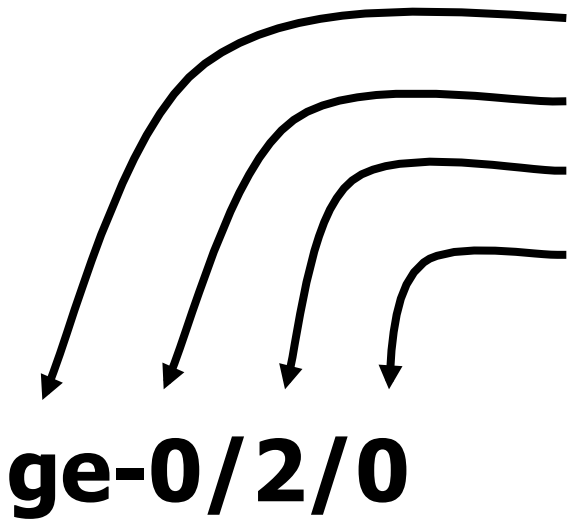
Workshop JUNOS

Parte B

Rogelio Alvez
ralvez@tiagora.com

Identificación de interfaces

- interpretaciòn de la numeraciòn:
  - tipo de la interfaz
  - slot
  - sub slot
  - número del puerto



Asignación de dirección IPv4

- **Ejemplo**

```
lab@omaha> configure
[edit]
lab@omaha# edit interfaces ge-1/0/3
[edit interfaces ge-1/0/3]
lab@omaha# set unit 40 family inet address 10.0.20.1/24
lab@omaha# commit
```

- **El equipo lo presenta a la terminal como:**

```
interfaces {
   ge-1/0/3 {
       unit 40 {
           vlan-id 40;
           family inet {
               address 10.0.20.1/24;
           }
       }
   }
```

Rutas estàticas

- son rutas manuales que van a parar a la tabla de rutas en forma permanente
- Necesitan un “next-hop”, que puede ser una direcciòn IP, “discard” o “reject”
- Se las configura en la sección “`routing-options`”

```
[edit]
routing-options {
   static {
      defaults {
         static-options;
      }
      route destination-prefix{
         next-hop;
         static-options;
      }
```

Rutas estàticas- Ejemplo

```
routing-options {
 static {
     route 192.168.20.0/24 next-hop 10.0.0.1;
      route 192.168.21.0/24 discard;
      route 192.168.22.0/24 reject;
      }
 }
```

El operador lo tuvo que haber hecho de la siguiente manera:

```
# set routing-options static route 192.168.20.0/24 next-hop 10.0.0.1
# set routing-options static route 192.168.21.0/24 discard
# set routing-options static route route 192.168.22.0/24 reject
```

O bien de esta manera:

```
# edit routing-options <ENTER>
# set static route 192.168.20.0/24 next-hop 10.0.0.1
# set static route 192.168.21.0/24 discard
# set static route route 192.168.22.0/24 reject
```

Tablas de rutas

- Los equipos Juniper tienen algunas tablas de rutas por default:
  - `inet.0` IPv4 unicast
  - `inet.1` Multicast
  - `inet.2` MBGP reverse path forwarding (RPF)
  - `inet.3` Mapeo MPLS para next-hops IP
  - `inet.6` IPv6
  - `mpls.0` MPLS next hops

Protocolos de propagaciòn

Protocolos que aportan informaciòn a la tabla de rutas:

- Direct
- Local
- Static
- RSVP
- LDP
- OSPF
- IS-IS
- RIP
- Aggregate
- BGP

Preference

- Cada protocolo tiene una “preferencia” para el equipo
  - Es lo que en IOS se llama “distancia administrativa”
  - No confundir con la mètrica de los protocolos

- Valores que los protocolos tienen por default:

Direct/Local: 0
Static: 5
RSVP: 7
LDP: 9
OSPF internal route: 10
IS-IS Level 1 internal route: 15
IS-IS Level 2 internal route: 18
Default: 20
RIP: 100
. . .

. . .
RIPng: 100
PIM: 105
DVMRP: 110
Aggregate routes: 130
OSPF AS external routes: 150
IS-IS Level 1 external route: 160
IS-IS Level 2 external route: 165
BGP: 170
MSDP: 175

La tabla `inet.0`

Ejemplo de una tabla `inet.0` (de IPv4)

```
user@host> show route

inet.0: 49 destinations, 49 routes (49 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.11.0/24       *[Direct/0] 1d 08:19:20
                    > via at-0/1/0.100
10.0.11.1/32       *[Local/0] 1d 08:19:20
                      Local
192.168.1.0/24     *[BGP/170] 00:06:08, localpref 100
                       AS path: 1 I
                    > to 10.0.11.2 via at-0/1/0.100
192.168.16.0/21    *[Static/5] 00:02:40
                      Discard
                    [Aggregate/130] 00:36:17
                      Reject
192.168.20.0/24    *[Static/5] 00:06:12
                      Reject
```

JUNOS Workshop

Protocolos de propagación de rutas

Configuración OSPF

- **El operador “le dice” a OSPF cuàles interfaces pertenecen a cuál àrea**

```
protocols {
    ospf {
        area 0.0.0.0 {
            interface lo0.0;
            interface so-1/0/0.0;
        }
        area 0.0.0.1 {
            interface ge-0/1/0.0;
        }
    }
}
```

SU EQUIVALENTE IOS...

```
router ospf 1
 network 10.81.254.40 0.0.0.0 area 0
 network 10.81.40.14 0.0.0.1 area 0
 network 10.81.40.16 0.0.0.1 area 1
```

Cómo lo hizo el operador?

```
# set protocols ospf area 0.0.0.0 interface lo0.0
# set protocols ospf area 0.0.0.0 interface so-1/0/0.0
# set protocols ospf area 0.0.0.1 interface ge-0/1/0.0
```

O bien

```
# edit protocols ospf <ENTER>
# set area 0.0.0.0 interface lo0.0
# set area 0.0.0.0 interface so-1/0/0.0
# set area 0.0.0.1 interface ge-0/1/0.0
```

show ospf interface

```
user@host> show ospf interface
    Intf         State     Area      DR ID         BDR ID          Nbrs
    so-1/0/0.0   PtToPt    0.0.0.0   0.0.0.0       0.0.0.0           1
    ge-0/1/0.0   DR        0.0.0.1   10.81.254.20  0.0.0.0           0

IOS-RTR#sh ip ospf int
POS2/1 is up, line protocol is up
  Internet Address 10.81.40.17/31, Area 1
  Process ID 1, Router ID 10.81.254.40, Network Type POINT_TO_POINT, Cost: 1
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:09
  Supports Link-local Signaling (LLS)
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 2
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.81.254.20
  Suppress hello for 0 neighbor(s)
```

show ospf neighbor

- **Default output is a summary-like screen**
  - **`detail` option provides more information for each neighbor**

```
user@host> show ospf neighbor
  Address          Interface              State     ID               Pri  Dead
10.81.40.35        so-1/0/0.0             Full      10.81.254.31     128  35
10.81.40.37        ge-0/1/0.0             Full      10.81.254.40     128  38

IOS-RTR#sh ip ospf neighbor
Neighbor ID     Pri   State           Dead Time   Address         Interface
10.81.254.24      0   FULL/  -        00:00:33    10.81.40.14     POS2/0
10.81.254.23      0   FULL/  -        00:00:36    10.81.40.16     POS2/1
```

show ospf database

```
user@host> show ospf database

    OSPF link state database, Area 0.0.0.0
 Type       ID               Adv Rtr          Seq      Age  Opt  Cksum  Len
Router  *10.81.254.23     10.81.254.23     0x80000007    38  0x22 0xa184  84
Router   10.81.254.24     10.81.254.24     0x80000009    39  0x2  0x2eda  72
Router   10.81.254.40     10.81.254.40     0x80000002    43  0x22 0x2c2d  84
Network  10.81.9.46       10.81.254.24     0x80000002   190  0x2  0xa8ca  32

IOS-RTR#sh ip ospf da

          OSPF Router with ID (10.81.254.40) (Process ID 1)

              Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
10.81.254.23    10.81.254.23    14          0x80000007 0xA184   5
10.81.254.24    10.81.254.24    13          0x80000009 0x2EDA   4
10.81.254.40    10.81.254.40    17          0x80000002 0x2C2D   5

              Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.81.9.46      10.81.254.24    165         0x80000002 0xA8CA
```

Rutas que OSPF aporta a la tabla IP

```
user@host> show route protocol ospf

inet.0: 1898 destinations, 2009 routes (1898 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.15.15.0/24      *[OSPF/10] 00:01:14, metric 2, tag 0
                    > to 10.81.40.17 via so-1/0/0.0
10.15.16.0/24      *[OSPF/10] 00:01:14, metric 2, tag 0
                    > to 10.81.40.17 via so-1/0/0.0
10.15.17.0/24      *[OSPF/10] 00:01:14, metric 2, tag 0
                    > to 10.81.40.17 via so-1/0/0.0
10.81.40.14/31     *[OSPF/10] 00:21:54, metric 2, tag 0
                    > to 10.81.40.17 via so-1/0/0.0
224.0.0.5/32       *[OSPF/10] 00:14:22, metric 1
                      MultiRecv
```

JUNOS BGP

- JUNOS soporta las típicas especificaciones de BGP:
  - Route refresh
  - Multiprotocol extensions
  - Authentication
  - Communities
  - Route damping
  - Graceful restart
  - Route reflection
  - Confederations
  - Otros

Configuraciòn de BGP

- Acá es donde empezamos a ver diferencias con IOS
- Por ejemplo, en la asignaciòn del AS
  - JUNOS aplica el AS en la sección “routing-options”
    - Permite que múltiples procesos usen el mismo AS (típico cuando se configuran VPNs)
  - IOS codifica el AS en el comando “`router bgp`”

JUNOS

```
routing-options {
    autonomous-system 65000;
}
```

IOS

```
router bgp 65000
```

¿ cómo se configuró?

```
set routing-options autonomous-system 65000
```

Configuraciòn de BGP

- **Peer groups**
  - **JUNOS hace todo con peer groups**
  - **En IOS es opcional, pero se lo recomienda enfáticamente**
- **El número de AS con el que vamos a hacer vecindad**
  - **Puede ser definido para todo el grupo o solo para un vecino**

JUNOS

```
bgp {
    group EXTERNAL-PEERS {
        peer-as 65001;
        neighbor 10.81.254.1;
        neighbor 10.81.254.2 {
            peer-as 65002;
        }
    }
}
```

IOS

```
router bgp 65000
 neighbor 10.81.254.1 remote-as 65001
 neighbor 10.81.254.2 remote-as 65002
```

JUNOS

```
# edit protocols bgp  <ENTER>
# set group EXTERNAL-PEERS peer-as 65001
# set group EXTERNAL-PEERS  neighbor 10.81.254.1
# set group EXTERNAL-PEERS neighbor 10.81.254.2 peer-as 65002
```

Configuración de BGP

- **Synchronization**
  - JUNOS usa “no sync” por default
  - IOS usa (o al menos usaba) “sync”por default

- **JUNOS por default “no sumariza”, Cisco si**

JUNOS

```
bgp {
   group EXTERNAL-PEERS {
      peer-as 65001;
      neighbor 10.81.254.1;
      neighbor 10.81.254.2 {
         peer-as 65002;
      }
   }
}
```

IOS

```
router bgp 65000
 no synchronization
 neighbor 10.81.254.1 remote-as 65001
 neighbor 10.81.254.2 remote-as 65002
 no auto-summary
```

Configuración BGP

- **El comando "type" le dice al router cómo se debe interactuar con el vecino BGP**
  - **external (el default si no se incluye el comando) o internal**

JUNOS

```
bgp {
   group EXTERNAL-PEERS {
      type external;
      peer-as 65001;
      neighbor 10.81.254.1;
      neighbor 10.81.254.2 {
         peer-as 65002;
      }
   }
   group INTERNAL-PEERS {
      type internal;
      neighbor 10.81.254.3;
   }
}
```

IOS

```
router bgp 65000
 no synchronization
 neighbor 10.81.254.1 remote-as 65001
 neighbor 10.81.254.2 remote-as 65002
 neighbor 10.81.254.3 remote-as 65000
 no auto-summary
```

Configuración BGP

- `local-address` cambia la IP de origen de la sesiòn BGP
  - Idéntico al comando "`update-source`" de IOS

JUNOS

```
bgp {
    group EXTERNAL-PEERS {
        type external;
        peer-as 65001;
        neighbor 10.81.254.1;
    }
    group INTERNAL-PEERS {
        type internal;
        local-address 10.81.254.10;
        neighbor 10.81.254.3;
    }
}
```

IOS

```
router bgp 65000
 no synchronization
 neighbor 10.81.254.1 remote-as 65001
 neighbor 10.81.254.3 remote-as 65000
 neighbor 10.81.254.3 update-source Loopback0
 no auto-summary
```

“show bgp ...”

```
user@host> show bgp ?
Possible completions:
  group                        Show the BGP group database
  neighbor                     Show the BGP neighbor database
  summary                      Show an overview of the BGP information
```

show bgp summary

user@host> show bgp summary

Groups: 1 Peers: 2 Down peers: 0

Table	Tot Paths	Act Paths	Suppressed	History	Damp State	Pending
inet.0	12	12	0	0	0	0

Peer	AS	InPkt	OutPkt	OutQ	Flaps	Last Up/Dwn	State
10.81.254.1	65001	423	430	0	0	3:33:00	4/4/0
10.81.254.2	65002	428	430	0	0	3:32:56	4/4/0

IOS-RTR-1#sh ip bgp summary

BGP router identifier 10.81.254.10, local AS number 65000

BGP table version is 1, main routing table version 1

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
10.81.254.1	4	65001	5	22	1	0	0	00:01:09	4
10.81.254.2	4	65002	9	22	1	0	0	00:03:02	4

show bgp neighbor

user@host> show bgp neighbor

```
Peer: 10.81.254.1+179  AS 65001  Local: 10.81.254.10+1028 AS 65000
  Type: External    State: Established    Flags: <>
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Options: <Preference HoldTime PeerAS Refresh>
  Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 10.81.254.1      Local ID: 10.81.254.10      Active Holdtime: 90
  Keepalive Interval: 30
  Local Interface: so-0/0/1.0
  NLRI advertised by peer: inet-unicast
  NLRI for this session: inet-unicast
  Peer supports Refresh capability (2)
  Table inet.0 Bit: 10000
    Send state: in sync
    Active prefixes: 4
    Received prefixes: 4
    Suppressed due to damping: 0
  Last traffic (seconds): Received 13   Sent 13   Checked 13
  Input messages:  Total 438    Updates 4       Refreshes 0     Octets 8473
  Output messages: Total 440    Updates 4       Refreshes 0     Octets 8526
  Output Queue[0]: 0
```

¿Rutas que BGP anuncia a un vecino?

- Todas las rutas en JUNOS estàn en la tabla de rutas
  - Se usa `show route advertising-protocol bgp <peer>` para ver las rutas que se le anuncian a un vecino
  - Similar a "`show ip bgp neigh <peer> advertise`" de IOS
  - Nos permite ver el efecto de nuestras polìticas de rutas con un vecino
    - Excepción: la acciòn de hacer "prepend"

user@host> show route advertising-protocol bgp 10.81.254.1

inet.0: 21 destinations, 22 routes (21 active, 0 holddown, 0 hidden)

+ = Active Route, - = Last Active, * = Both

10.20.3.0/24

Self 0 100 I

10.20.4.0/24

Self 0 100 I

¿ rutas que recibimos de un vecino?

- Usamos "`show route receive-protocol bgp <peer>`" para ver las rutas que nos envía un vecino
- Similar a "`show ip bgp neigh <peer> received-routes`" de IOS
- Nos muestra las rutas antes de que apliquemos filtros de input
  - Excepto para rutas rechazadas con un "route-filter"

user@host> show route receive-protocol bgp 10.81.254.1

inet.0: 26 destinations, 27 routes (26 active, 0 holddown, 0 hidden)

+ = Active Route, - = Last Active, * = Both

10.20.1.0/24

10.81.40.15 0 100 I

10.20.2.0/24

10.81.40.15 0 100 I

Rutas que BGP aporta a la tabla IP

- Usamos "`show route protocol bgp`" para verlo
  - Nos permite ràpidamente ver algunos atributos
    - Local Preference, AS Path, Origin, MED
    - Usamos `detail` o `extensive` para tener màs detalle
  - Similar a "`show ip bgp`" de IOS

IOS y JUNOS

user@host> show route protocol bgp

inet.0: 26 destinations, 27 routes (26 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.20.1.0/24 *[BGP/170] 00:23:21, MED 0, localpref 100
AS path: 65001 I
> to 10.81.40.15 via so-0/0/0.0
10.20.2.0/24 *[BGP/170] 00:23:21, MED 0, localpref 100
AS path: 65001 I
> to 10.81.40.15 via so-0/0/0.0

IOS-RTR-1#sh ip bgp

BGP table version is 2003, local router ID is 10.81.254.10
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

Network	Next Hop	Metric	LocPrf	Weight	Path
>i10.20.3.0/24	10.81.254.20		100	0	i
>i10.20.4.0/24	10.81.254.20		100	0	i

Informaciòn detallada de una ruta BGP

user@host> show route 10.20.3/24 detail

inet.0: 26 destinations, 27 routes (26 active, 0 holddown, 0 hidden)

10.20.3.0/24 (1 entry, 1 announced)

*BGP Preference: 170/-101

Source: 10.81.254.20

Nexthop: via ge-0/1/0.0, selected

Protocol Nexthop: 10.81.254.20 Indirect nexthop: 8458088 44

State: <Active Int Ext>

Local AS: 65000 Peer AS: 65000

Age: 2:39:44 Metric: 0 Metric2: 1

Task: BGP_20.10.81.254.20+1127

Announcement bits (3): 0-KRT 3-BGP.0.0.0.0+179 4-Resolve inet.0

AS path: I

Localpref: 100

Router ID: 10.81.254.20

“Tracing” de BGP

- **Podemos activar el debugging de actividad de BGP:**

```
[edit protocols bgp]
user@host# show
traceoptions {
    file bgp-trace;
    flag open detail;
    flag update detail;
}
```

Vemos lo que se graba en el archivo *`bgp-trace`* usando

- **el comando operacional “`monitor start` *`log-file-name`* “**
- **o el comando operacional “`show log` *`log-file-name`*”**

JUNOS Workshop

Routing Policies

Policies: concepto

- Las “policies” permiten controlar cómo la información de rutas fluye desde o hacia la tabla de rutas
  - Permite ignorar o cambiar informaciòn asociada a las rutas que nos anunciar
  - Puede eliminar o cambiar informaciòn de las rutas que nosotros anunciamos

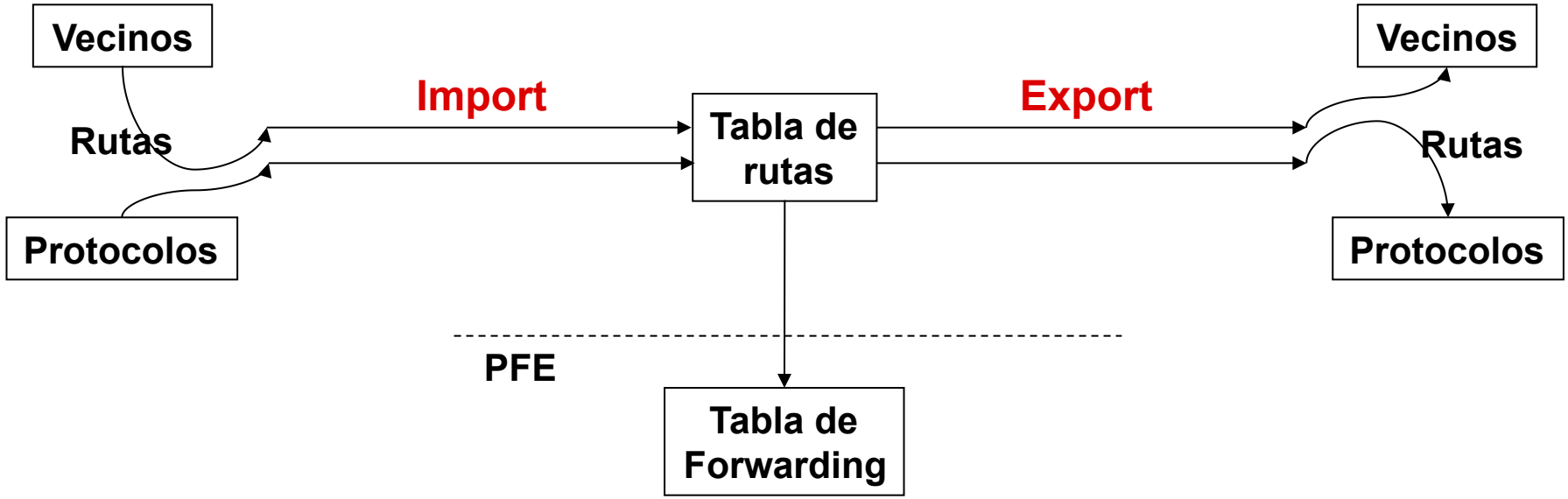
- Estas “policies” están compuestas por pares del tipo “match/action”
  - En las secciones “match” y “action” uno juega con los atributos específicos de cada protocolo con el que interactùa

Cuando usar policies

- Cuando...
  - uno no desea que la tabla de rutas herede ciertos tipos de rutas que nos envían nuestros vecinos
  - no queremos anunciar ciertos tipos de rutas a nuestros vecinos
  - queremos que un protocolo herede rutas aprendidas por otro protocolo
  - deseamos modificar atributos de una o màs rutas

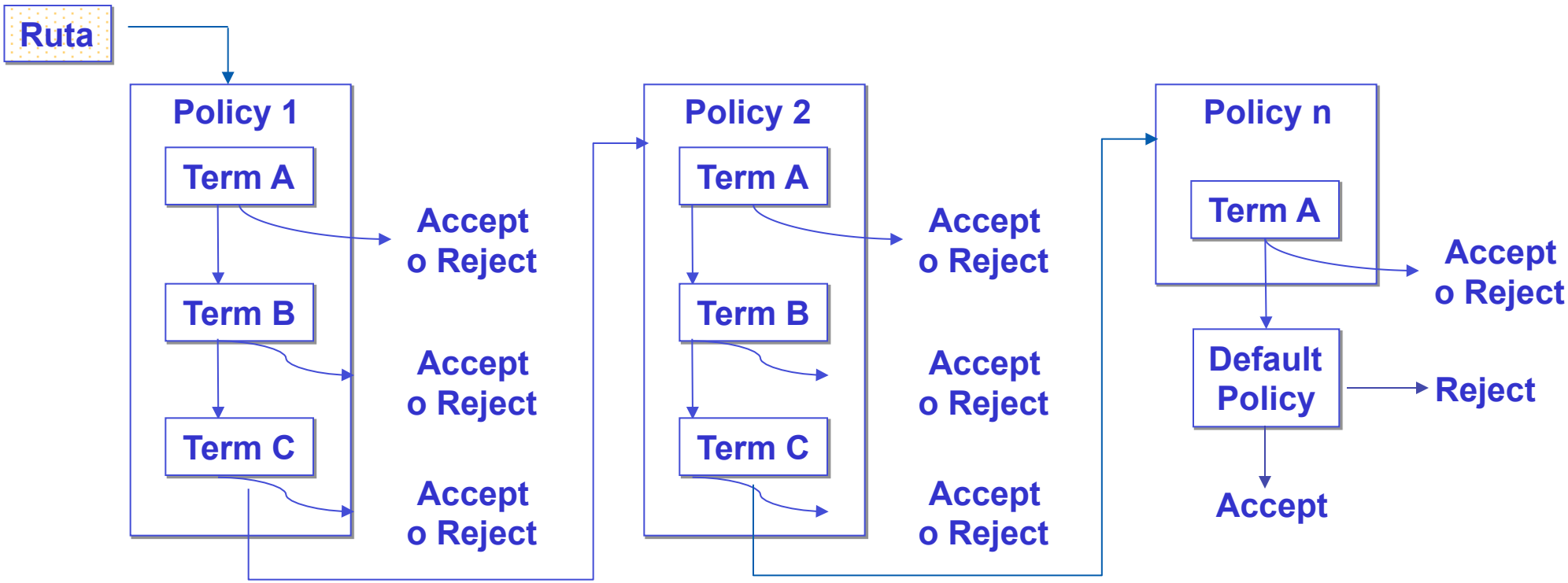
Policies de tipo “import” o “export”

- Se ejecutan con respecto a la tabla de rutas de JUNOS
  - JUNOS aplica una “import policy” sobre un conjunto de rutas antes de ponerlas en la tabla de rutas
  - JUNOS aplica una “export policy” solamente a rutas ACTIVAS en la tabla de rutasl



Flujo de una o màs policies

- Se pueden “encadenar” policies
- Se evalúa hasta que se encuentra una “*terminating* action”
  - *Terminating actions*: “accept” o “reject”
- Cada policy es una lista de “tèrminos”
  - Se puede saltar a otra policy usando el comando “`next-policy`”



Sintaxis de una policy

```
policy-options {
    policy-statement policy-name {
        term term-name {
            from {
                match-conditions;
            }
            then {
                action;
            }
        }
    }
}
```

Una policy puede tener uno o màs tèrminos

Condiciones de matching en un tèrmino

- Hay variados criterios de matching disponibles dentro del tèrmino de una policy
- Posibles opciones (hay màs):
  - Dirección IP de un vecino
  - Protocolo (fuente de información)
    - BGP, direct, DVMRP, IS-IS, local, MPLS, OSPF, PIM, RIP, static, aggregate
  - Información que nos provee un protocolo
    - OSPF area ID
    - IS-IS level number
    - Atributos BGP
  - Condiciones basadas en expresiones regulares para paths o comunidades de BGP

Acciones ejecutables en un tèrmino

- Se pueden tomar las siguientes acciones:
  - "Terminating actions"
    - "Accept" de la ruta
    - "Reject" de la ruta
  - Acciones de control de flujo
    - "Skip" a otra policy
    - "Skip" a otro término
  - Acciones que modifican atributos de una ruta
    - Metric
    - Preference
    - Color
    - Next-hop address

Policies por default

- A cada protocolo JUNOS le aplica una policy por default; esta policy està implícita al final de una cadena de policies
  - Puede modificarse por uso del comando "`default-action`"
- IS-IS, OSPF, RIP
  - Importar todas las rutas
  - Exportar todas las rutas que el protocolo aprendiò
    - RIP es una excepción; por default no anuncia rutas y se debe forzar el anuncio
- BGP
  - Importar todas las rutas que aprende de los vecinos
  - Exportar a otros vecinos BGP todas las rutas aprendidas desde vecinos BGP
    - Las rutas aprendidas por eBGP se exportan a todos los vecinos BGP
    - Las rutas aprendidas por iBGP se exportan a todos los vecinos eBGP (se asume que existe una mesh de iBGP dentro del AS)

Un ejemplo

- **Se escriben las policies en la sección `[edit policy-options]`**

[edit policy-options]
user@host# **show policy-statement** ***advertise-ospf***
term pick-ospf {
    from protocol ospf;
    then accept;
}

- **Se aplica esa policy en modo `import` o `export (o ambos)`**

[edit protocols bgp]
user@host# **set export** ***advertise-ospf***

Otro ejemplo

Si se usan varias condiciones en la sección "`from`", significa que se busca que todas las condiciones se cumplan

```
[edit]
user@host# show policy-options
policy-statement isis-level2 {
    term find-level2-routes {
        from {
            protocol isis;
            level 2;
        }
        then accept;
    }
}
```

AND lógico

Ejemplo de aplicaciòn de una policy

```
[edit protocols]
user@host# show
bgp {
    import bgp-import;
    export bgp-export;
}
```

BGP y las policies

- En BGP tenemos tres jerarquías disponibles para aplicar policies (tanto en input como en output):
  - Global a nivel de todo el proceso BGP
  - Grupos de vecinos
  - Vecinos individuales
- Solamente la policy màs especìfica es la que se aplica a un vecino en particular
  - Una “neighbor policy” hace que se ignore para ese vecino las policies grupales o globales
  - Una policy para un grupo hace que se ignore la policy global de BGP

Policies y BGP: ejemplo

```
[edit protocols]
user@host# show
bgp {
    export local-customers;
    group meganet-inc {
        type external;
        import [ martian-filter long-prefix-filter as-47-filter ];
        peer-as 47;
        neighbor 1.2.2.4;
        neighbor 1.2.2.5;
    }
    group problem-child {
        type external;
        import [ as-47-filter long-prefix-filter martian-filter ];
        export kill-private-addresses;
        peer-as 54;
        neighbor 1.2.2.6;
        neighbor 1.2.2.7;
        neighbor 1.2.2.8 {
            import [ reject-unwanted as-666-routes ];
        }
    }
}
```

Route Filters

- Un “route filter” busca el matching de una ruta o un grupo de rutas
  - En un término se pueden poner varios “route filters”
  - La sintaxis es la siguiente:

```
route-filter prefix/prefix-length match-type actions
```

- Hay ciertas reglas en función del tipo de match
  - Los “tipos” de match son los siguientes:
    - `exact`
    - `orlonger`
    - `longer`
    - `upto`
    - `through`
    - `prefix-length-range`

Route Filter; tipos de match (1 de 2)

- **`exact`**
  - **Matching exacto en el prefijo y la màscara**
  - **No se incluirà ninguna otra ruta**

    from route-filter 192.168/16 exact;
- **`orlonger`**
  - **Matching en exactamente el prefijo y la màscara**
  - **También en rutas que tienen el mismo prefijo pero màscaras más grandes**

    from route-filter 192.168/16 orlonger;
- **`longer`**
  - **No acepta ESE prefijo y su màscara en forma exacta**
  - **Acepta solo las rutas que comienzan con ESE prefijo pero con máscaras màs grandes**

    from route-filter 192.168/16 longer;

Route Filter; tipos de match (2 de 2)

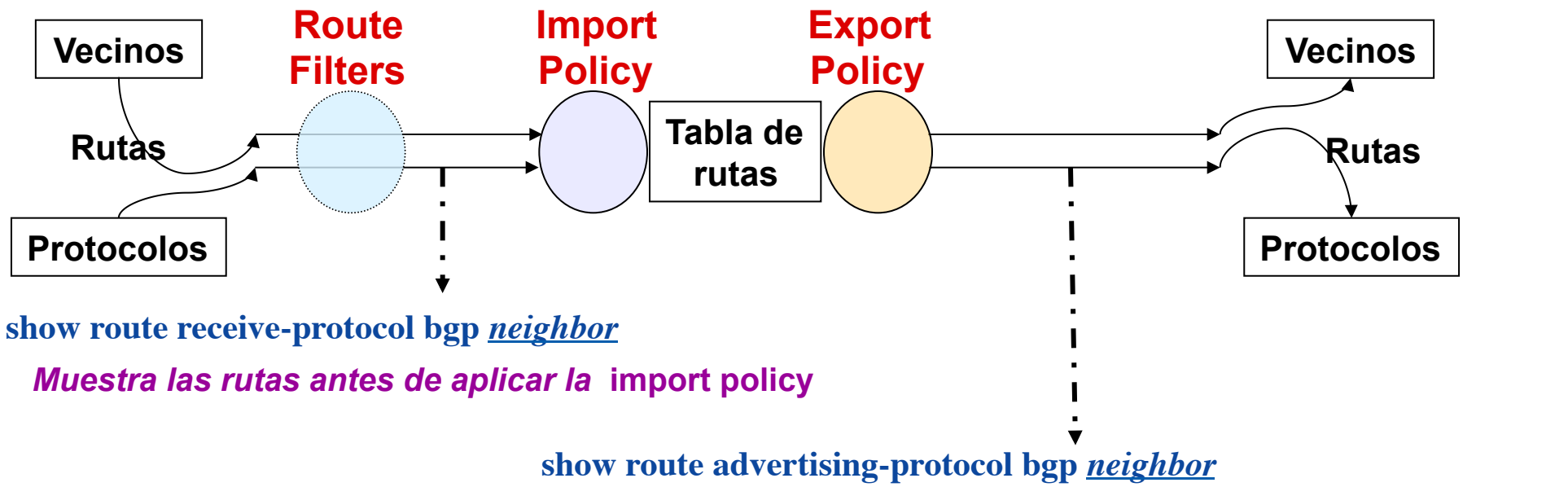
- `upto`
  - Matching exacto en prefijo y màscara
  - También en el mismo prefijo si la màscara no es mayor al segundo valor especificado

    from route-filter 192.168/16 upto /24;
- `through`
  - Match exacto en el primer prefijo y màscara especificado
  - Match exacto en el segundo prefijo y màscara especificado
  - Match de todos los prefijos encuadrados entre ambos prefijos

    from route-filter 192.168/16 through 192.168.16/20;
- `prefix-length-range`
  - Match solo en las rutas que comienzan con el mismo prefijo y que tienen su máscara entre los valores indicados

    from route-filter 192.168/16 prefix-length-range /20-/24;

Control de cómo funcionan las policies



- `show route receive-protocol` y `show route advertising-protocol` :
  - Muestran los anuncios ANTES de aplicar la policy de import y LUEGO de haber aplicado la policy de export, respectivamente
    - La excepciòn es el uso de route-filters en la policy de import
- Pregunta: ¿Cómo se pueden "ver" los efectos de la import policy?
  - Usen "show route protocol *protocol* "

JUNOS Workshop

Firewall Filters

Qué son los Firewall Filters?

- **Nos permiten:**
  - **Hacer matching de paquetes en modo “stateless” (sentencia “*from*”)**
  - **Tomar acciones sobre esos paquetes (sentencia *then*)**
- **Son siempre *compilados* para garantizar buen rendimiento**
- **Se ejecutan en bajo nivel (PFE)**
- **Son lo que en IOS se conocen como ACLs (Access Control Lists)**

Términos de un firewall filter

- **Contienen sentencias del tipo *if…then (“from”..”then”)***
  - ***from* = el matching**
  - **El *then* describe las acciones que se ejecutan cuando se logran el matching**
  - **Ejemplo:**

```
term example-term {
    from {
        source-address {
            172.17.38.4/32;
        }
        destination-address {
            172.17.37.4/32;
            172.17.47.52/32;
            172.17.17.17/32;
        }
        protocol [ tcp udp ];
    }
    then accept;
}
```

Cómo se evalúan los filtros

- Filtros con un sólo tèrmino
  - Si el paquete hace matching en todas las condiciones, se ejecuta lo que dice el `then`
  - Si no cumple todas las condiciones, se lo descarta
- Filtros con varios términos (evaluados secuencialmente)
  - JUNOS busca el primer tèrmino que haga matching:
    - En aquel tèrmino en que encuentra matching, ejecuta el “then”
    - Si ningún término tuvo matching, se lo descarta
- “Filter Terms”
  - A filtro puede referenciar a otro filtro

Qué puede hacer matching?

- Lo que uno usualmente puede describir en los componentes de un paquete

Condiciones por rangos

- En campos que pueden ser descriptos por un nùmero
  - Ejemplo: Ports y nùmeros de protocolos
- Ejemplos
  - `source-port 1024-65535;`
  - `source-port ssh;`
  - `source-port [ telnet smtp ];`
- Claves para identificar campos:
  - `destination-port`, `dscp`, `esp-spi`, `forwarding-class`, `fragment-offset`, `icmp-code`, `icmp-type`, `interface-group`, `ip-options`, `packet-length`, `port`, `precedence`, `protocol`, `source-port`, `*-except`

Condiciones de match por direcciones

- IP source / destination
- Longest match
  - Se puede usar la clave `except` para usar condiciones negativas
- Keywords disponibles:
  - `address`
  - `source-address`
  - `destination-address`
  - `prefix-list`
  - `source-prefix-list`
  - `destination-prefix-list`

Ejemplo

```
[edit]
lab@London# show firewall
family inet {
    filter reject-some-addresses {
        term unused-private-addresses {
            from {
                source-address {
                    10.0.0.0/8;
                    10.14.10.0/24 except;
                    172.16.0.0/12;
                    192.168.0.0/16;
                    192.168.200.0/24 except;
                }
            }
            then {
                reject;
            }
        }
    }
}
```

Ejemplo

```
[edit]
lab@London# show policy-options
prefix-list internal-routes {
    10.14.10.0/24;
    192.168.200.0/24;
}
prefix-list rfc1918-addresses {
    10.0.0.0/8;
    172.16.0.0/12;
    192.168.0.0/16;
}
[edit]
lab@London# show firewall
family inet {
    filter reject-some-addresses {
        term unused-private-addresses {
            from {
                source-prefix-list {
                    rfc1918-addresses;
                    internal-routes except;
                }
            }
            then {
                reject;
            }
        }
    }
}
```

Filtros en direcciones no contiguas

- Similar al "*wildcard mask*" de otros sistemas
- Su notación es de la forma "address/mask"¡
  - "mask" NO es una "wildcard mask" — Un 1 indica que el bit debe hacer match, 0 que el bit no tiene que hacerlo
  - Aún es posible usar la clave "`except`"
  - Ejemplos:

```
destination-address {
    0.16.0.0/0.255.0.0;
    172.16.0.1/255.255.0.255;
}
destination-address {
    172.16.0.0/12;
    172.16.0.1/255.240.1.255 except;
}
```

Condiciones de match en bits

- Se puede hacer match en ciertos bits de ciertos campos del paquete
- Casos: `ip-options`, `fragment-flags`, y `tcp-flags`
  - Note: La especificacipòn de un campo de bits no implica un cierto protocolo
- Grupos `(`paréntesis`)`, negación (`!`), y soporte de funciones AND (`&` o +) y OR (`|` o `,`)
  - Ejemplo: `tcp-flags` “`(syn & !ack) | rst`” hace matching en cualquier paquete que es el paquete inicial de una sesiòn TCP o un TCP reset

Sinónimos de texto

- Usamos sinónimos de texto para matchings comunes
  - `first-fragment`: Primer fragmento de un paquete fragmentado (sinónimo de `fragment-offset 0;` y `fragment-flags more-fragments;`)
  - `is-fragment`: Fragmentos que no son el primero (sinónimo de `fragment-offset-except 0;`)
  - `tcp-established`: Sinónimo para `tcp-flags “(ack | rst)”;`
  - `tcp-initial`: Sinónimo para `tcp-flags “(syn & !ack)”;`

Acciones tradicionales

- **Acciones tradicionales en los firewall filters:**
  - **“Terminating actions”:**
    - `accept`
    - `discard`
    - `reject`
  - **“Flow control”:**
    - `next term`
  - **Modificadores:**
    - `count`, `log`, `syslog`
    - `forwarding-class`, `loss-priority`
    - `policer`
    - `sample`

Conteo y logging

- **Contamos paquetes con el modificador** `count` ***counter***
  - **Comandos** `show` **asociados:**
    `show firewall filter` ***filter***
    `show firewall filter` ***filter*** `counter` ***counter***

- **Hacemos log con el modificador** `log`
  - **Comando "show" asociado:** `show firewall log`

lab@router> **show firewall log**
Log :

Time	Filter	Action	Interface	Protocol	Src Addr	Dest Addr
22:34:02	classify-pak	A	local	UDP	172.17.37.4	172.17.38.4
21:57:53	classify-pak	A	local	ICMP	172.17.37.4	172.17.38.4
21:57:47	pfe	A	fe-2/0/1.11	GRE	172.17.37.4	172.17.38.4
21:57:46	pfe	A	fe-2/0/1.11	GRE	172.17.37.4	172.17.38.4
21:57:45	pfe	A	fe-2/0/1.11	GRE	172.17.37.4	172.17.38.4
21:57:17	classify-pak	A	local	ICMP	172.17.37.4	172.17.38.4
21:57:16	classify-pak	A	local	ICMP	172.17.37.4	172.17.38.4

Policers

- Se puede usar un “policer” dentro de un filtro para ponerle un lìmite al ritmo de ciertos paquetes
  - El policer tambièn se lo puede aplicar directamente a una interfaz para limitar el ritmo de paquetes de toda una familia de protocolo
- Tráfico que cae en el filtro se lo regula segùn el bandwidth configurado
  - El bandwidth puede ser especificado como porcentaje de la velocidad de la interfaz
- Cuando el ritmo del tràfico excede los paràmetros de velocidad, se pueden especificar acciones o modificadores

Ejemplo de policing

```
[edit firewall]
lab@router# show
policer p1 {
    if-exceeding {
        bandwidth-limit 400k;
        burst-size-limit 100k;
    }
    then discard;
}
family inet {
    filter limit-ftp {
        term ftp {
            from {
                source-address {
                    1.2.3.0/24;
                }
                protocol tcp;
                destination-port [ ftp ftp-data ];
            }
            then {
                policer p1;
                count count-ftp;
            }
        }
    }
}
```

- **Ejemplo:**
  - **bandwidth-limit**
    - **bits per second**
    - **30,520 bps a 4.29 Gbps**
  - **burst-size-limit**
    - **en bytes**
    - **Minimo debería ser 10 veces la MTU (en bajas velocidades) o bandwidth multiplicado 3–5 milisegundos (en altas velocidades)**

Policers en interfaces

```
interfaces {
    interface-name {
        unit logical-unit-number {
            family inet {
                     filter {
                    input filter-name;
                    output filter-name;
                }
                policer {
                    input policer-template;
                    output policer-template;
                }
            }
        }
    }
}
```

Aplicabilibad del firewall filter

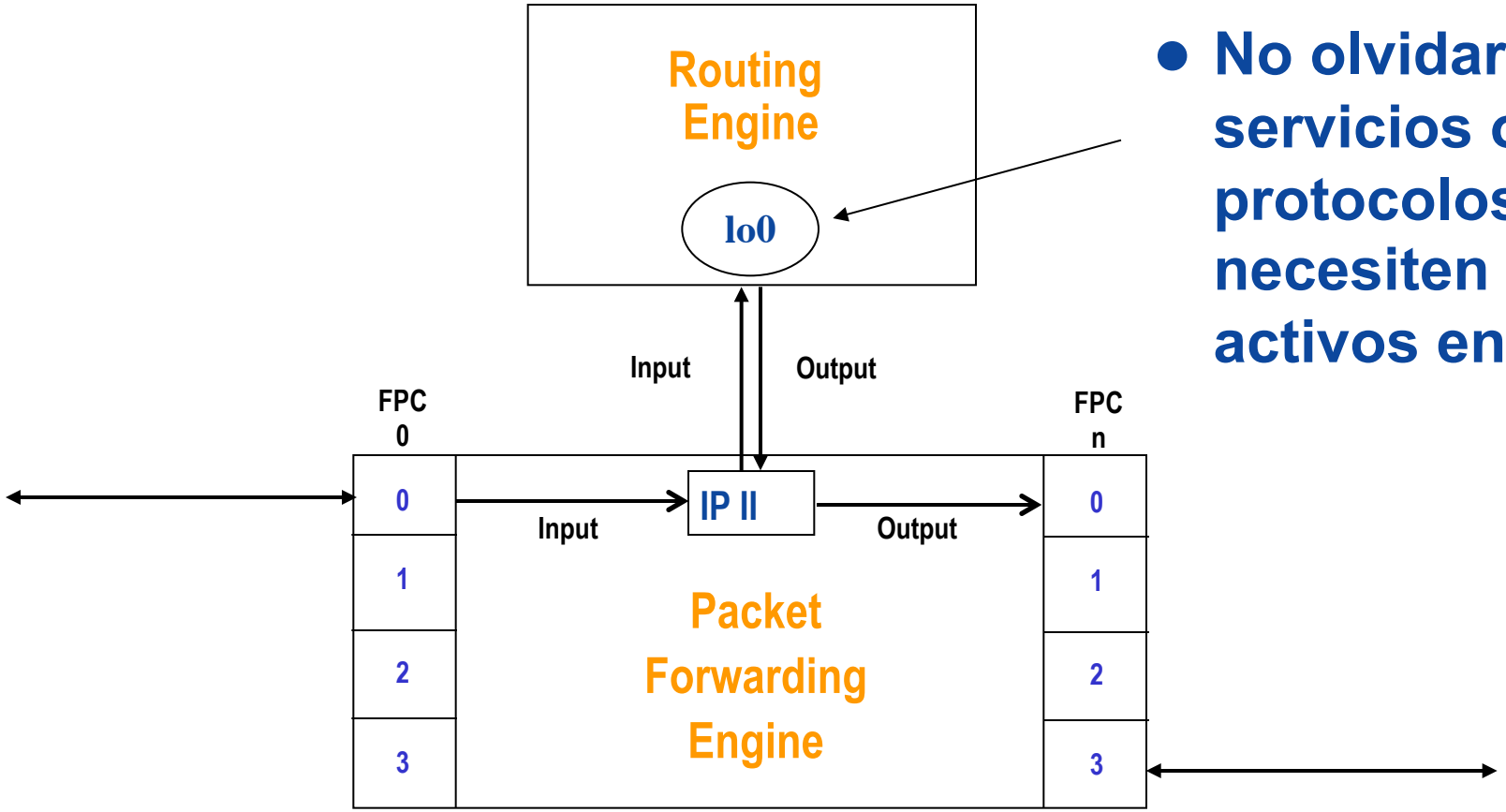
- Se aplican a interfaces
  - En interfaces lógicas: en input y/o output de la interfaz
  - `lo0`: Aplica a tràfico entrante o saliente a la Routing Engine
- Se pueden encadenar Firewall Filters
  - Funcionan de la misma manera que una cadena de policies explicada anteriormente

Ejemplo

```
interfaces {
    fe-0/0/0 {
        unit 0 {
            family inet {
                filter {
                    input filter-in;
                    output-list [ filter1 filter2 ];
                }
            }
        }
    }
}
firewall {
    family inet {
        filter filter-in {
            term block-some-packets {
                from {
                    source-address {
                        10.10.10.0/24;
                    }
                }
                then {
                    discard;
                }
            }
            term accept-others {
                then {
                    count input-packets;
                    accept;
                }
            }
        }
[...]
```

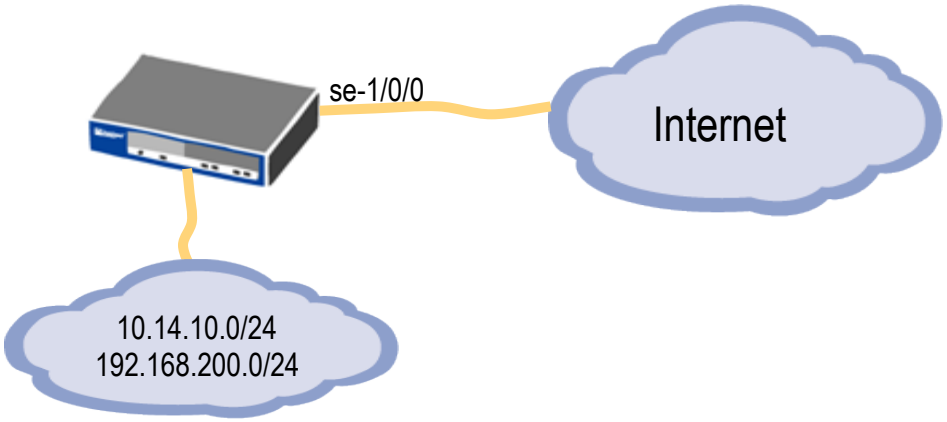
```
[...]
    filter filter1 {
      term count-packets {
        then {
          count output-packets;
          next term;
        }
      }
    }
    filter filter2 {
      term block-some-packets {
        from {
          destination-address {
            10.10.10.0/24;
          }
        }
        then {
          discard;
        }
      }
      term accept-others {
        then {
          accept;
        }
      }
    }
  }
}
```

Filtrado de tràfico “local”



- filtro en interfaz lo0
- No olvidarse de servicios o protocolos que se necesiten tener activos en el equipo

Ejemplo




```
interfaces {
    se-1/0/0 {
        description “Connection to ISP”;
        unit 0 {
            family inet {
                filter {
                    input anti-spoof-in;
                    output anti-spoof-out;
                }
            }
        }
    }
}
policy-options {
    prefix-list internal-routes {
        10.14.10.0/24;
        192.168.200.0/24;
    }
}
[...]
```

```
firewall {
   family inet {
      filter anti-spoof-in {
         term deny-internal-addresses {
            from {
               source-prefix-list {
                  internal-routes;
               }
            }
            then {
               count spoofed-in;
               reject tcp-reset;
            }
         }
         term allow-all-others {
            then accept;
         }
      }
      filter anti-spoof-out {
         term permit-internal-addresses {
            from {
               source-prefix-list {
                  internal-routes;
               }
            }
            then accept;
         }
         term reject-all-others {
            then {
               count spoofed-out;
               reject tcp-reset;
            }
         }
      }
   }
}
```

Ejemplo

```
interfaces {
    lo0 {
        unit 0 {
            family inet {
                filter {
                    input protect-re;
                }
            }
        }
    }
}
policy-options {
    prefix-list internal-routes {
        192.168.0.0/16;
    }
    prefix-list bgp-peers {
        apply-path “protocols bgp group <*> neighbor <*>”;
    }
}
firewall {
    family inet {
        filter protect-re {
            term allow-telnet-ssh {
                from {
                    source-prefix-list {
                        internal-routes;
                    }
                    protocol tcp;
                    port [ ssh telnet ];
                }
                then accept;
            }
            term allow-bgp {
                from {
                    source-prefix-list {
                        bgp-peers;
                    }
                    protocol tcp;
                    port 179;
                }
                then accept;
            }
            term allow-ospf {
                from protocol ospf;
                then accept;
            }
            term permit-ping {
                from protocol icmp;
                then accept;
            }
        }
    }
}
```

Juniper™
NETWORKS