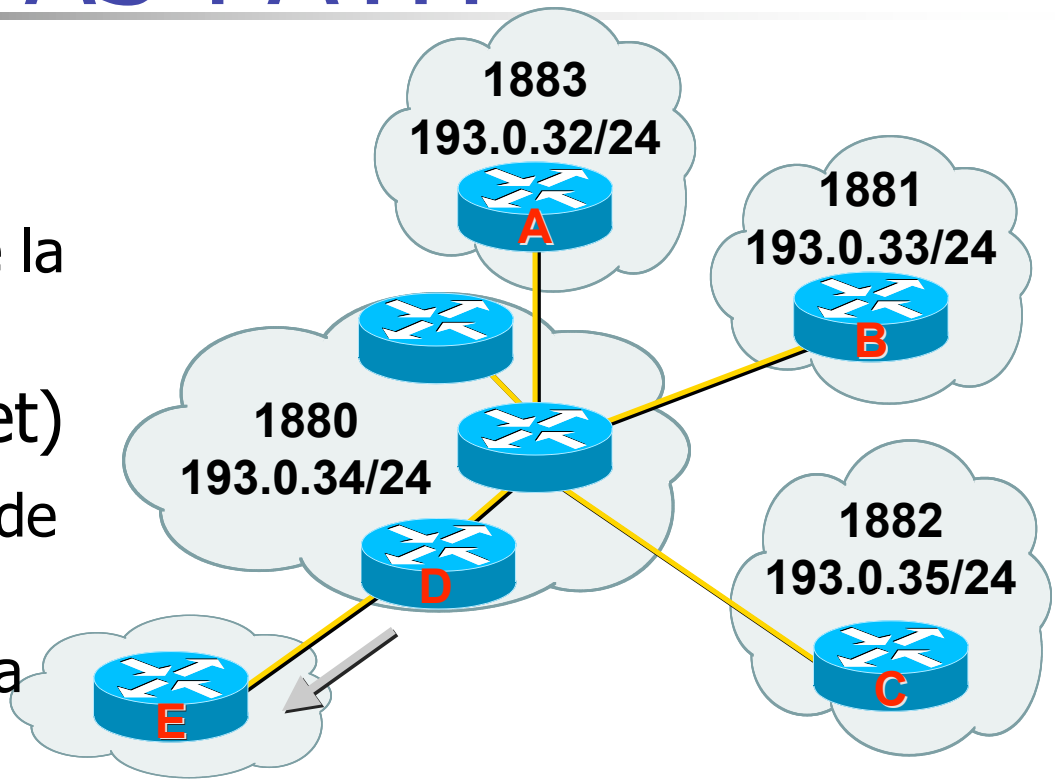
Modificando los defaults:

Solo para EBGP NLRI:
neighbor x.x.x.x next-hop-self

Modificando con route-map:
set ip next-hop { A.B.C.D | peeraddress}

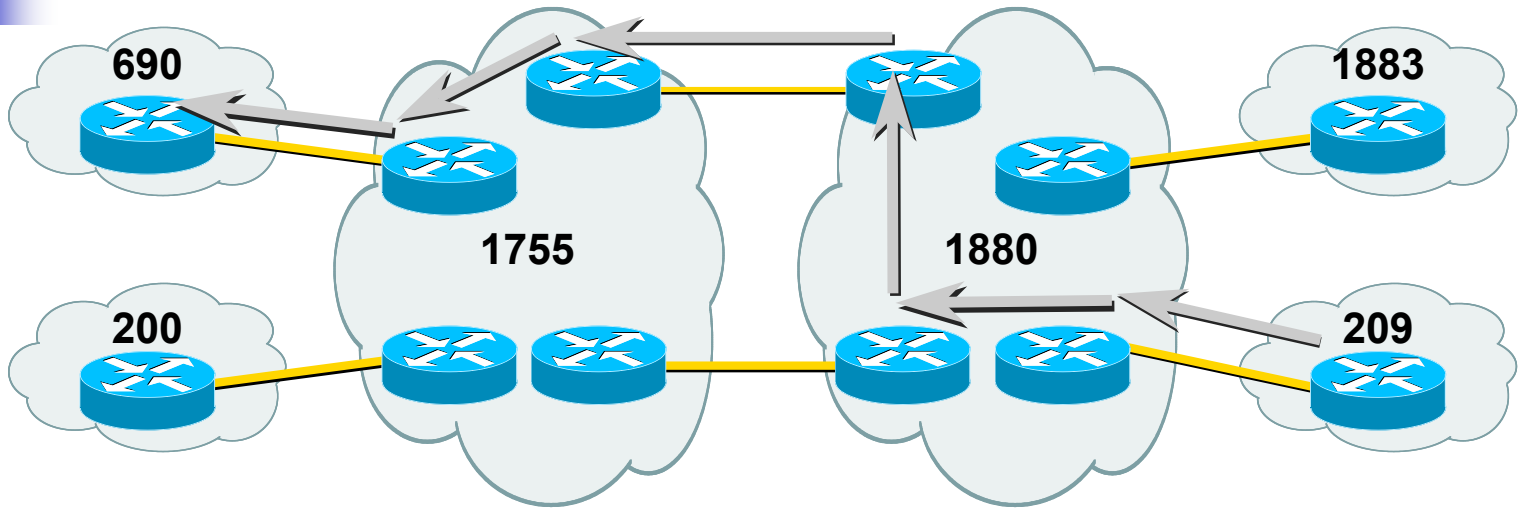
Problema: Detección de Loop, Políticas
Solución: AS-PATH

- Secuencia de ASs
  - Lista de AS por los que la ruta ha pasado
- Conjunto de AS (AS Set)
  - Sumariza la secuencia de ASs
  - El order de la secuencia se pierde
- Prefijo con route-map: *set as-path*



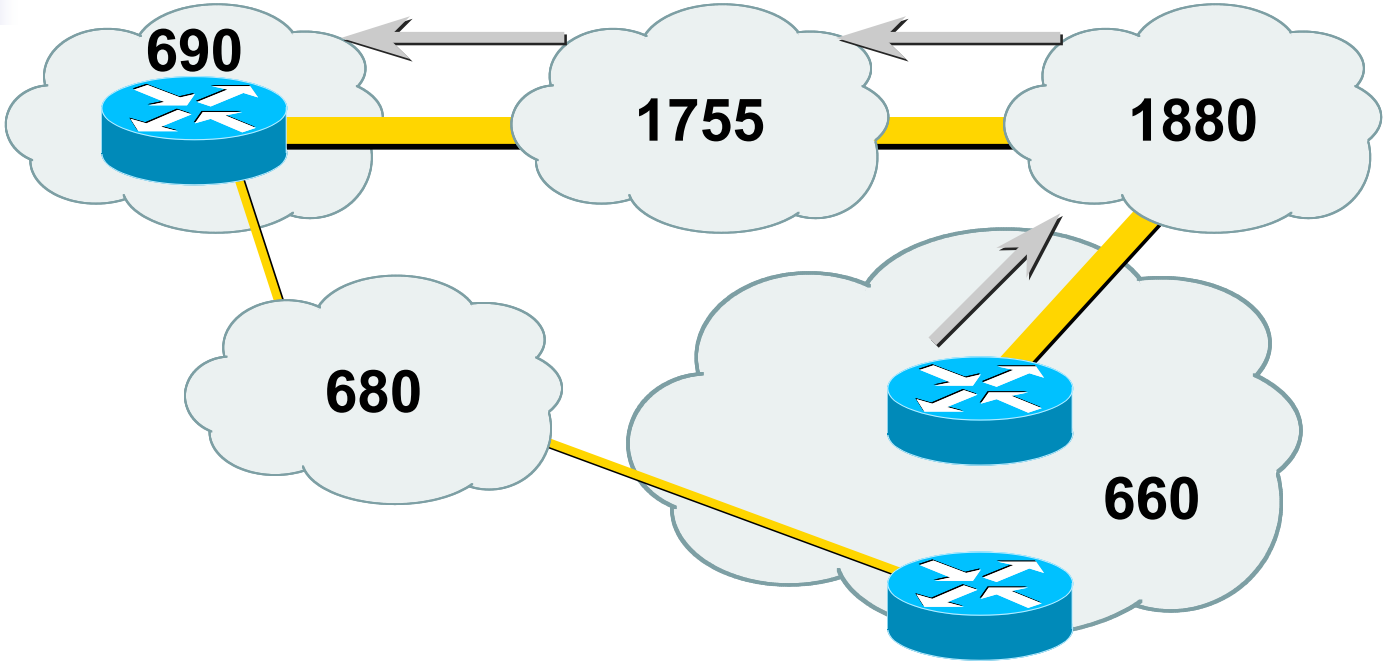
A: 193.0.33/24 1880 1881
B: 193.0.34/24 1880
C: 193.0.32/24 1880 1883
E: 193.0.32/22 1880 {1881, 1882,1883}

Problema: Indicar mejor camino a
Solución: MED



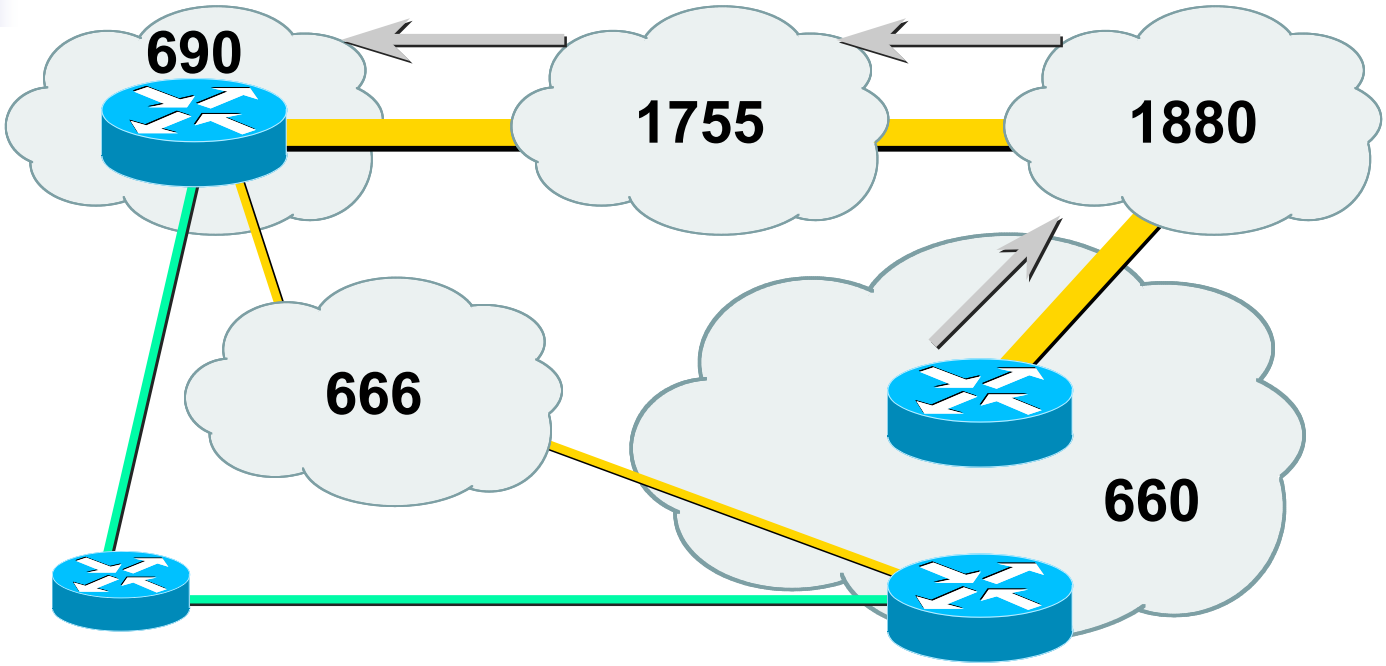
- Informa sobre la preferencia de punto de entrada
- Se compara si los caminos son del mismo AS
  - A no ser que se us "bgp always-compare-med"
- Es un atributo no-transitivo
- route-map: *set metric*
  set metric-type internal

Problema: Sobrepasando As-path/MED? Solución: LOCAL PREFERENCE



- Atributo es local al AS — mandatorio para las actualizaciones de iBGP
- route-map: *set local-preference*

Problema: Sobrepasando Local Preference
Solución: WEIGHT



- Local al router en que es configurado
- route-map: *set weight*
- El mayor peso (weight) gana sobre todos los caminos válidos

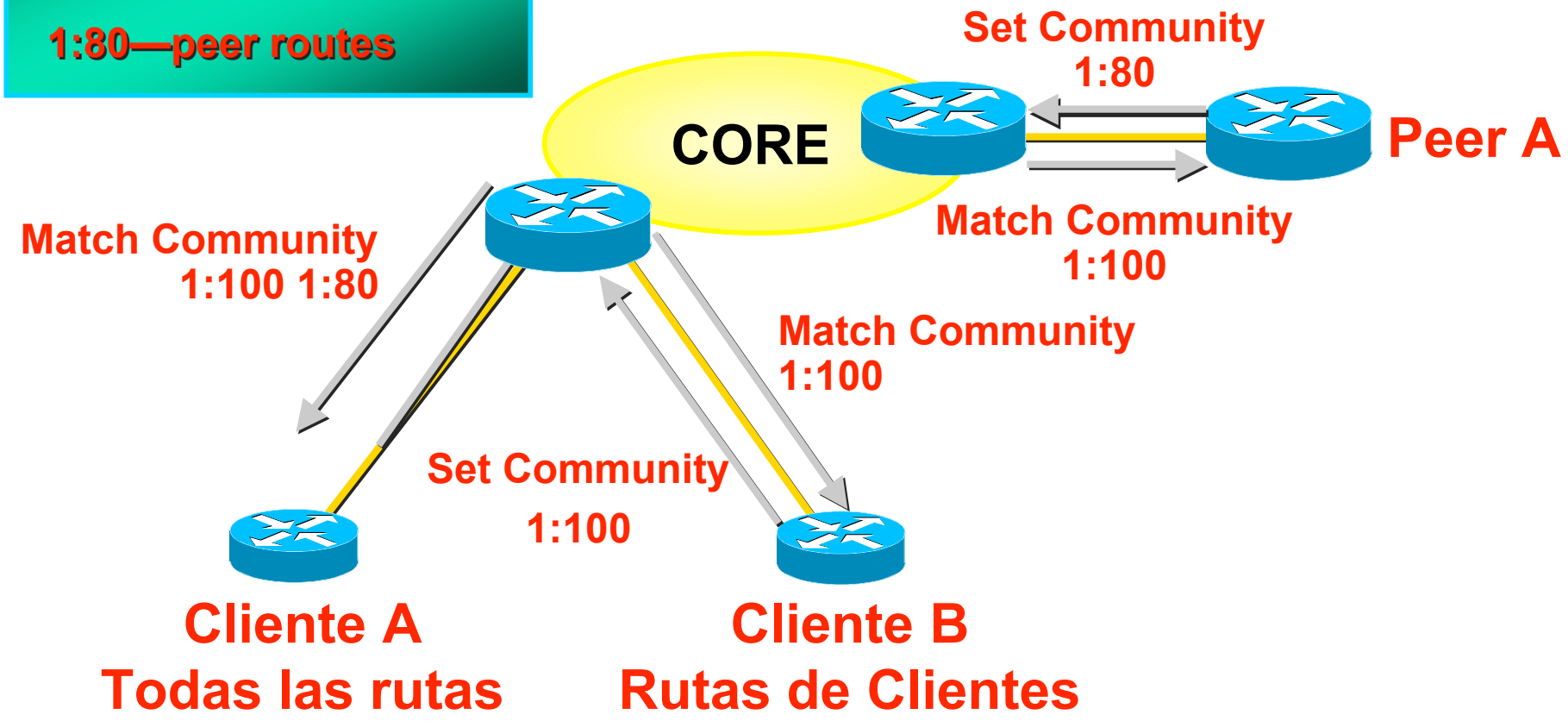
Problema: Escalando Políticas de Enrutamiento

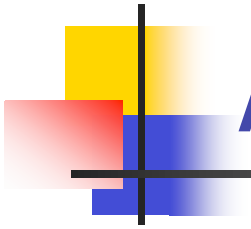
Solución: COMMUNITY

Communities:

1:100—customer routes

1:80—peer routes

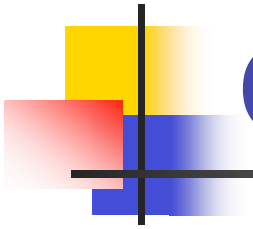




Atributo de BGP: COMMUNITY

- Agrupa los destinos para ayudar a escalar la aplicación de políticas
- Comunidades Típicas:
  - Destinos aprendidos de los clientes
  - Destinos aprendidos de los peers
  - Destinos en la VPN
  - Destinos que reciben tratamiento preferencial en la cola

Atributos de BGP: COMMUNITY



- Activación por neighbor/peer-group:
  - *neighbor {peer-address | peer-group-name} send-community*
- Transferidos a través de ASs
- Formato común es una cadena de 4 bytes: <AS>:[0-65536]

Atributos de BGP: COMMUNITY (Cont.)

- Cada destino puede ser miembro de varias comunidades
- Route-map: *set community*
  - <1-4294967295> número de comunidad
  - aa:nn número de comunidad en formato aa:nn
  - *additive* *Añade a una comunidad existente*
  - *local-AS* *No enviar a los peers EBGP (well-known community)*
  - *no-advertise* *No enviar a ningún peer (well-known community)*
  - *no-export* *No exportar fuera del AS/Conf. (well-known community)*
  - *none* *No atributo de comunidad*

Atributo de menor uso: ORIGIN

- IGP—creado con comando *network* en la configuración de BGP
- EGP—Redistribuido de EGP
- Incomplete—Redistribute IGP en la configuración de BGP
- NOTA: siempre usar route-map para modificar: *set origin igp*

Comando *set* en un route-map

- as-path — Añade una cadena de AS para el atributo AS-PATH
- comm-list — set BGP community list (for deletion)
- community — Atributo de Comunidad
- dampening — Configura parámetros para dampening
- local-preference — Atributo de preferencia local de BGP
- metric — Valor Metric para el protocolo de enrutamiento
- origin — Codigo de origen BGP
- weight — Peso BGP para la tabla de enrutamiento
- ip next-hop { A.B.C.D | peer-address }

Atributos de BGP

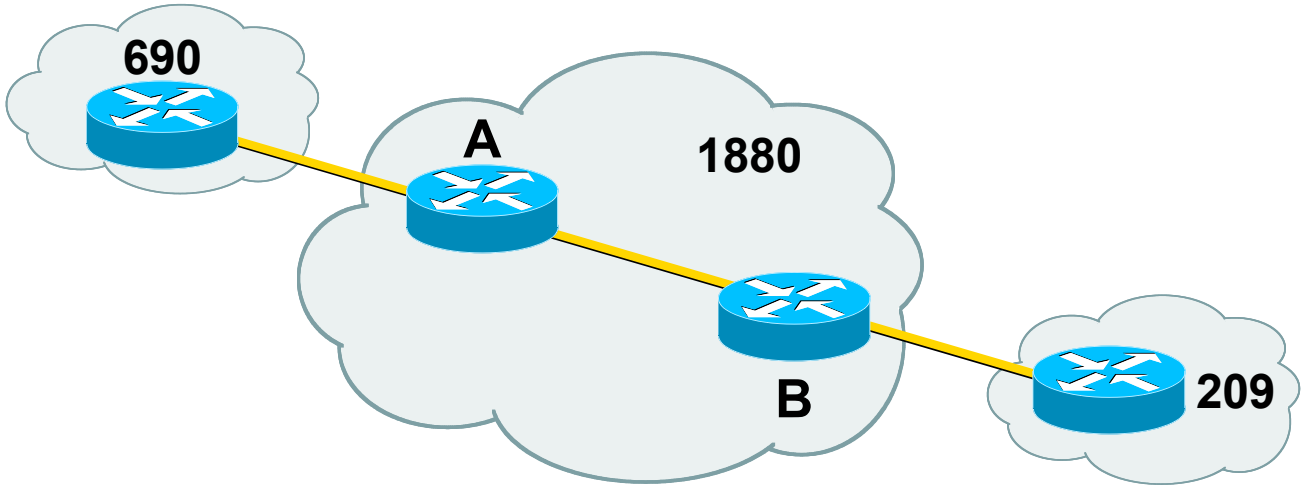
```
75k1#sh ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/24, version 139267814
Paths: (1 available, best #1)
  Not advertised to any peer as
  ! AS-PATH                                              AS      ID
  65000 64000 {100 200},  (aggregated by 64000 16.0.0.2)
  !  NEXT-HOP    IGP METRIC          PEER-IP PEER-ID
   10.0.10.4        (metric 10)    from 10.0.0.1 (10.0.0.2)
     Origin IGP, metric 100, localpref 230, valid, aggregated
     internal (or external or local),
     atomic-aggregate, best
     Community: 64000:3 100:0 200:10
     Originator: 10.0.0.1, Cluster list: 16.0.0.4, 16.0.0.14
```

Algoritmo Básico Para Decidir

Considera solo las rutas sincronizadas sin referencia circular y un next-hop válido, entonces prefiere:

Mayor WEIGHT
Mayor LOCAL PREFERENCE
ORIGINADA LOCALMENTE (eg network/aggregate)
Más Corto AS-PATH
Menor ORIGIN (IGP < EGP < incomplete)
Menor MED
EBGP
IBGP
Menor IGP METRIC al next-hop

Sincronización



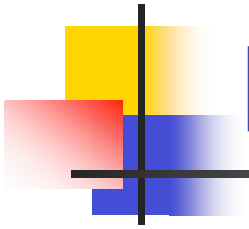
- **Router A no anunciará los prefijos de AS209 hasta que haya convergencia en el IGP.**
- **Asegurarse de que los next-hops del iBGP pueden ser vistos via IGP, entonces:**

```
router bgp 1880
no synchronization
```

Consideraciones Generales

- Sincronización: no requerida si se tiene una maya iBGP completa
- => No dejar que BGP tenga prioridad sobre IGP
- auto-summary: no. En su lugar usar comandos de agregación:

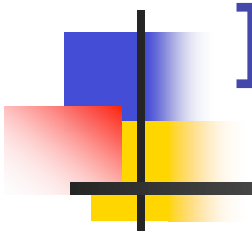
```
router bgp 100
no synchronization
no auto-summary
distance 200 200 200
```



Hasta Ahora ...

- Aplicar las políticas en base al AS
- Agrupar las rutas usando comunidades
- Seleccionar los puntos de entrada y salida para grandes grúpos de políticas usando MEDs y preferencia local

Pueden tús políticas escalar?

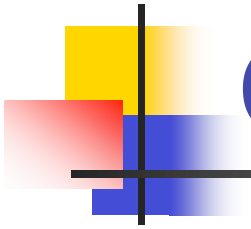


Implementando iBGP

Route Reflectors, Peer Groups

Guías para un iBGP Estable

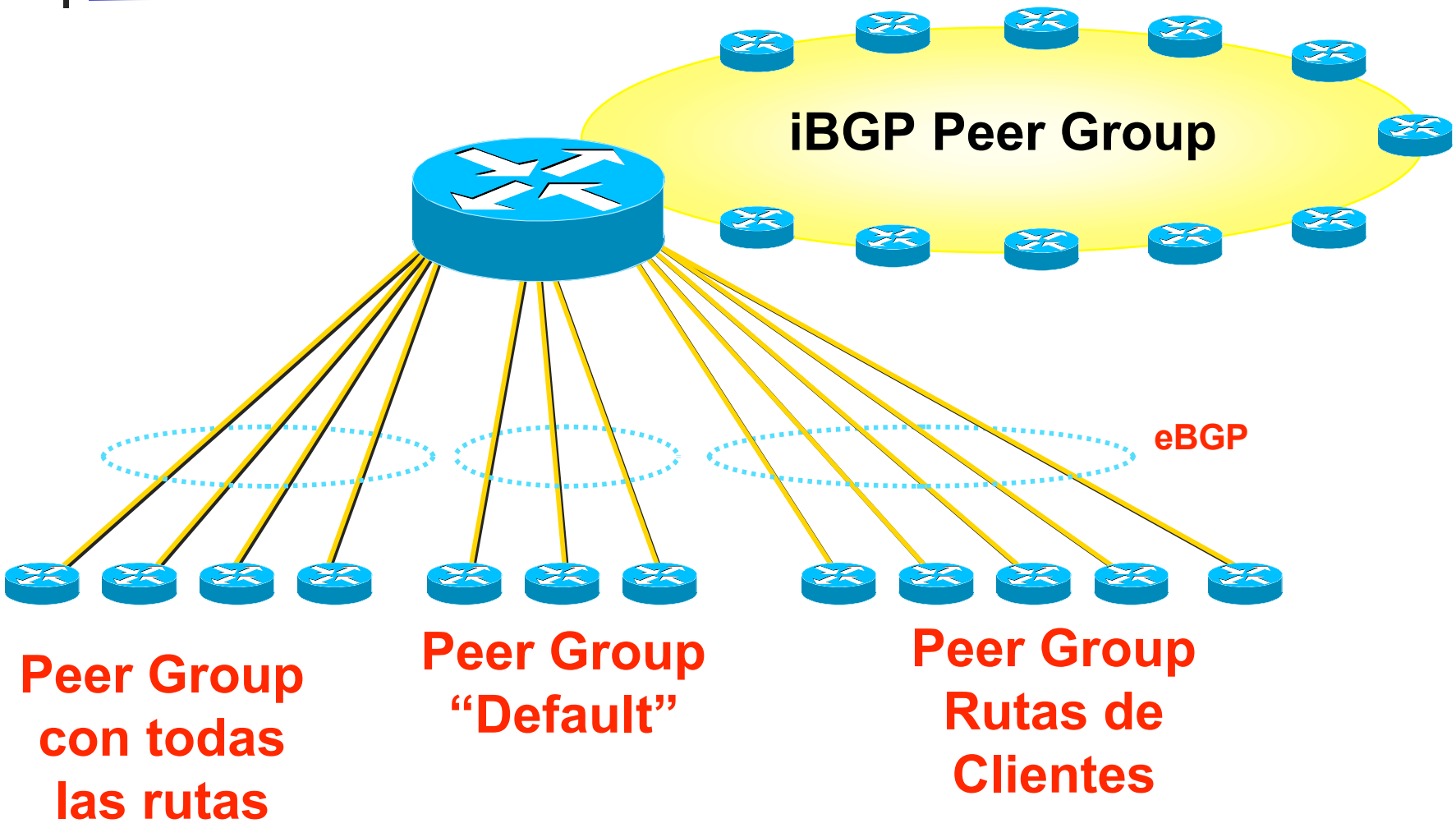
- Establecer la conexión usando direcciones de loopback
  - neighbor { ip address | peer-group}
    update-source loopback0
- Independiente de fallos de la interfase física
- Balanceamiento de la carga es realizado por el IGP

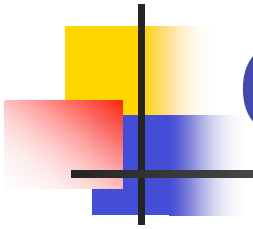


Guía Para Escalar iBGP

- Usar *peer-group* y *route-reflector*
- Solo llevar *next-hop* en el IGP
- Solo llevar todas las rutas en BGP si es necesario
- No redistribuir BGP en el IGP

Usando *Peer-Groups*



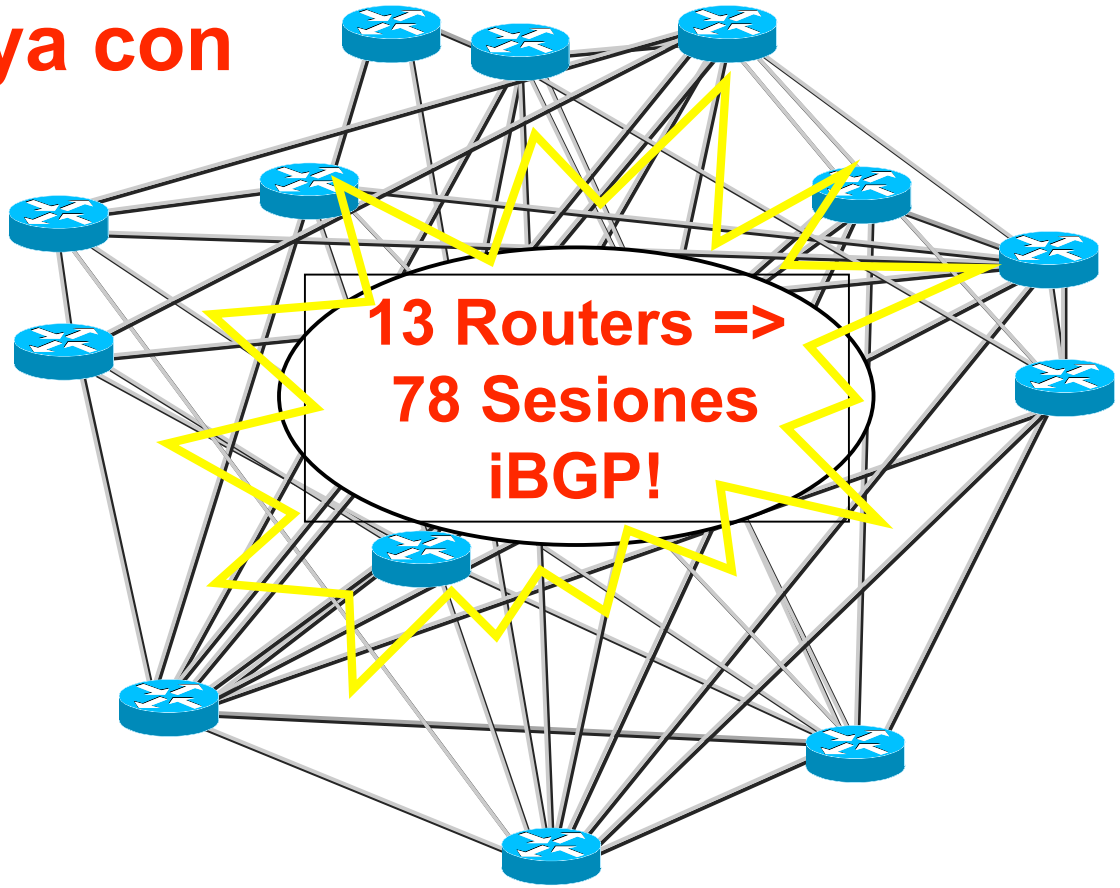


Qué es un *peer-group*?

- Todos los miembros del *peer-group* tienen una política de salida común
- Actualizaciones generadas solo una vez para el *peer-group*
- Simplifica la configuración
- Miembros pueden tener diferentes políticas de entrada

Por qué usar *Route-Reflectors*?

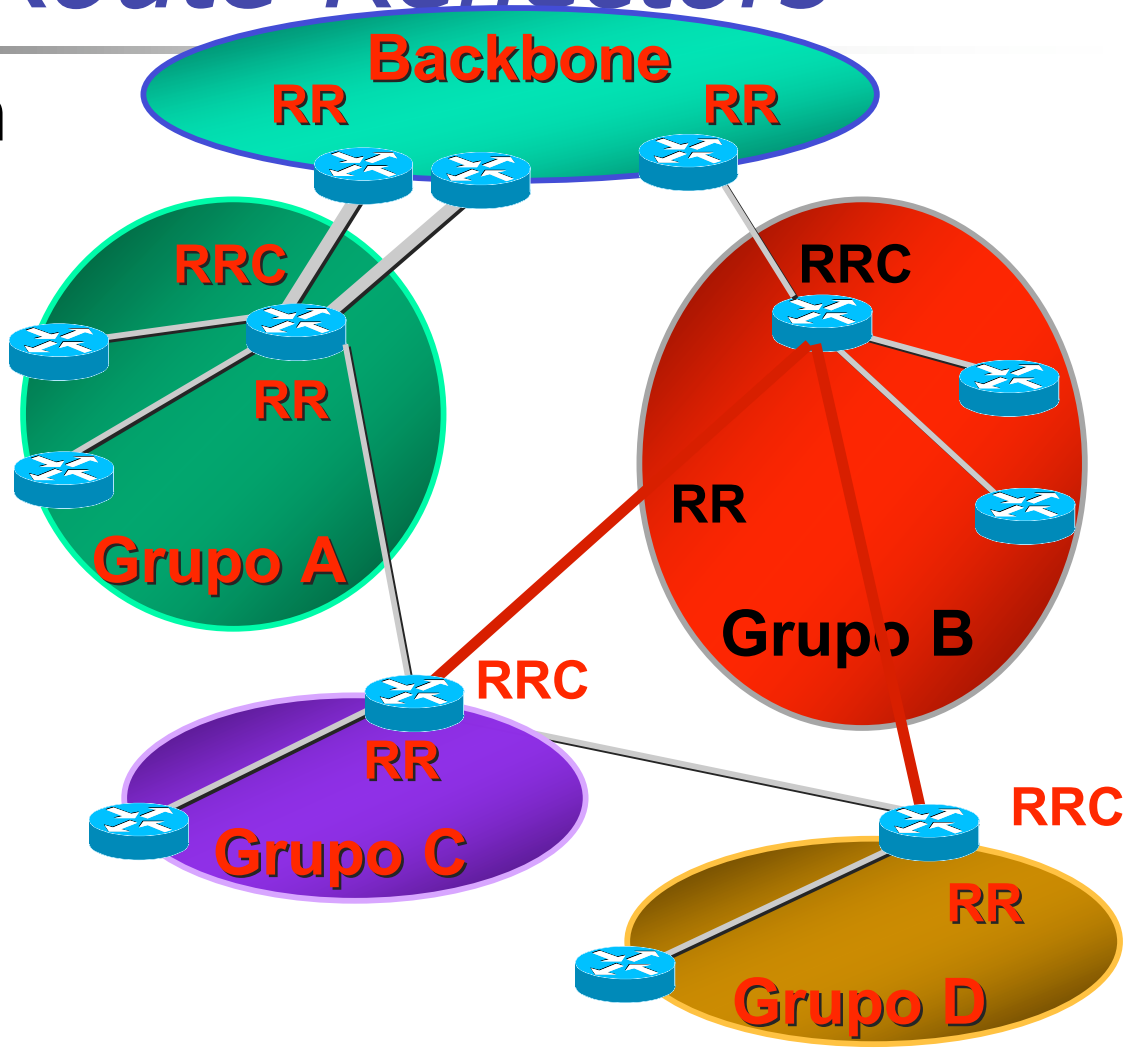
Para evitar una maya con N(n-1)/2 sesiones

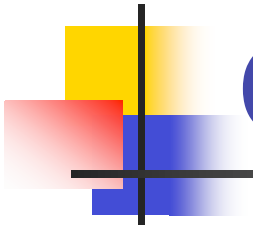


n=1000 => casi medio millón de sesiones iBGP!

Usando *Route-Reflectors*

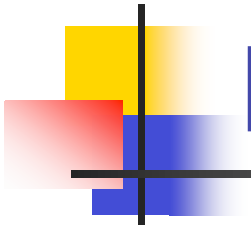
Regla para evtar un circulo de RR: Topología de RR debe reflejar la topología física





Qué es un *Route-Reflector*?

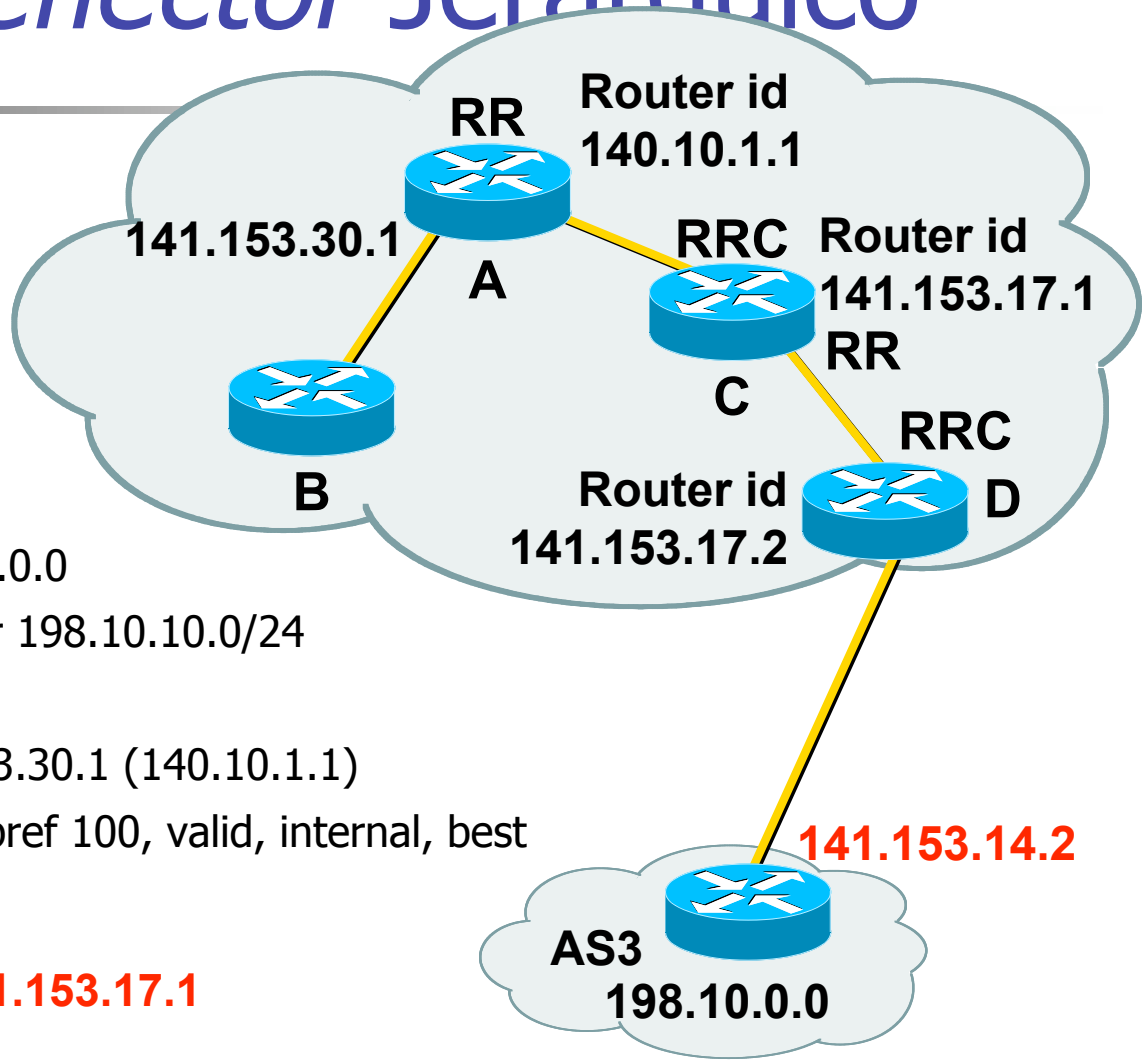
- El Reflector recibe información de clientes y no clientes
- Si el mejor camino es de un cliente, reflejarlo a clientes y no clientes
- Si el mejor camino es de no-cliente, reflejarlo a los clientes



Desplegando *Route-Reflectors*

- Divida el backbone en varios grupos
- Cada grupo contiene al menos 1 RR (multiples para redundancia), y multiples clientes
- Los RRs crean una maya completa de iBGP
- Utilizar solo un IGP—next-hop que no es modificado por el RR

Route-Reflector Jerárquico



- Ejemplo:

  RouterB>sh ip bgp 198.10.0.0

  BGP routing table entry for 198.10.10.0/24

  3

  141.153.14.2 from 141.153.30.1 (140.10.1.1)

  Origin IGP, metric 0, localpref 100, valid, internal, best

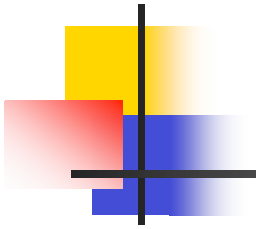
Originator: 141.153.17.2

Cluster list: 144.10.1.1, 141.153.17.1

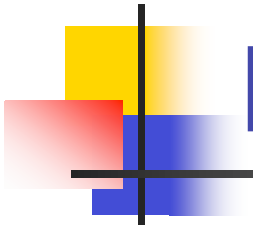
Atributos de BGP: ORIGINATOR_ID

- ORIGINATOR_ID
  - Router ID del vecino iBGP qye refleja rutas del cliente RR a no clientes
  - Sobrepasado por: *bgp cluster-id x.x.x.x*
- De uso para resolver problemas y chequear por relaciones circulares

Atributos de BGP: CLUSTER_LIST

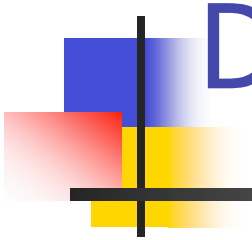


- CLUSTER_LIST
  - Cadena de ORIGINATOR_IDs a través de los cuales la ruta ha pasado
- De uso para resolver problemas y chequear relaciones circulares



Hasta Ahora…

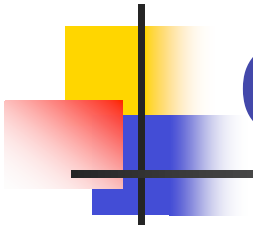
- Es la conexión iBGP Estable?
  - Use loopbacks para la conexión
- Escalará?
  - Use *peer-groups*
  - Use *route-reflectors*
- Simple, configuración jerárquica?



Desplegando eBGP

Consideraciones de Clientes

Consideraciones de ISPs

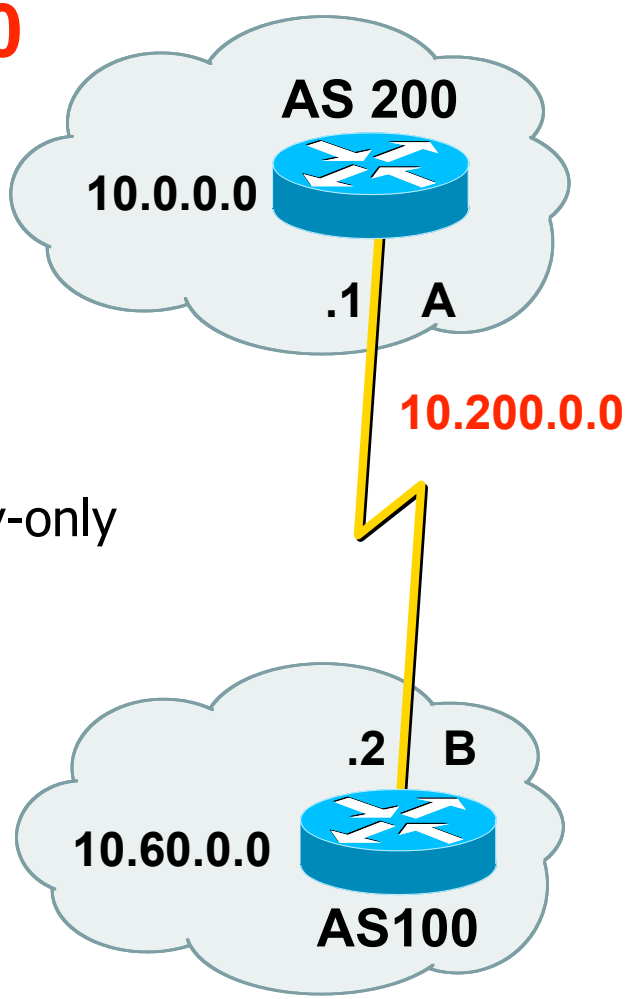


Consideraciones de Clientes

- Procedimiento
  - Configure BGP (use passwords para la sesión!)
  - Genere una ruta agregada estable
  - Configure la política de entrada
  - Configure la política de salida
  - Configure loadsharing/multihoming

Conectandose a un ISP

- **AS 100 es un cliente de AS 200**
- **Usualmente con conexión directa**

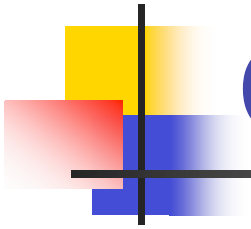


Router B:

```
router bgp 109
aggregate-address 10.60.0.0 255.255.0.0 summary-only
neighbor 10.200.0.1 remote-as 200
neighbor 10.200.0.1 route-map ispout out
neighbor 10.200.0.1 route-map ispin in
```

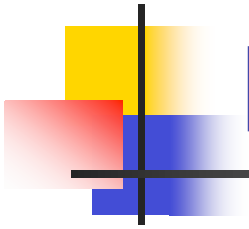
Que es agregación?

- Sumarización basada en rutas específicas *la tabla de enrutamiento BGP*
  - 10.1.1.0 255.255.255.0
  - 10.2.0.0 255.255.0.0
  - => 10.0.0.0 255.0.0.0



Cómo Agregar?

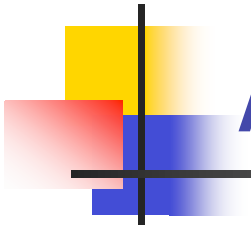
- aggregate-address 10.0.0.0 255.0.0.0 {as-set} {summary-only} {route-map}
- Use *as-set* para incluir la información de camino y comunidad basado en las rutas específicas
- *summary-only* suprime las rutas específicas
- route-map para configurar otros atributos



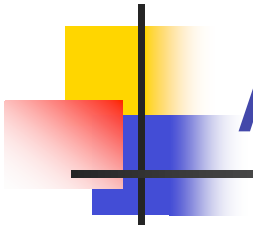
Por qué Agregar?

- Reducir el número de prefijos a anunciar
- Incrementar estabilidad — rutas agregadas se mantienen aún si las específicas son inestables
- Generación de rutas agragadas estables:
  - router bgp 100
  - aggregate-address 10.0.0.0 255.0.0.0 as-set summary-only
  - network 10.1.0.0 255.255.0.0
  - :
  - ip route 10.1.0.0 255.255.0.0 null0

Atributos de BGP: Atomic Aggregate

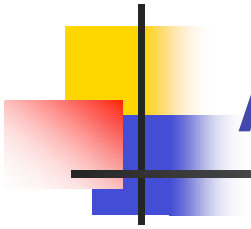


- Indica perdida de información de AS-PATH
- No debe ser removido una vez configurado
- Configuración: *aggregate-address x.x.x.x*
- No configurado si la clave as-set es utilizada, AS-SET y COMMUNITY contienen la información sobre las rutas específicas



Atributos de BGP: Aggregator

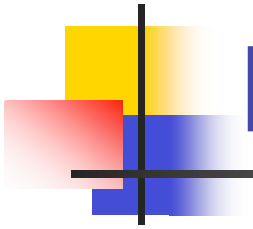
- Número de AS e IP del enrutador generando el agregado
- De uso para resolver problemas



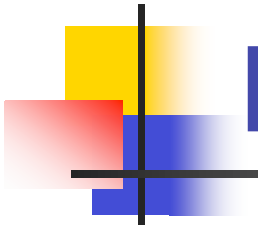
Atributos de Agregación

- NEXT_HOP = local (0.0.0.0)
- WEIGHT = 32768
- LOCAL_PREF = ninguna (asume 100)
- AS_PATH = AS_SET o nada
- ORIGIN = IGP
- MED = ninguna

Por qué una Política de Entrada?



- Aplicar una comunidad reconocible para usar en los filtros de salida u otras políticas
- Configurar local-preference para modificar el default de 100
- Balanceo de las cargas en ambientes de conexión dual—ver mas adelante
  - route-map ISPin permit 10
  - set local-preference 200
  - set community 100:2

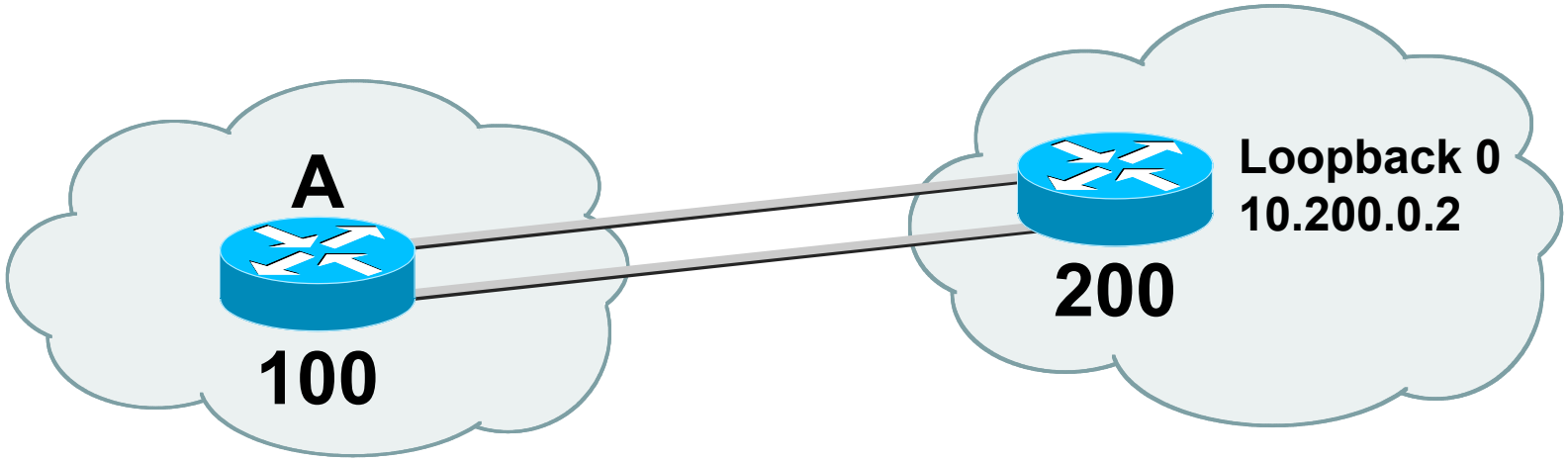


Por qué una política de Salida?

- El filtrado de prefijos de salida ayuda a protegernos contra errores (también podemos aplicar filtros de comunidades y as-path)
- Enviar comunidades basado en acuerdos con el ISP
  - route-map ISPout permit 10
  - match ip address prefix-list outgoing
  - set community 100:1 additive

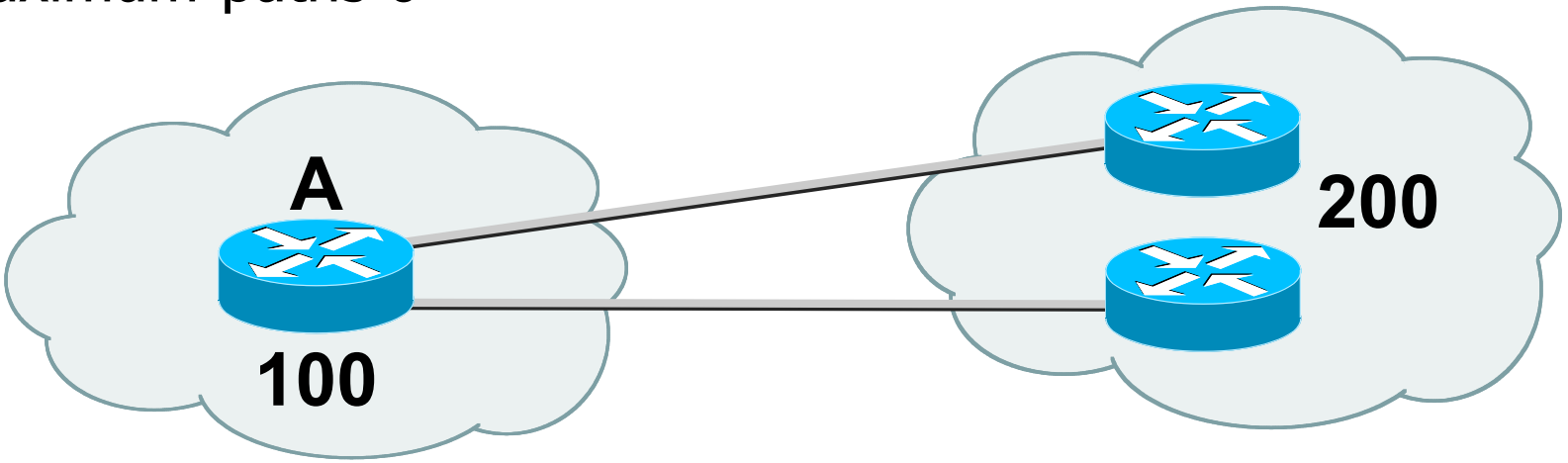
Balanceo de Cargas —Un ISP

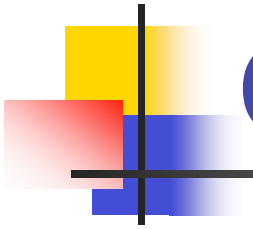
Router A:
interface loopback 0
ip address 10.60.0.1 255.255.255.255
!
router bgp 100
neighbor 10.200.0.2 remote-as 200
neighbor 10.200.0.2 update-source loopback0
neighbor 10.200.0.2 ebgp-multi-hop 2



Balanceo de Cargas—Multiples Caminos desde el mismo AS

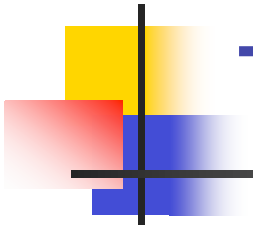
Router A:
router bgp 100
neighbor 10.200.0.1 remote-as 200
neighbor 10.300.0.1 remote-as 200
maximum-paths 6





Qué es Multihoming?

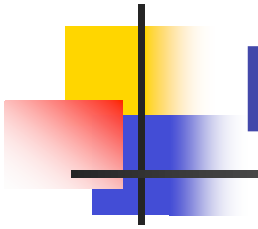
- Conectarse a dos o más ISPs para incrementar:
  - Confiabilidad—si un ISP falla, todavía funciona
  - Desempeño—mejores caminos a destinos comunes en Internet



Tipos de Multihoming

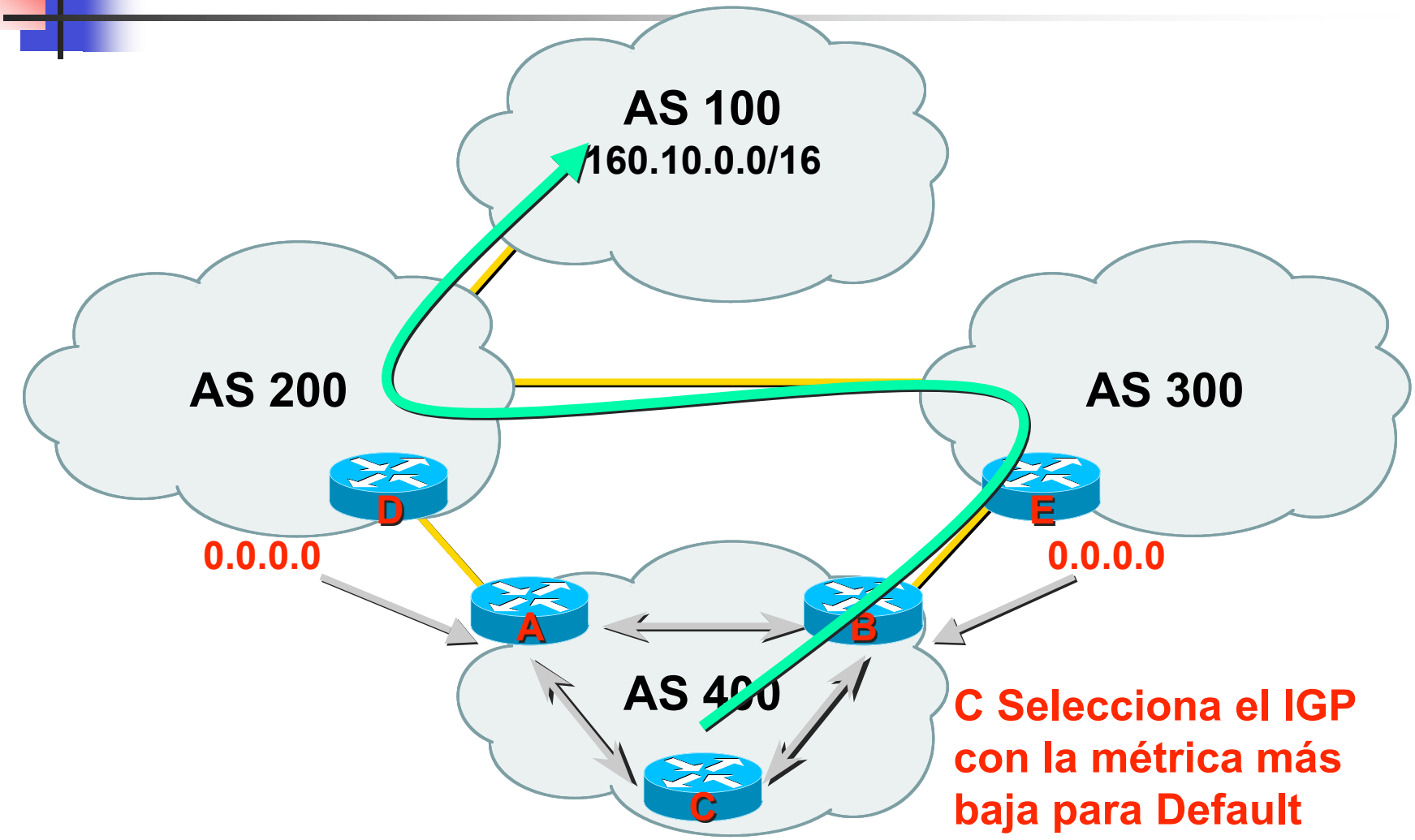
- Tres casos comunes:
  - Ruta Default de todos los proveedores
  - Rutas de Clientes+default de todos
  - Rutas completas de todos

Default de Todos los Proveedores

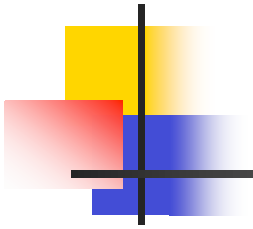


- Solución bajos requerimientos de memoria/CPU
- Proveedor default de BGP => proveedor decide basado en la métrica de IGP cual es el default
- Tu envias todas tus rutas al proveedor => camino de entrada decidido por el Internet
  - Tu puedes influenciar agregando tu AS varias veces (AS-path prepend)

Default de Todos los Proveedores

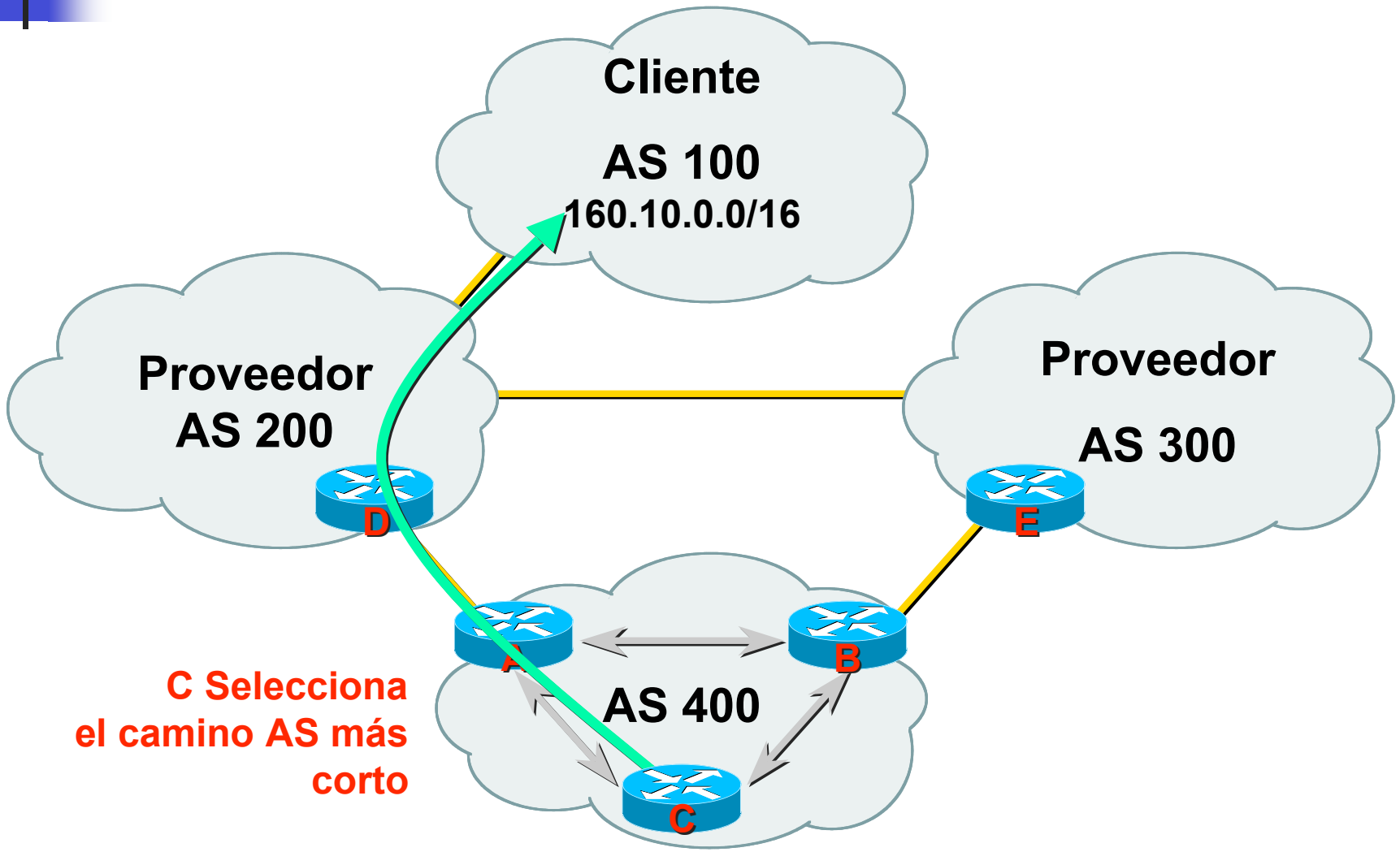


Clientes+Default de Todos los Proveedores

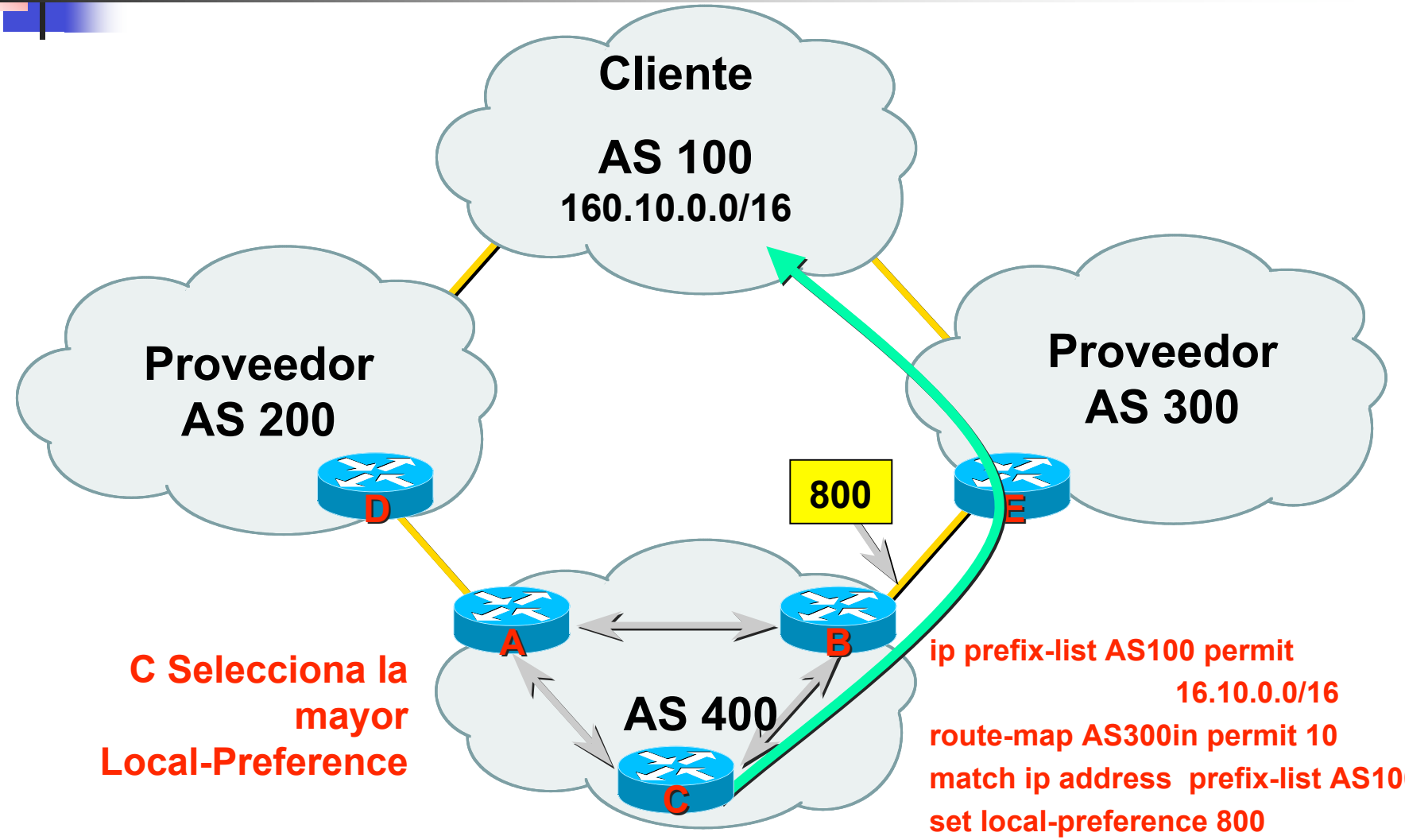


- Uso mediano de memoria y CPU
- “Mejor” camino—usualmente el camino AS (AS PATH) más corto
- Uso de local-preference para modificar basado en prefijo, as-path, o comunidad
- Métrica de IGP al default usado para todos los destinos

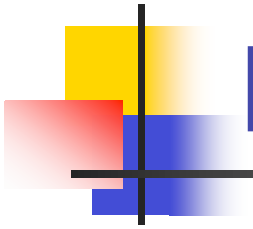
Rutas de Clientes de Todos los Proveedores



Rutas de Clientes de Todos los Proveedores

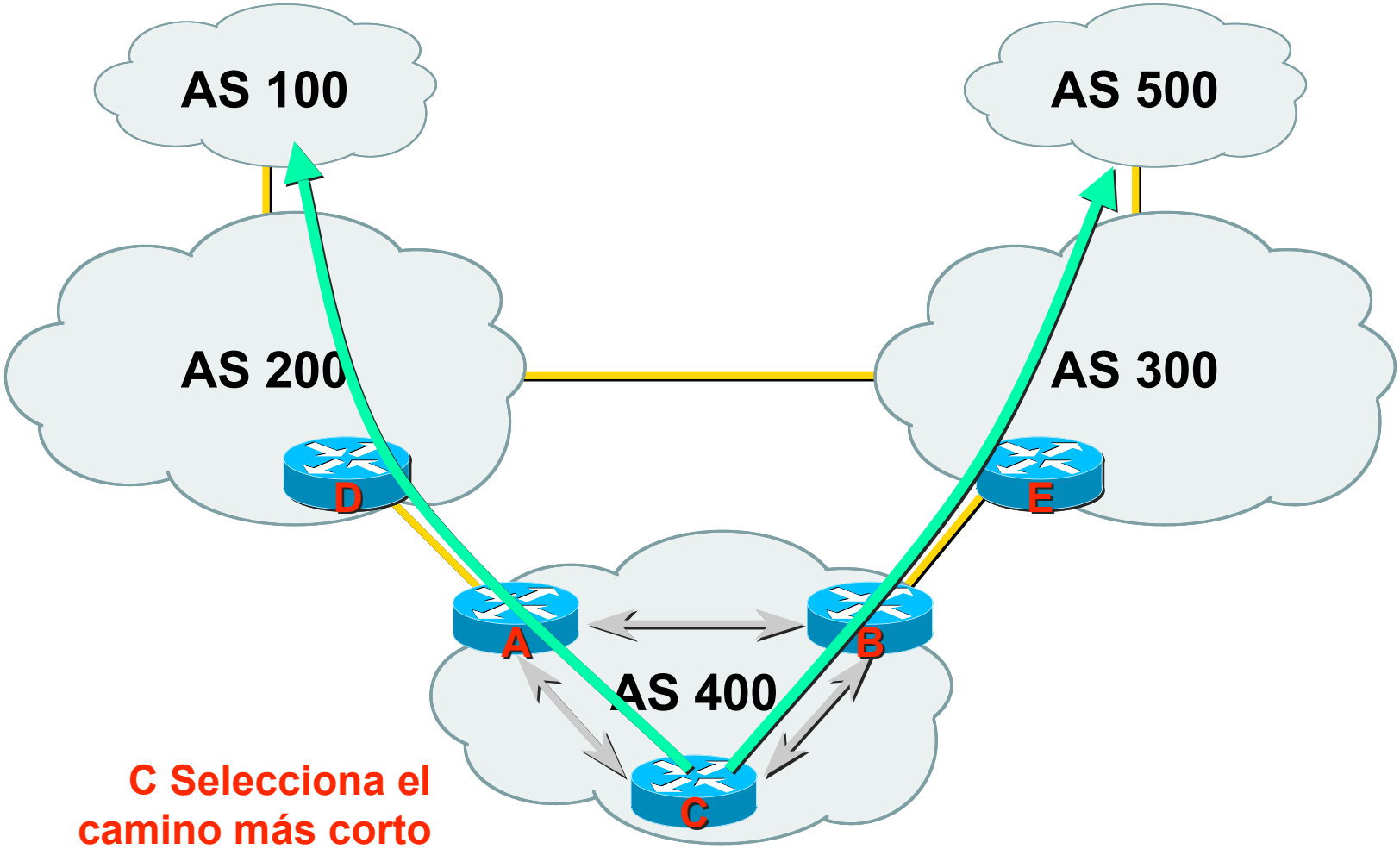


Rutas Completas de Ambos Proveedores

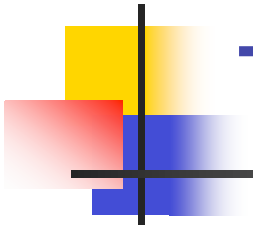


- Solución de mayores requerimientos de memoria/CPU
- Alcanza todos destinos basado en el “mejor” camino—usualmente el que tiene el camino mas corto
- Todavía puede ajustar manualmente usando local-pref y comparación de as-path/comunidad/prefijo

Rutas Completas de Ambos Proveedores

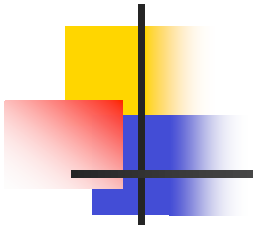


Controlando la Entrada de Tráfico?



- La entrada es muy dificil de controlar debido a la falta de una métrica transitiva
- Puedes dividir los anuncios de prefijos entre los proveedores, pero entonces, que pasa con la redundancia?

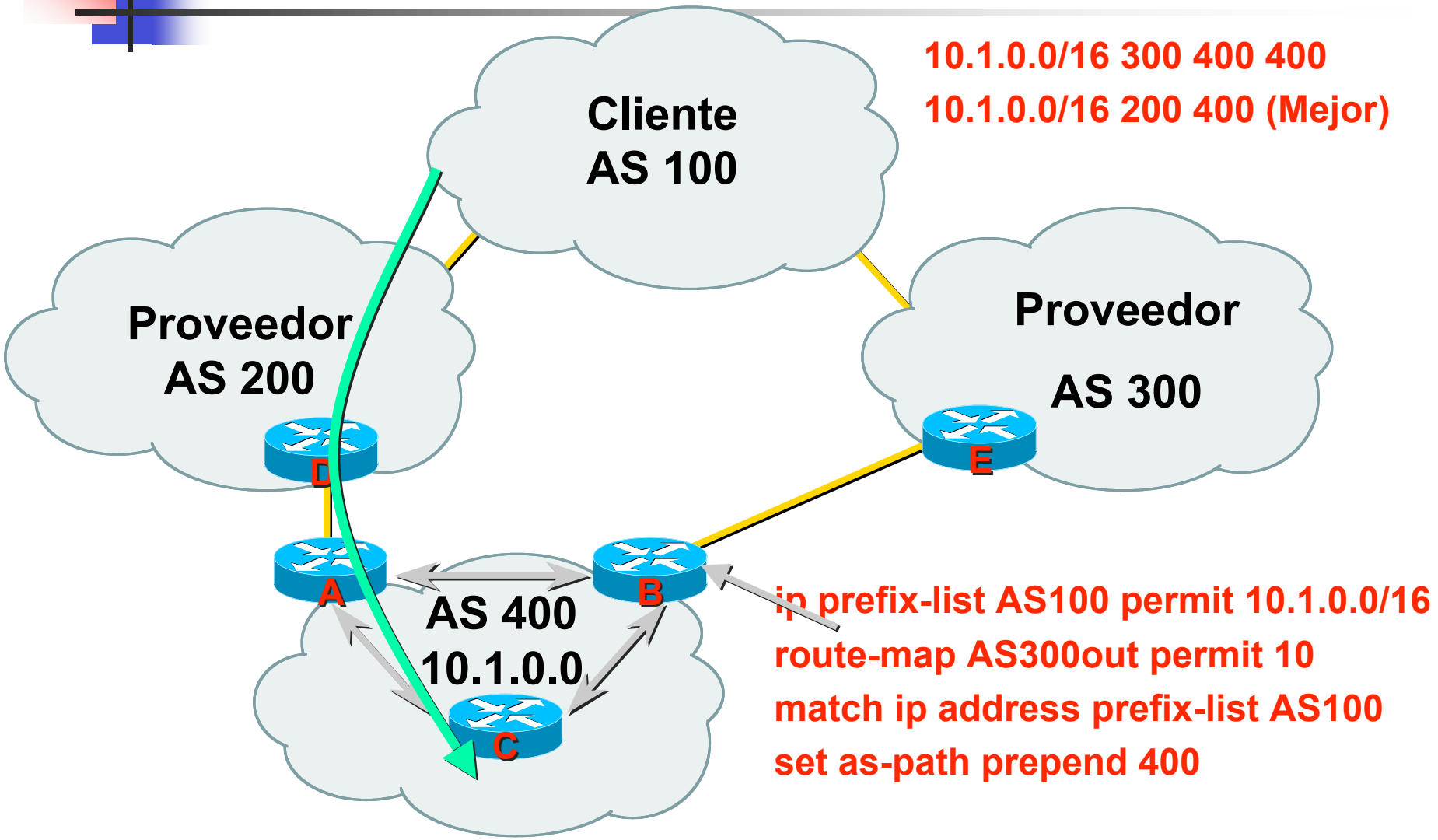
Controlando la Entrada de Tráfico? (Cont.)



- **Mal Ciudadano Internet:**
  - **Divide el espacio de direcciones**
  - **Usa "as-path prepend"**

- Buen Ciudadano :
  - Divide el espacio de direcciones
  - Usa "advertise maps"

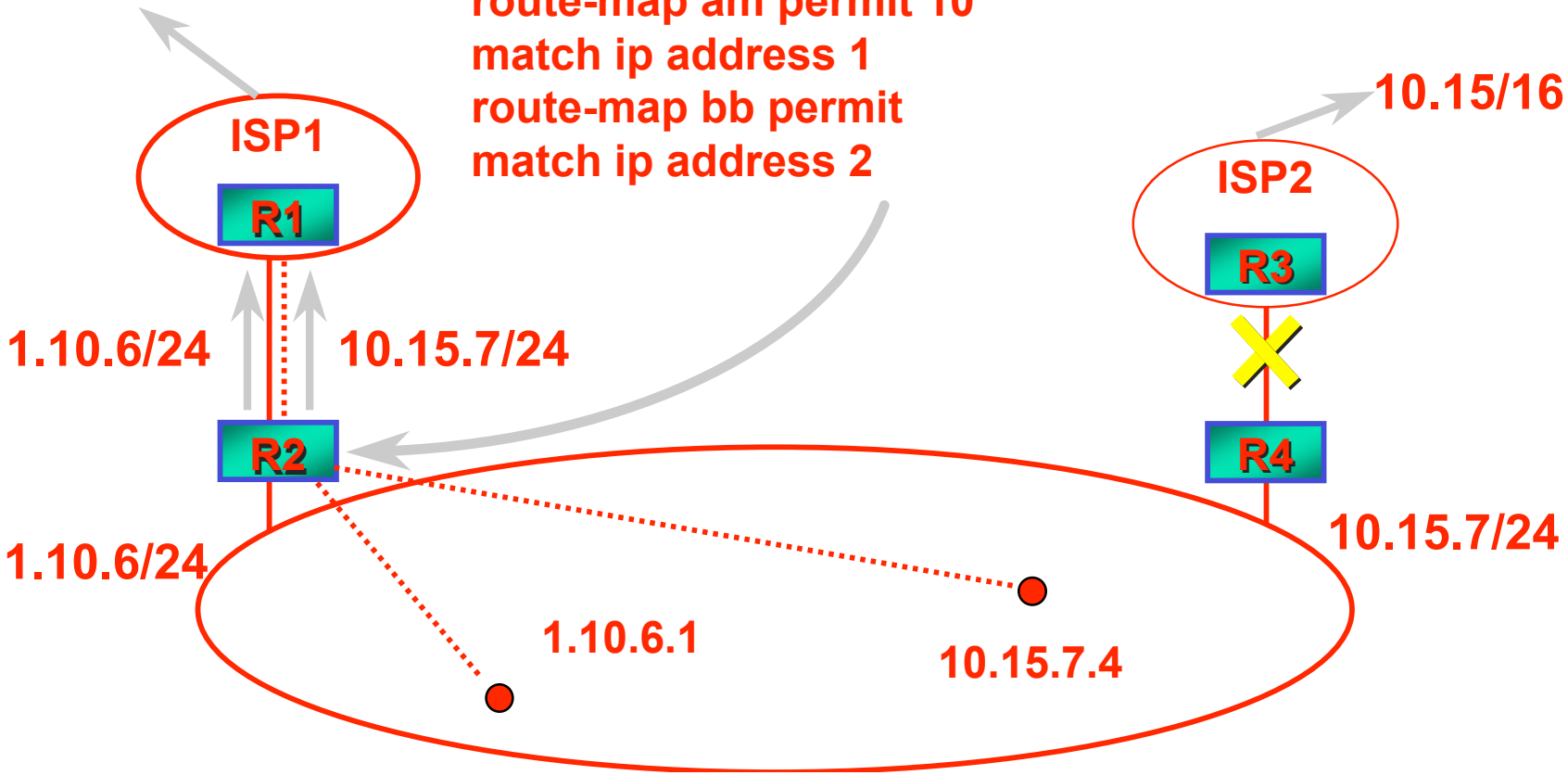
Usando "*AS-PATH prepend*"

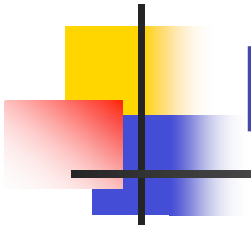


Usando un "*Advertise-Map*"

1.10/16
10.15.7/24 auto-inject

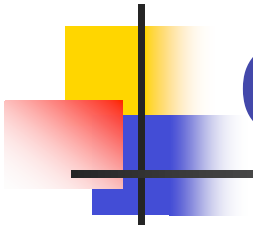
```
access-list 1 permit 10.15.7.0     !Anuncia cuando...
access-list 2 permit 10.15.0.0     !… este desaparece
neighbor <R1> advertise-map am non-exist-map bb
route-map am permit 10
match ip address 1
route-map bb permit
match ip address 2
```





Hasta Ahora…

- Estabilidad por Medio de:
  - Agregación
  - Multihoming
  - Filtrado de Entreda/Salida
- Escalabilidad de memoria/CPU:
  - Default, rutas de clientes, todas las rutas
- Simplicidada usando soluciones “estandares”



Consideraciones de ISPs

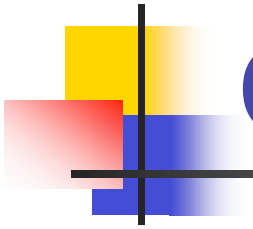
- Escalar la agragación de clientes en BGP
- Ofrecer una selección del número de rutas a anunciar
- Intercambio con otros proveedores
- Minimizar la actividad de BGP y protegerse contra los problemas de configuración de los clientes
- Proveer un servicio alternativo
- Propagar una política de Calidad de Servicio

Guías para el Agregado de Clientes

- Definir por los menos tres *"peer-groups"*:
  - cliente-default — envía solo default
  - cliente-cliente — envía solo las rutas de clientes
  - Cliente-completo — envía todas las rutas
- Identificar las rutas a traves de comunidades
  - 2:100=clientes; 2:80=peers
- Aplicar claves y un "prefix-list" para cada vecino BGP

Agregado de Clientes

NOTA: Aplicar claves y "prefix-list" de entrada para cada Cliente



Cliente-completo peer-group

neighbor cliente-completo peer-group

neighbor cliente-completo description Envía todas las rutas

neighbor cliente-completo remove-private-AS

neighbor cliente-completo version 4

neighbor cliente-completo route-map cliente-entrada in

neighbor cliente-completo prefix-list cidr-block out

neighbor cliente-completo route-map rutas-completas out

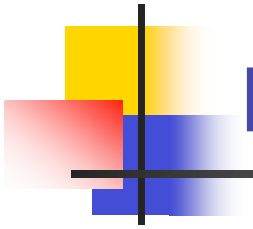
.

ip prefix-list cidr-block seq 5 deny 10.0.0.0/8 ge 9

ip prefix-list cidr-block seq 10 permit 0.0.0.0/0 le 32

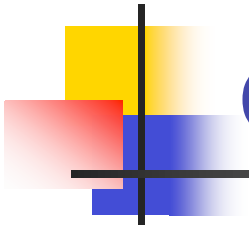
route-map de salida para cliente-completo

```
ip community-list 1 permit 2:100
ip community-list 80 permit 2:80
.
route-map rutas-completas permit 10
match community 1 80      ;  clientes & peers
set metric-type internal     ; MED = métrica IGP
set ip next-hop peer-address ; la nuestra
```



route-map cliente-entrada

```
route-map cliente-entrada permit 10
set metric 4294967294    ; ignora MED
set ip next-hop peer-address
set community 2:100 additive
```



cliente-cliente peer-group

neighbor cliente-cliente peer-group

neighbor cliente-cliente description Rutas de Clientes

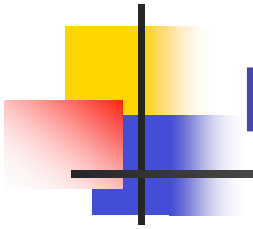
neighbor cliente-cliente remove-private-AS

neighbor cliente-cliente version 4

neighbor cliente-cliente route-map cliente-entrada in

neighbor cliente-cliente prefix-list cidr-block out

neighbor cliente-cliente route-map rutas-clientes out



route-map rutas-clientes

route-map rutas-clientes permit 10

match community 1 ; solo clientes

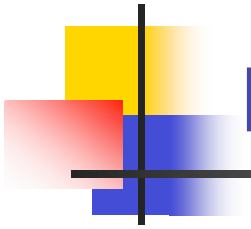
set metric-type internal ; MED = métrica igp

set ip next-hop peer-address ; la nuestra

route-map ruta-default

neighbor cliente-default peer-group
neighbor cliente-default description Envía Default
neighbor cliente-default default-originate
route-map ruta-default
neighbor cliente-default remove-private-AS
neighbor cliente-default version 4
neighbor cliente-default route-map cliente-entrada
neighbor cliente-default prefix-list niega-todo out

ip prefix-list niega-todo seq 5 deny 0.0.0.0/0 le 32



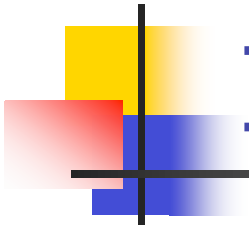
route-map ruta-default

route-map ruta-default permit 10

set metric-type internal ; MED = métrica igp

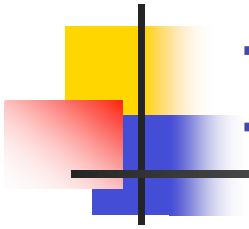
set ip next-hop peer-address ; la nuestra

Peer Groups para Puntos de Intercambio



- Similar al EBGP para agregado de clientes excepto que no se usa el filtrado de prefijos (porque no hay un registro)
- En su lugar se usa *maximum-prefix* y chequeos de sanidad de prefijos
- Continua usando claves para cada vecino!

Peer Groups para Puntos de Intercambio (Cont.)



neighbor nap peer-group

neighbor nap descripción de ISP

neighbor nap remove-private-AS

neighbor nap version 4

neighbor nap prefix-list chequeo sanidad in

neighbor nap prefix-list cidr-block out

neighbor nap route-map nap-salidas out

neighbor nap maximum prefix 30000

Peer Groups para Puntos de Intercambio (Cont.)

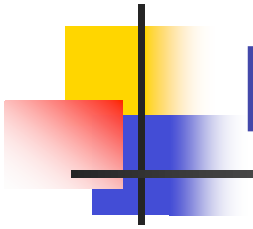
```
route-map nap-salida permit 10
match community 1          ; solo clientes
set metric-type internal  ; MED = métrica IGP
set ip next-hop peer-address ; la nuestra
```

Peer Groups para Puntos de Intercambio : Prefix-List chequeo-sanidad

```
# Primero filtramos nuestro espacio de direcciones!!
ip prefix-list chequeo-sanidad seq 5 deny 0.0.0.0/32
# no aceptamos default
ip prefix-list chequeo-sanidad seq 10 deny 0.0.0.0/8 le 32
# no aceptamos nada que comience con  0
ip prefix-list chequeo-sanidad seq 15 deny 0.0.0.0/1 le 32
# no aceptamos mascaras  > 20 para todas las redes clase A (1-127)
ip prefix-list chequeo-sanidad seq 15 deny 0.0.0.0/8 ge 20
# no aceptamos 10/8 per RFC1918
ip prefix-list chequeo-sanidad seq 20 deny 10.0.0.0/8 le 32
# reservado por IANA – dirección de loopback
ip prefix-list chequeo-sanidad seq 25 deny 127.0.0.0/8 le 32
# no acepta mascaras >= 17 para todas las redes clase B (129-191)
ip prefix-list chequeo-sanidad seq 30 deny 128.0.0.0/2 ge 17
# no acepta la red 128.0 – reservado por IANA
ip prefix-list chequeo-sanidad seq 35 deny 128.0.0.0/16 le 32
```

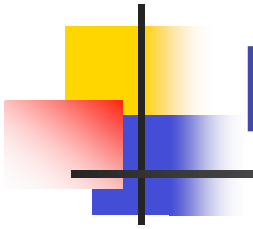
Peer Groups for NAPs: Prefix-List chequeo-sanidad

```
# no acepta 172.16 por RFC1918
ip prefix-list chequeo-sanidad seq 40 deny 172.16.0.0/12 le 32
# no acepta clase C 192.0.20.0 reservado por IANA
ip prefix-list chequeo-sanidad seq 45 deny 192.0.2.0/24 le 32
# no acepta clase C 192.0.0.0 reservado por IANA
ip prefix-list chequeo-sanidad seq 50 deny 192.0.0.0/24 le 32
# no acepta 192.168/16 por RFC1918
ip prefix-list chequeo-sanidad seq 55 deny 192.168.0.0/16 le 32
# no acepta 191.255.0.0 – reservado por IANA (Creo ??)
ip prefix-list chequeo-sanidad seq 60 deny 191.255.0.0/16 le 32
# no acepta mascaras > 25 para clase C (192-222)
ip prefix-list chequeo-sanidad seq 65 deny 192.0.0.0/3 ge 25
# no acepta nada en red 223 – reservado por IANA
ip prefix-list chequeo-sanidad seq 70 deny 223.255.255.0/24 le 32
# no acepta clase D/Experimental
ip prefix-list chequeo-sanidad seq 75 deny 224.0.0.0/3 le 32
```



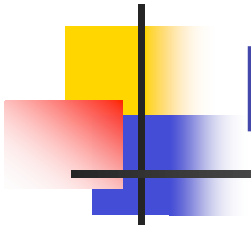
Resumen

- Escalabilidad:
  - Uso de atributos, especialmente comunidad (community)
  - Uso de peer-groups y route-reflectors
- Estabilidad:
  - Uso de dirección de loopback para el iBGP
  - Generación de agregados
  - Aplicación de claves
  - Siempre filtrar anuncios de entrada y salida



Resumen

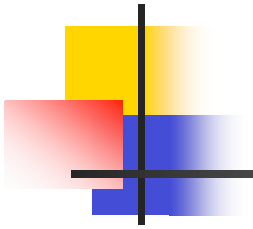
- Simplicidad—soluciones estandares:
  - Tres opciones de multihoming
  - Agrupar clientes en comunidades
  - Aplicación de políticas estandares en el borde
  - Evitar “configuraciones especiales”
  - Automatice la generación de la configuracion (RR & RtConfig)



Referencias/Fuentes:

- Cisco (www.cisco.com)
- Dave Meyer (dmm@sprint.net)
- John Stewart, BGP4, Addison Wesley
- Sam Halabi, “Internet Routing Architectures”, Cisco Press
- RFCs

Ejemplos de filtros para clientes



ip prefix-list anuncia-mi-prefijo seq 10 permit <numero_red>/<mascara> ge 23

ip prefix-list anuncia-mi-prefijo seq 100 deny 0.0.0.0/32 le 32

ip prefix-list acepta-default seq 10 permit 0.0.0.0/0 ge 32

ip prefix-list acepta-default seq 100 deny 0.0.0.0/0 le 31

access-list 10 permit <numreo_red> <mascara_wildcard>

access-list 10 deny any

access-list 20 permit 0.0.0.0 0.0.0.0

access-list 20 deny any