Practical use of RPSL and IRR tools

15th APNIC Open Policy Meeting
Taipei, Taiwan

Andy Linton <asjl@lpnz.org>

Goals of the tutorial

- To make the life of LIRs and ISPs easier by:
  - familiarising LIRs and ISPs with the features of Routing Registry (RR)
  - introducing tools & services provided by APNIC
- To promote usage of the RR
- A chance for practical exercise

- NOT to teach the basics of routing
- NOT to explain how to obtain Internet resources (IP & ASN)
- NOT to help decisions on network setup

Assumptions

- The audience
  - Knowledgeable about BGP routing
  - Familiar with LIR terms & procedures
  - Familiar with basic RIPE DB operations
  - Curious about Routing Registry usage
- The course does not give everything
  - Gives: Introduction, examples, references
  - Is NOT a replacement for hands-on experience!
- Questions anytime!

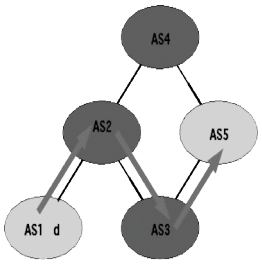
Agenda

- Routing Policy
  - What is Routing Policy?
  - Why define one?
- RPSL
  - What is RPSL?
  - RR as a part of the APNIC whois DB & IRR
  - Specifying Routing Policies Using RPSL & Configuring Routers Using RtConfig
  - IRRToolSet
- Summary, discussion, evaluation & homework

What is a Routing Policy?

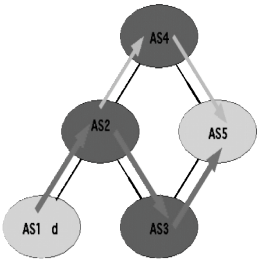
- Public description of the relationship between external BGP peers:
  - Who are my BGP peers?
  - What routes are
    - Originated by a peer
    - Imported from each peer
    - Exported to each peer
    - Preferred when multiple routes exist
  - What to do if no route exists
- Can also describe internal BGP peer relationship

Routing Policy Example



- AS1 originates prefix "d"
- AS1 exports "d" to AS2, AS2 imports
- AS2 exports "d" to AS3, AS3 imports
- AS3 exports "d" to AS5, AS5 imports

Routing Policy Example (cont)



- AS5 also imports "d" from AS4
- Which route does it prefer?
  - Does it matter?
  - Consider case where
    - AS3 = Commercial Internet
    - AS4 = Internet2

Why define a Routing Policy?

- Documentation
- Provides a debugging aid
  - Compare policy versus reality
- Consistency across your AS
  - routers / implementations
- Provides routing security
  - Can peer originate the route?
  - Can peer act as transit for the route?
- Scalability
  - allows automatic generation of router configurations

What is RPSL?

- Object oriented language
- Development of RIPE 181
- Structured whois objects
- Describes things interesting to routing policy:
  - Routes
  - AS Numbers
  - Relationships between BGP peers
  - Management responsibility

Why use RR to store your policies?

- Consistent configuration between BGP peers (peers & customers & upstreams)
- Expertise encoded in the tools that generate the policy rather than engineer configuring peering session
- Automated, manageable solution for filter generation / router configuration
- Provides a debugging aid
  - Compare reality versus policy

Exercise: Determining Routing Policy

- Who are my BGP neighbours?
  - (customers/ peers/ upstreams)
- What routes are:
  - Originated by each neighbour?
  - Imported from each neighbour?
  - Exported to each neighbour?
  - Preferred when multiple routes exist?
  - How are they treated (modified routing parameters?)
- What to do if no route exists?

What is the Routing Registry and why should I Use it?

- Policy based routing
  - Allows different criteria as basis for routing decisions
- Routing policy - description of the relationship between external BGP peers
- Next level of abstraction: RPSL
- RR & Existing tools
- Ultimately: easier maintenance of routing configuration in big & complex networks
- See http://www.apnic.net/services/apnic-rr/rr-benefits.html

Real-life examples of RR usefulness

- connect.com.au
  - `whois –h whois.ripe.net -s RADB -r -T aut-num AS2764`
- C&W, running private RR for their customers
- Some AS numbers with detailed policy:
- `whois –h whois.ripe.net -r -T aut-num AS286 (KPN Eurorings)`
- `whois –h whois.ripe.net -r -T aut-num AS5400 (BT)`
- `whois –h whois.ripe.net -r -T aut-num as1299 (Telia)`

APNIC Database & the Internet Routing Registry

- Public Network Management Database
  - "whois" info about networks & contact persons
- Routing Registry contains routing information
  - Using RPSL
- APNIC RR is part of the IRR:
  - http://www.apnic.net/services/apnic-rr-guide.html
  - Distributed databases that mirror each other
    - Enough to register your objects and policy in one
  - IRR = RIPE + RADB + APNIC + ARIN + …

Use of RPSL

- Use RtConfig v4 (part of IRRToolSet from RIPE) to generate filters based on information stored in our routing registry
  - Avoid filter errors (typos)
  - Filters consistent with documented policy (need to get policy correct though)
  - Engineers don't need to understand filter rules (it just works :-)

RtConfig

- Part of the IRRToolSet
- Generates router configuration based on the RR
  - Cisco, Bay's BCC, Juniper's Junos and Gated/RSd
- Creates route-map and AS path filters
- Can also create ingress / egress filters
  - (documentation says Cisco only)

RtConfig: Command-line Usage

- Environment variables
  - IRR_HOST=whois.apnic.net
  - IRR_PORT=43
  - IRR_SOURCES=APNIC
  - Must specify: -protocol ripe
- Overridden by command line options
  - # RtConfig -h localhost -p 43 -s RRTEST -protocol ripe

RtConfig –protocol ripe
>RtConfig @RtConfig [command]

RtConfig Example: Creating Access Lists

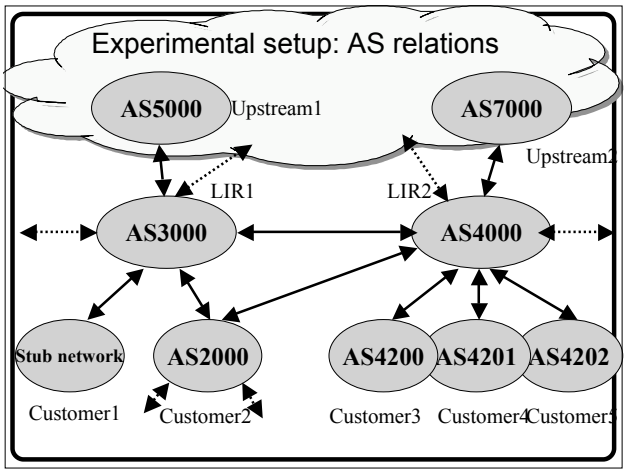
route:	10.4.192.0/19
origin:	AS4000
[…]	

```
$ RtConfig –protocol ripe
RtConfig> @RtConfig access_list filter AS4000
!
no access-list 101
access-list 101 permit ip 10.4.192.0 0.0.0.0   255.255.224.0
   0.0.0.0
access-list 101 deny ip 0.0.0.0 255.255.255.255 0.0.0.0
   255.255.255.255
```

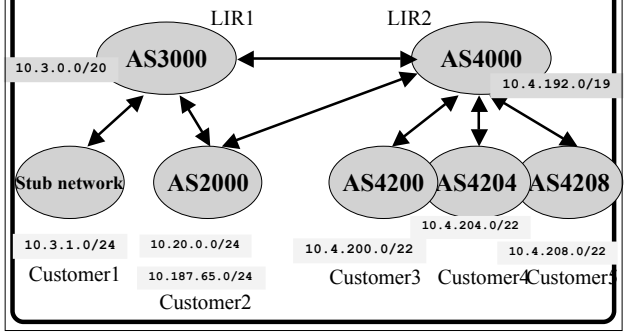
RtConfig Example 2: 'Martians' Filter

```
# RtConfig -protocol ripe -supress_martian
RtConfig> @RtConfig access_list filter AS4000
!
no access-list 100
access-list 100 deny   ip host 0.0.0.0 any
access-list 100 deny   ip 127.0.0.0 0.255.255.255 255.0.0.0 0.255.255.255
access-list 100 deny   ip 10.0.0.0 0.255.255.255 255.0.0.0 0.255.255.255
access-list 100 deny   ip 172.16.0.0 0.15.255.255 255.240.0.0 0.15.255.255
access-list 100 deny   ip 192.168.0.0 0.0.255.255 255.255.0.0 0.0.255.255
access-list 100 deny   ip 192.0.2.0 0.0.0.255 255.255.255.0 0.0.0.255
access-list 100 deny   ip 128.0.0.0 0.0.255.255 255.255.0.0 0.0.255.255
access-list 100 deny   ip 191.255.0.0 0.0.255.255 255.255.0.0 0.0.255.255
access-list 100 deny   ip 192.0.0.0 0.0.0.255 255.255.255.0 0.0.0.255
access-list 100 deny   ip 223.255.255.0 0.0.0.255 255.255.255.0 0.0.0.255
access-list 100 deny   ip 224.0.0.0 31.255.255.255 224.0.0.0 31.255.255.255
access-list 100 deny   ip 169.254.0.0 0.0.255.255 255.255.0.0 0.0.255.255
access-list 100 permit ip 10.4.192.0   0.0.0.0   255.255.224.0   0.0.0.0
access-list 100 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
```

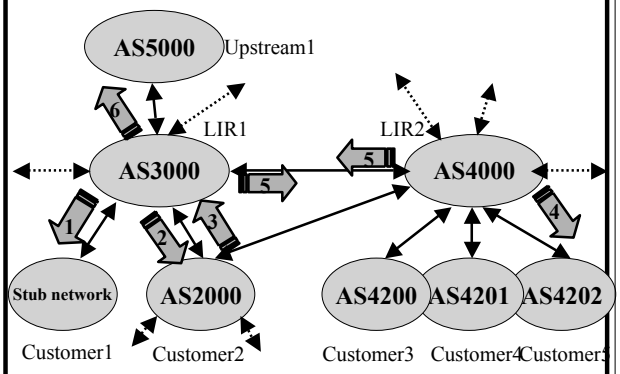
Experimental setup: AS relations



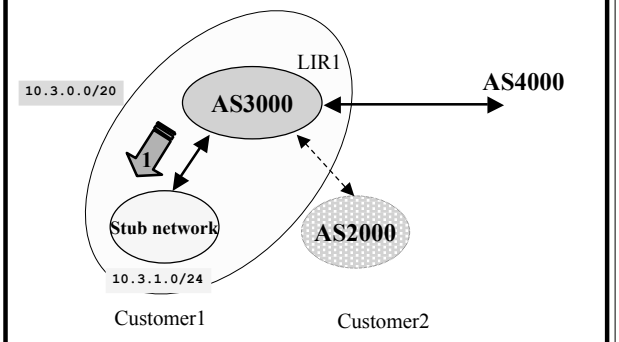
AS relations, including allocations & assignments



Case studies, overview



Case1: Static end-user set-up



Case 1: Static route importation into BGP

- Use policy to filter static routes into BGP
  - Allows for martian filtering
  - AS path stuffing
  - Tagging routes with special communities
  - Other filtering, such as filter host routes

Case 1: Static route importation - aut-num

```
aut-num: AS3000
import:  protocol STATIC into BGP4
            from AS3000
               accept {10.3.1.0/24}
export: to AS4000 announce AS3000
[…]
```

Use this to create a filter that allows static routes to be injected into BGP

RtConfig command: static2bgp ASN router

Case 1: Static import, RtConfig Output

```
RtConfig> @RtConfig static2bgp AS3000 0.0.0.0
!
no ip prefix-list pl130
ip prefix-list pl130 permit 10.3.1.0/24
ip prefix-list pl130 deny 0.0.0.0/0 le 32
!
no route-map AS3000-STATIC-EXPORT
!
route-map AS3000-STATIC-EXPORT permit 10
 match ip address prefix-list pl130
exit
!
router bgp 3000
 redistribute static route-map AS3000-STATIC-EXPORT
exit
```

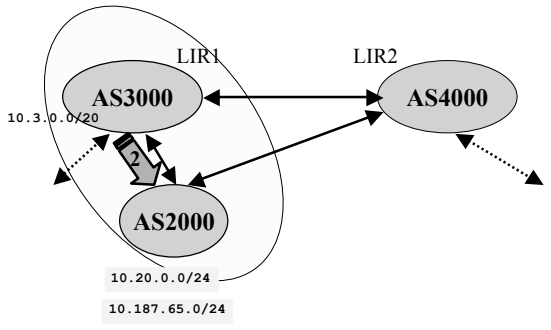
Case 1: Route-set for static routes

- Create route-set object which collects routes together with similar properties:
  - route-set: name starts with RS-
  - members: lists the address ranges or other sets
  - mbrs-by-ref: <mntner-name>
- Modify the aut-num object
- Enables modification of router configuration in indirect way by adding the new customer's static prefix in the DB object
  - You can let admin staff to do this

Case 1: route-set object example

```
route-set:    AS3000:RS-STATIC
descr:        AS3000 Static routes
members:      10.3.1.0/24
admin-c:      BM110-RRTEST
tech-c:       BM110-RRTEST
notify:       bert@example.net
mnt-by:       LIR1-MNT
changed:      bert@example.net 20021001
source:       RRTEST
```

Case 2: Multi-homed customer, provider set-up



Case 2: BGP customers, provider aut-num

```
aut-num: AS3000
import: from AS2000
            accept AS2000
export: to AS2000 announce ANY
[…]
```

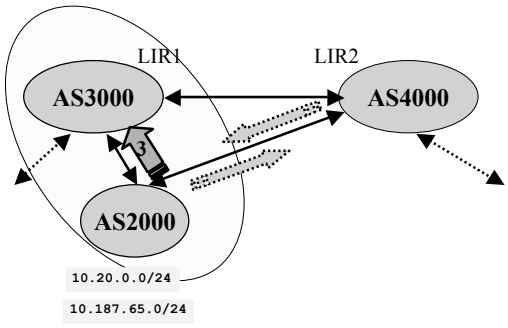
- The simplest policy is strict customer/provider relationship
  - Customer sends its routes to provider
  - Customer accepts everything the provider sends
- RtConfig commands for import:
  - @RtConfig set cisco_map_name = "AS%d-IMPORT"
  - @RtConfig import yourASN your-routerIP neighbourASN neighbour-routerIP

Case 2: Provider setup, RtConfig Output

```
@RtConfig set cisco_map_name = "AS%d-IMPORT"
@RtConfig import AS3000 10.0.1.3 AS2000 10.0.1.2

no ip prefix-list pl137
ip prefix-list pl137 permit 10.20.0.0/24
ip prefix-list pl137 permit 10.187.65.0/24
ip prefix-list pl137 deny 0.0.0.0/0 le 32
!
no route-map AS2000-IMPORT
!
route-map AS2000-IMPORT permit 10
 match ip address prefix-list pl137
exit
!
router bgp 3000
neighbor 10.0.1.2 route-map AS2000-IMPORT in
```

Case 3: Multi-homed customer, customer set-up



Case 3.1: Not Full Multihoming

DB objects:

```
aut-num: AS2000
import:  from AS3000 accept ANY
export:  to AS3000 announce AS2000
import:  from AS4000 accept AS4000
export:  to AS4000 announce AS2000
[…]
```

```
route: 10.20.0.0/24
origin: AS2000
[…]
```

```
route: 10.187.65.0/24
origin: AS2000
[…]
```

Same RtConfig commands

- inverse values (from the case 2 example)
- need set of export/import statements for each provider

Case 3.1: RtConfig Output (export in the notes)

```
no route-map AS3000-IMPORT
!
route-map AS3000-IMPORT permit 10
!
router bgp 2000
neighbor 10.0.1.3 route-map AS3000-IMPORT in
!
!
no ip prefix-list pl134
ip prefix-list pl134 permit 10.4.192.0/19
ip prefix-list pl134 deny 0.0.0.0/0 le 32
!
no route-map AS4000-IMPORT
!
route-map AS4000-IMPORT permit 10
 match ip address prefix-list pl134
exit
!
router bgp 2000
neighbor 10.0.1.4 route-map AS4000-IMPORT in
```

Case 3.2: Full Multihoming

Introducing policy, setting the ***pref*** value
- lower the **pref**, the more preferred the route

aut-num: AS2001
import: from AS3000 action pref=50; accept ANY
export: to AS3000 announce AS2001
import: from AS4000 action pref=100; accept ANY
export: to AS4000 announce AS2001

The difference in the router setup:
- route-map AS3000-IMPORT: set local-preference 950
- route-map AS4000-IMPORT: set local-preference 900

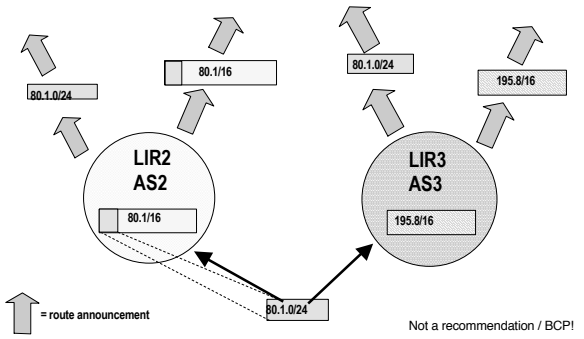
and does not specify address range, since the policy is ANY

Case 3.2: RtConfig Output (export in the notes)

```
no route-map AS3000-IMPORT
!
route-map AS3000-IMPORT permit 1
 set local-preference 950
!
router bgp 2001
neighbor 10.3.15.2 route-map AS3000-IMPORT in
!
!
no route-map AS4000-IMPORT
!
route-map AS4000-IMPORT permit 1
 set local-preference 900
!
router bgp 2001
neighbor 10.4.192.2 route-map AS4000-IMPORT in
```

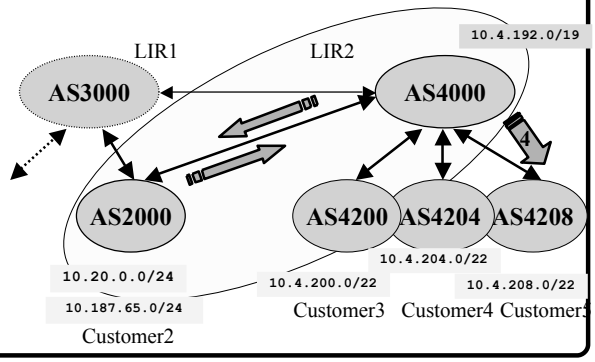
Extra: Multihoming with PA Addresses

announcing more specific prefix from two different ASN



Not a recommendation / BCP!

Case 4: Usage of as-set objects



Case 4: Multiple Customers, Same Policy

- Use as-set objects to group aut-nums
  - as-set: name, starting with AS-; can be hierarchical, using ':'
  - members: ASNs, or as-sets
  - mbrs-by-ref: <mntner-name>
- Refine the aut-num to use as-set
  - In the **from** and **to** statements
- Special expression: PeerAS
  - in the import statement
  - loops through the list from as-set

Case 4: as-set object example

```
as-set:     AS4000:AS-CUSTOMERS
descr:      AS4000 Customers
members:    AS4200,AS4204,AS4208
tech-c:     BM110-RRTEST
admin-c:    BM110-RRTEST
notify:     bert@example.net
mnt-by:     LIR2-MNT
changed:    bert@example.net 20021001
source:     RRTEST
```

Case 4: aut-num object example

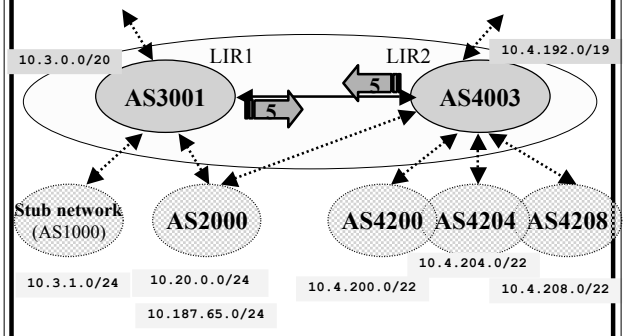
```
aut-num:  AS4000
import:   from AS2000
          accept AS2000
import:   from AS4000:AS-CUSTOMERS
          accept PeerAS
import:   from AS3000
          accept AS3000 AS2000
export:   to AS2000
          announce AS4000
export:   to AS4000:AS-CUSTOMERS
          announce ANY
export:   to AS4000:AS-PEERS
          announce AS4000 AS2000 AS4000:AS-CUSTOMERS
```

Case 4: Adding a New Peer / Customer

Automating the process:

- Obtain and register an AS
- Create route objects for the new AS
- Add the new AS to (one of) your as-set object(s)
- Modify your scripts/programs e.g.
  - add a {IP-address,AS-num,Description}-tuple to a master RtConfig file
  - use Make to rebuild RtConfig file(s)

Case 5: Peering Setup



Case 5.0: BGP with peers - AS4003 view

- Peering policy between peers does not need to be exactly the same:
  - E.g. AS4003 is announcing AS2000 to AS3001, but he is not accepting it!

```
aut-num: AS4003
import:  from AS3001
       accept AS3001 AS2000
export:  to AS4003:AS-PEERS
       announce AS4003 AS2000 AS4003:AS-CUSTOMERS
[... ]

aut-num: AS3001
import:  from AS4003
       accept <^AS4003+AS4003:AS-CUSTOMERS*$>
export:  to AS4003
       announce AS3001 AS2000
```

Case 5.0: RtConfig Output

(*import* in the notes)

```
no access-list 101
access-list 101 permit ip 10.4.200.0   0.0.4.0   255.255.252.0   0.0.0.0
access-list 101 permit ip 10.4.208.0   0.0.0.0   255.255.252.0   0.0.0.0
access-list 101 permit ip 10.20.0.0   0.0.0.0   255.255.255.0   0.0.0.0
access-list 101 permit ip 10.187.65.0   0.0.0.0   255.255.255.0   0.0.0.0
access-list 101 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
!
no route-map AS3001-EXPORT
!
route-map AS3001-EXPORT permit 1
 match ip address 101
!
router bgp 4003
neighbor 10.3.15.4 route-map AS3001-EXPORT out
```

Case 5.0: *-cisco_no_compress_acls*

- Instead of:

```
access-list 101 permit ip 10.4.200.0   0.0.4.0   255.255.252.0   0.0.0.0
```

- We'll have:

```
access-list 101 permit ip 10.4.200.0   0.0.0.0   255.255.252.0   0.0.0.0
access-list 101 permit ip 10.4.204.0   0.0.0.0   255.255.252.0   0.0.0.0
```

Case 5.0: *-cisco_use_prefix_lists*

(import in the notes)

```
no ip prefix-list pl101
ip prefix-list pl101 permit 10.4.200.0/21 ge 22 le 22
ip prefix-list pl101 permit 10.4.208.0/22
ip prefix-list pl101 permit 10.20.0.0/24
ip prefix-list pl101 permit 10.187.65.0/24
ip prefix-list pl101 deny 0.0.0.0/0 le 32
!
no route-map AS3001-EXPORT
!
route-map AS3001-EXPORT permit 1
 match ip address prefix-list pl101
!
router bgp 4003
neighbor 10.3.15.4 route-map AS3001-EXPORT out
```

Case 5.1: BGP with peers - AS3001 view

- This example uses AS Path Filters
  - the <filter> is expressed using regular expression
- It also shows asymmetric policy
  - (AS3001 does not listen to the routes from AS2000 announced back to them by AS4003)

```
aut-num: AS3001
import: from AS4003
     accept <^AS4003+AS4003:AS-customers*$>
export:  to AS4003
       announce AS3001 AS2000
[…]
```

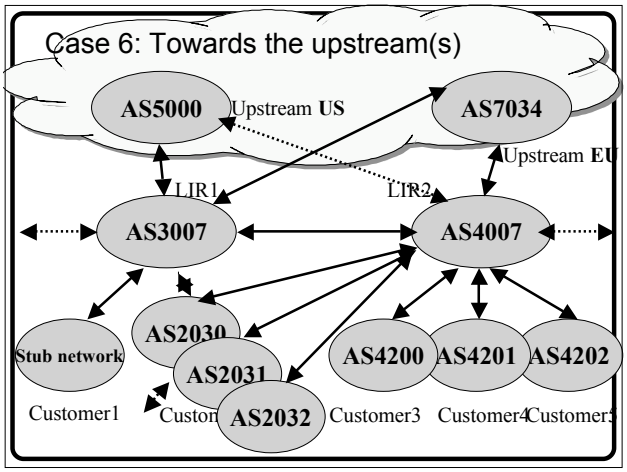
Case 5.1: RtConfig Output (export in the notes)

```
@RtConfig set cisco_map_name = "AS%d-IMPORT"
@RtConfig import AS3001 10.3.15.4 AS4003 10.4.192.3
!
no ip as-path access-list  1
ip as-path access-list 1 permit
   ^(_4003)+(_(4200|4204|4208))*$
!
no route-map AS4003-IMPORT
!
route-map AS4003-IMPORT permit 1
 match as-path 1
!
router bgp 3001
neighbor 10.4.192.3 route-map AS4003-IMPORT in
```

Case 5: Exercise

- How can AS2000 achieve full multihoming / load sharing with two of his upstreams?
  - Both AS3000 & AS4000 should listen to each other's announcements of their multihomed customer, *but* give less preference to the indirect route;
  - This can (maybe) be achieved using "pref"?!
  - Task: create AS3002 & AS4002 to reflect this!
- Time: 5 mins

Case 6: Towards the upstream(s)



Case 6: Using Communities - AS3007

- 3007:20 - multihomed customers, preferred route
- 3007:30 - multihomed customers, backup route
  - (pref=30, localpref=70) (etc)
- 3007:440 - only local traffic
  - Community set to **no export**
- 3007:112 - prepend 2 times to peers
- 3007:222 - prepend 2 times to US upstreams
- 3007:332 - prepend 2 times to EU upstreams

The same community definitions for AS4007!

Case 6: Relevant parts of AS3007

```
# Multihomed customers, backup route
# match community 3007:30, pref=30, localpref=970
import:       from AS3007:AS-BGP-CUSTOMERS
              action pref=30 ;
              accept   community.contains (3007:30)
              AND AS3007:AS-BGP-CUSTOMERS;
# Announce only to customers (not to peers)
import:       from AS3007:AS-BGP-CUSTOMERS
              action
              community = {no_export};
              accept   community.contains (3007:440)
              AND AS3007:AS-BGP-CUSTOMERS;
import:       from AS3007:AS-PEERS
              action pref=40 ;
              accept <^PeerAS$>
import:       from AS3007:AS-PEERS
              action  pref=50;
              accept <^PeerAS+PeerAS:AS-customers$>
```

Case 6: Relevant outputs: for upstreams

```
no access-list 101
access-list 101 permit ip 10.20.0.0 0.0.0.0 255.255.255.0 0.0.0.0
access-list 101 permit ip 10.187.65.0 0.0.0.0 255.255.255.0 0.0.0.0
access-list 101 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255

ip bgp-community new-format

no ip community-list 1
ip community-list 1 permit 3007:222

route-map AS5000-EXPORT permit 1
 match community 1
 match ip address 101
 set as-path prepend 3007 3007
```

Case 6: Relevant outputs: for peers

```
no ip as-path access-list  1
ip as-path access-list 1 permit ^_4000$
!
no route-map AS4000-IMPORT
!
route-map AS4000-IMPORT permit 1
 match as-path 1
 set local-preference 60
!
no ip as-path access-list  2
ip as-path access-list 2 permit ^(_4000)+_(4200|4204|4208)$
!
route-map AS4000-IMPORT permit 2
 match as-path 2
 set local-preference 50
!
router bgp 3007
neighbor 10.4.192.3 route-map AS4000-IMPORT in
```

Case 6: Relevant outputs: for customers

```
no ip community-list 4
ip community-list 4 permit 3007:20
!
no route-map AS2000-IMPORT
!
route-map AS2000-IMPORT permit 1
 match community 4
 match ip address 101
 set local-preference 80
!
no ip community-list 5
ip community-list 5 permit 3007:30
!
route-map AS2000-IMPORT permit 2
 match community 5
 match ip address 101
 set local-preference 70
```

Case 6: Controlling traffic using communities and “pref” value

- AS2030: all traffic from AS3007, AS4007 backup only
- AS2031: load sharing
  - Provider & it’s customers through their link
  - US traffic through AS3007, EU from the AS4007
- AS2032: AS4007 only for “local” traffic
- **Note: there is an implicit logical OR when combining filter rules in aut-num!**
  - **Therefore an explicit AND has to be used!**

Case 6: Exercises / Questions?

- Look into the AS4007 & the config files
  - (case 6.4)
- Look into the different customer setups
  - AS2030, 2031, 2032…
- Use prefix-lists instead
  - -cisco_use_prefix_lists
- Create your own AS60xy
  - XY is your number on the attendees list
  - Choose your policy to AS3007 & AS4007
  - Create RtConfig input file
  - Analyse the resulting output
- Time: 15 minutes

Usage: Potential Practical Problems

Policy can easily get very complex and result in even more complex router configuration

Line limit on cisco AS path filters
- need to be careful when using as-sets

Nervous about configuring routers from public data? Compare this with anti-virus SW updates!

Usage: Preliminary Work (summary)

- Either in the RIPE RR
  - Or in your own routing registry database
- Tasks for your own AS:
  - Create person and maintainer objects
    - Set up PGP authentication
  - Create aut-num objects for each AS
  - Identify IP prefixes associated with each AS
    - Create route objects in the database
  - Create as-set objects where policy is common

Usage: How to Set-up Your Own RR

- Download server SW
  - Choose: RIPE DB SW or IRRd
- Install and set-up server SW
- Register your RR with the IRR (see notes)
- Get the mirroring agreement with the RIPE DB
- Give your customers access to your RR
  - Read-only?
  - With privileges to update objects?

The rest of the IRRToolSet

- peval
- prtraceroute
- aoe
- prpath
- CIDRAdvisor
- roe

IRRToolSet: Intro

- Started as RAToolSet
- Now maintained by RIPE NCC:
  - http://www.ripe.net/ripencc/db/irrtoolset/
  - Mailing list: <irrtoolset@ripe.net>
  - Contact: <ripe-dbm@ripe.net>
- Download: ftp://ftp.ripe.net/tools/IRRToolSet/
- Installation needs: lex, yacc and C++ compiler

IRRToolSet: peval

- Lightweight policy evaluation tool
- Transforms policy expressions in the matching set of routes (e.g. expands AS numbers)
  - may require connection to RR server
- Handy to compose and check your RPSL filter before putting it into RR server
  - Can be used to write router configuration generators
- Web interface:
  - http://www.ripe.net/cgi-bin/peval.cgi

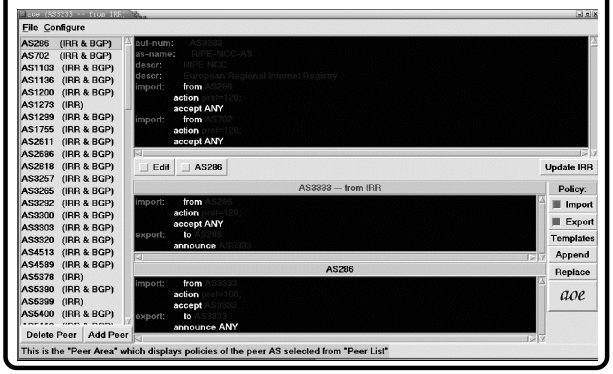
IRRToolSet: prtraceroute

- Prints the route packets take - including policy information (as registered in RR)
  - Requires root privileges and access to RR
- Used as diagnostics tool
- Reports in 3 parts:
  - [ASN] inaddr-name (IP) time
  - Traversed ASNs
  - If the hop was within AS, external, preferred or backup

IRRToolSet: aoe

- Displays the aut-num object for the specified AS
  - GUI (C++/Tcl/Tk)
- Given a BGP dump from a router inside the AS
  - aoe parses the AS_PATH attributes
  - determines the peer ASes
    - by taking the first AS number in the AS_PATH
  - takes the import policies for each peer AS
    - by taking the last AS number in the AS_PATHs that start with the peer's AS number

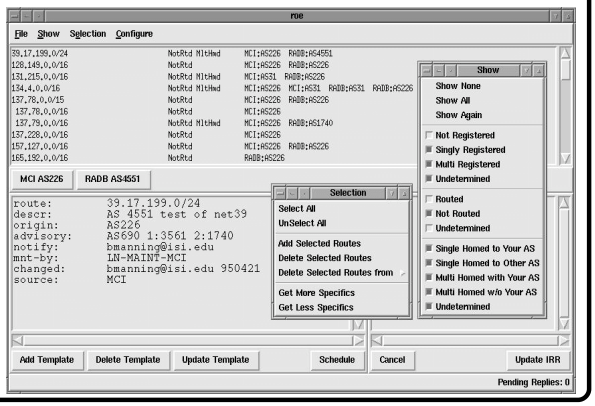
IRRToolSet: aoe (screen dump)



IRRToolSet: The Rest

- *prpath* enumerates possible paths between two AS-s, as registered in RR
- *CIDRAdvisor* suggests safe aggregates per AS
  - Practical usage: http://www.cidr-report.org/
- *rpslcheck* syntax checks objects for IRR
  - But the RIPE DB rules are slightly different
- *roe* GUI, lists the routes & dependencies, can add / delete specified routes

IRRToolSet: roe, Screendump



IRRToolSet: Conclusions

- The quality of data provided by tools strongly depends on the data you have in the RR!
  - Crucial to maintain RR objects up-to-date
- Tools can work with both RIPE and IRRd based RR's
- Using the tools will help you to 100% benefit from registering your data in RR, to achieve:
  - automating access-list generation
  - avoiding mistakes
  - improving configuration/operation process

IRRToolSet: Practical Exercise

- Task: Use one of the IRRTools (15 minutes)
  - on the web or command-line
  - http://www.ripe.net/ripencc/pub-services/db/irrtoolset/index.html
- Have you used the IRRTools before?
- What are their most useful features?
- Which new features would you like to see?
- Can you suggest any improvements? Bug reports?
- Do you know of any similar tools/projects/analysis?

Extra: Course/Workshop Server Setup

- RedHat 8 Linux Server , running :
  - zebra (for BGP)
  - whoisd (RIPE NCC whois server, latest version (3.1.1))
  - Ssh
  - whois client

Extra: RtConfig

- Version 4.0 supports RPSL (Latest version is 4.7.3 as at 13 February 2003)
- Generates Cisco, Bay's BCC, Juniper's Junos and Gated/RSd configurations
- Creates route and AS path filters.
- Can also create ingress/egress filters (Cisco only)

Extra: RtConfig options

-help
-version
-s <source-list>
-f <file name>
-config <config-format>
-supress-martian
-T [whois_query | whois_response | input | all]

Extra: Initialise Cisco list parameters

$ RtConfig -cisco_use_prefix_lists

>RTConfig

@RtConfig set cisco_map_first_no = 10
@RtConfig set cisco_map_increment_by = 10
@RtConfig set cisco_prefix_acl_no = 130
@RtConfig set cisco_aspath_acl_no = 130
@RtConfig set cisco_pktfilter_acl_no = 130
@RtConfig set cisco_community_acl_no = 30
@RtConfig set cisco_max_preference = 100

Extra: Cisco: Martians filter access list

```
$ RtConfig-cisco_use_prefix_lists -supress_martian
RtConfig> @RtConfig access_list filter AS4000
!
no ip prefix-list pl100
ip prefix-list pl100 deny   0.0.0.0/0 ge 32
ip prefix-list pl100 deny   127.0.0.0/8 le 32
ip prefix-list pl100 deny   10.0.0.0/8 le 32
ip prefix-list pl100 deny   172.16.0.0/12 le 32
ip prefix-list pl100 deny   192.168.0.0/16 le 32
ip prefix-list pl100 deny   192.0.2.0/24 le 32
ip prefix-list pl100 deny   128.0.0.0/16 le 32
ip prefix-list pl100 deny   191.255.0.0/16 le 32
ip prefix-list pl100 deny   192.0.0.0/24 le 32
ip prefix-list pl100 deny   223.255.255.0/24 le 32
ip prefix-list pl100 deny   224.0.0.0/3 le 32
ip prefix-list pl100 deny   169.254.0.0/16 le 32
ip prefix-list pl100 permit 10.4.192.0/19
ip prefix-list pl100 deny 0.0.0.0/0 le 32
```

Extra: Juniper: access list

$ RtConfig –protocol ripe -config junos
RtConfig> @RtConfig access_list filter AS4000

```
policy-statement prefix-list-100 {
  term prefixes {
    from {
      route-filter 10.4.192.0/19 exact accept;
    }
  }
  term catch-rest {
     then reject;
  }
}
```

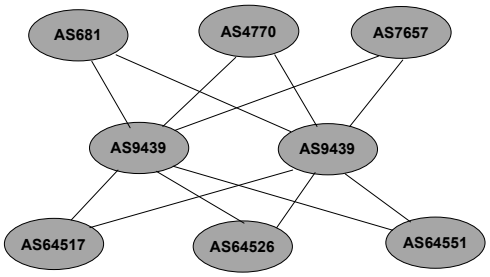
Extra: Mailing Lists

- <db-wg@ripe.net>
  - RIPE network management database
- <irrtoolset@ripe.net>
  - Internet Routing Registry ToolSet project
- **<rpsIng@ripe.net>**
  - **extensions to RPSL related to IPv6 and multicast**

Wellington Internet Exchange

- Distributed exchange running over Citylink
  - over 60 Km of fibre in city centre
  - approx 100 participants
  - extensive use of Linux routers with Zebra
  - many small players with no BGP clue
  - larger players wary because of lack of clue
- route reflectors need to implement policies to "make it safe"

WIX network (part of)



Preliminary work

- Because we have lots of Private AS numbers we have to run our own routing registry database
  - We chose irrd because our requirements are modest
- Create maintainer and person objects
- Set up PGP authentication
- Create aut-num objects for each AS
- Identify IP prefixes associated with each AS
  - Create route objects in database
- Create as-set objects where policy is common

AS9439 Configuration

- AS9439 has a relatively simple set of routing requirements
  - BGP peering with peers
  - Number of private AS = 73
  - Number of public AS = 25
  - AS9439 has no prefixes of its own!
- Use RPSL and RtConfig

AS9439 policies

```
aut-num:  AS9439
as-name:  WIX-AS9439
descr:    WIX Master AS
import:   from AS9439:AS-PRIVATE
          accept PeerAS
import:   from AS9439:AS-PUBLIC
          accept PeerAS
export:   to AS9439:AS-PRIVATE
          announce AS9439:AS-PRIVATE
                   AS9439:AS-PUBLIC
export:   to AS9439:AS-PUBLIC
          announce AS9439:AS-PRIVATE
                   AS9439:AS-PUBLIC
```

AS9439:AS-PUBLIC

```
as-set:     AS9439:AS-PUBLIC
descr:      Public Ases for WIX
members:    AS681, AS10022, AS17412, AS17792,
  AS18119, AS4740, AS4768, AS4770, AS7657,
  AS9325, AS9338, AS9436, AS9495, AS9503, AS9736,
  AS9790, AS9872, AS9887
tech-c:     AL325-WIX
notify:     rpsl@lpnz.org
mnt-by:     MAINT-WIX-NZ
changed:    asjl@lpnz.org 20020612
source:     WIX
```

AS9439:AS-PRIVATE

as-set: AS9439:AS-PRIVATE
descr: Private ASes for WIX
members: AS64512, AS64517, AS64525, AS64530, AS64537, AS64543, AS64548, AS64553, AS64513, AS64518, AS64526, AS64532, AS64538, AS64544, AS64549, AS65025, AS64514, AS64520, AS64527, AS64534, AS64539, AS64545, AS64550, AS65498, AS64515, AS64521, AS64528, AS64535, AS64540, AS64546, AS64551, AS65518, AS64516, AS64523, AS64529, AS64536, AS64541, AS64547, AS64552, AS65531
tech-c: AL325-WIX
source: WIX

AS64512 policies

aut-num: AS64512
as-name: WIX-AS64512
descr: Citylink
admin-c: AL325-WIX
tech-c: AL325-WIX
import: from AS9439
accept ANY
export: to AS9439
announce AS64512
notify: rpsl@lpnz.org
mnt-by: MAINT-WIX-NZ
changed: asjl@lpnz.org 20020610
source: WIX

AS64512 prefixes

- AS64512 has these prefixes:

  210.86.11.236/30 210.48.103.144/28 210.48.103.136/29 210.48.103.0/28 203.97.231.224/28 203.96.131.96/29 203.79.85.80/29 203.109.154.32/28 203.109.148.24/29

- Note small address blocks that wouldn't normally be seen at an Internet Exchange
- This is not unusual on the WIX!

Software Tools (1)

- Cisco output from RtConfig almost works with Zebra
  - Use cisco2zebra filter to massage the output
  - It's a hack. The solution is to fix RtConfig
- Use mk-cisco to generate input for RtConfig processing
  - Input to mk-cisco looks like:
    - 202.7.0.1:64512:Citylink
    - 202.7.0.5:64546:Puskas
    - 202.7.0.12:64526:CitylinkVoIP

Makefile

```
#
# $Id: Makefile,v 1.8 2002/07/05 04:44:41 asjl Exp $
#
IRR_HOST=cheviot.lpnz.org
IRR_PORT=43
IRR_SOURCES=WIX

Zico.cfg: Zico.master mk-cisco Makefile
        /home/asjl/NZNOG/mk-cisco < Zico.master > Zico.rpsl
        RtConfig -h $(IRR_HOST) -p $(IRR_PORT) \
                -s $(IRR_SOURCES) \
                -cisco_use_prefix_lists < Zico.rpsl \
                | /home/asjl/NZNOG/cisco2zebra > Zico.cfg
```

Software Tools (2)

- Tools hide complexity:

```
$ wc -l Zico.master Zico.rpsl Zico.cfg
        62 Zico.master
       755 Zico.rpsl
      3442 Zico.cfg
```

- Can use mk-junos to build Juniper configs if Juniper donate a router!

```
$ wc -l Zico.cfg-j
      5410 Zico.cfg-j
```

Software Tools (3)

- BGP naïve customers get a sample BGP configuration
  - Generated using mk-clients tool

Adding a new peer

- Register an AS in the WIX database
- Add routes for the new AS
- Add the new AS to AS9439:AS-PUBLIC or AS9439:AS-PRIVATE
- Add a {IP-address,AS-num,Description}-tuple to master config file
- Use Make to rebuild config file(s)

What Next?

- Run your own routing registry?
  - Decide which software to run
    - IRRd or RIPE v3
- Or register your routes in a public registry such as APNIC?
- Or both?
  - You may not want to reveal all your internal secrets!

What Next? (cont)

- Look at your customers, peers, providers and decide how to represent policy in RPSL
- Implement router configuration using RPSL and associated tools!

References

- Using RPSL in Practice - RFC 2650
- RPSL - RFC 2622
  - http://www.rfc-editor.org/rfcsearch.html
- IRRToolSet
  - http://www.ripe.net/ripencc/pub-services/db/irrtoolset/
- RPSL Training Page
  - http://www.isi.edu/ra/rps/training/
- RIPE database manual
  - http://www.ripe.net/ripe/docs/databaseref-manual.html

References (cont)

- RADB
  - http://www.merit.edu/radb/
- RIPE database software
  - ftp://ftp.ripe.net/ripe/dbase/software
- IRRd software
  - http://www.irrd.net/
- Zebra
  - http://www.zebra.org

Acknowledgements

- Mark Prior <mrp@iagu.net>
- Ambrose Magee <ambrose.magee@ericsson.com>
- Simon Blake <simon@katipo.co.nz>
- APNIC
  - for asking me to present the tutorial

Contact Details

$ whois -h whois.apnic.net Andy Linton

```
person:      Andy Linton
address:     149 Cecil Road
address:     Wilton
address:     Wellington
country:     NZ
phone:       +64 4 970 1764
e-mail:      asjl@lpnz.org
nic-hdl:     AL325-AP
mnt-by:      MAINT-AU-AL325-AP
changed:     asjl@lpnz.org 20021119
source:      APNIC
```