Automating and Securing Route Servers with

Barry O'Donovan, INEX

APNIC Academy Webinar Series, June 15th 2021

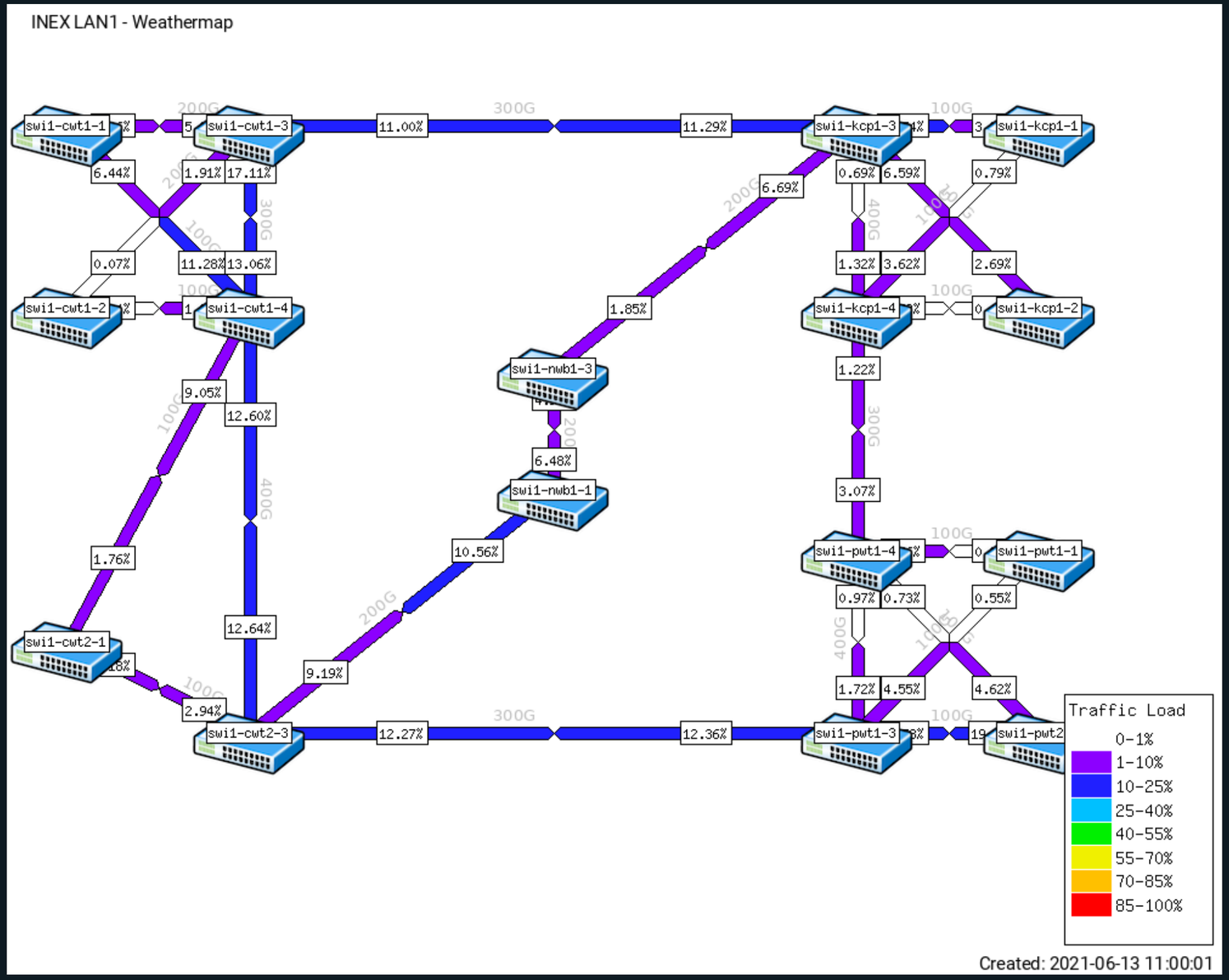
Barry O'Donovan

- Twitter @barryo79 - https://www.barryodonovan.com/
- Husband, dad, geek, sci fi, politics
- Business owner - Island Bridge Networks
- Management and operations team @ INEX
- Project manager / lead developer for @ixpmanager
- Open source projects, financial programming and accountancy

Agenda for Today's Webinar

- Introductions (Barry, INEX, IXP Manager)
- IXPs and Route Servers
  - What can go wrong?
- Automating and Securing Route Servers with IXP Manager
  - Looking Glass
  - IRRDB Filtering
  - RPKI

- internet exchange point for the island of Ireland
- 3 IXPs / 7+ PoPs / >100 members / >600Gb peaks
- founder and core team for IXP Manager for 15+ years
- 12 production route server instances
  - 3 IXPs x two IP protocols x resilience


INEX LAN1 - Weathermap
swi1-cwt1-1
swi1-cwt1-3
swi1-cwt1-2
swi1-cwt1-4
swi1-cwt2-1
swi1-cwt2-3
swi1-nwb1-3
swi1-nwb1-1
swi1-kcp1-3
swi1-kcp1-1
swi1-kcp1-4
swi1-kcp1-2
swi1-pwt1-4
swi1-pwt1-1
swi1-pwt1-3
swi1-pwt2
11.00%
11.29%
6.44%
1.91%
17.11%
6.69%
0.69%
6.59%
0.79%
0.07%
11.28%
13.06%
1.32%
3.62%
2.69%
1.85%
1.22%
9.05%
12.60%
6.48%
3.07%
1.76%
10.56%
0.97%
0.73%
0.55%
12.64%
9.19%
1.72%
4.55%
4.62%
2.94%
12.27%
12.36%
Traffic Load
0-1%
1-10%
10-25%
25-40%
40-55%
55-70%
70-85%
85-100%
Created: 2021-06-13 11:00:01

— full stack management system for IXPs
— teaches and implements best practice
— in use by at least 166 IXPs
— free and open source software
— lots of resources @ https://www.ixpmanager.org/

Development Effort (1/2)

— continue to develop to solve our own problems
  — our problems are usually your problems!
— core team, project management and oversight
— code quality reviews
— first user / eat our own dog food

Development Effort (2/2)

Full-time developer, thanks to patrons:

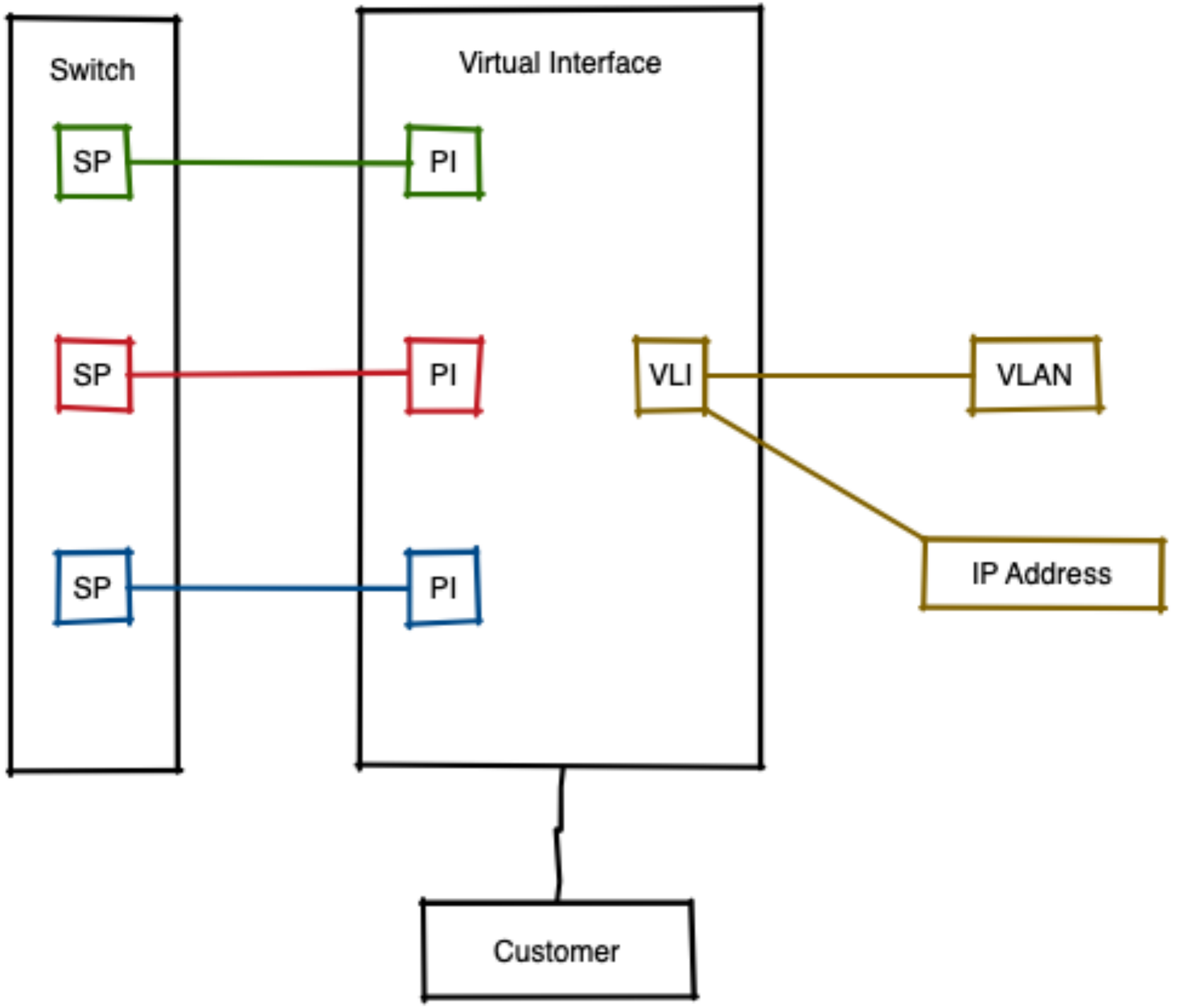
And sponsors: INX-ZA, LONAP and STHIX; GRIX, InterLAN and NAMEX

Top Level Objects in IXP Manager

— Facilities (locations / data centres)
— Racks (cabinets) [∈ facility]
— Infrastructures (think of these as an IXP)
— VLANs [∈ infrastructure]
— Switches [∈ infrastructure, ∈ rack]
— Switch Ports [∈ switch]

Interfaces in IXP Manager

- Flexible schema that has survived 20 years
- Member port represented by a: virtual interface (VI)
  - a VI has one or more physical interfaces (PI)
    - each PI has a switch port (SP)
  - a VI has one or more VLAN interfaces (VLI)
    - each VLI has a parent VLAN
    - optionally has an IPv4 and/or IPv6 Address
    - options such as route server participation


Switch
Virtual Interface
SP
PI
SP
PI
VLI
VLAN
SP
PI
IP Address
Customer

Demo

— Quick look around

— Add a new member

Auto-Provisioning in IXP Manager

When a interface is added to IXP Manager, you (can) get:

— Route Collector BGP session auto-provisioned
— Secure Route Server BGP sessions auto-provisioned
— MRTG / graphing auto-provisioned
— Peer to peer graphs auto-provisioned
— Smokeping target for member’s interface
— Nagios monitoring of member’s interface
— AS112 BGP session
— ARPA DNS for IXP assigned address
— RIR AS-SET / ASN objects
— And more...

Route Servers

Route Servers

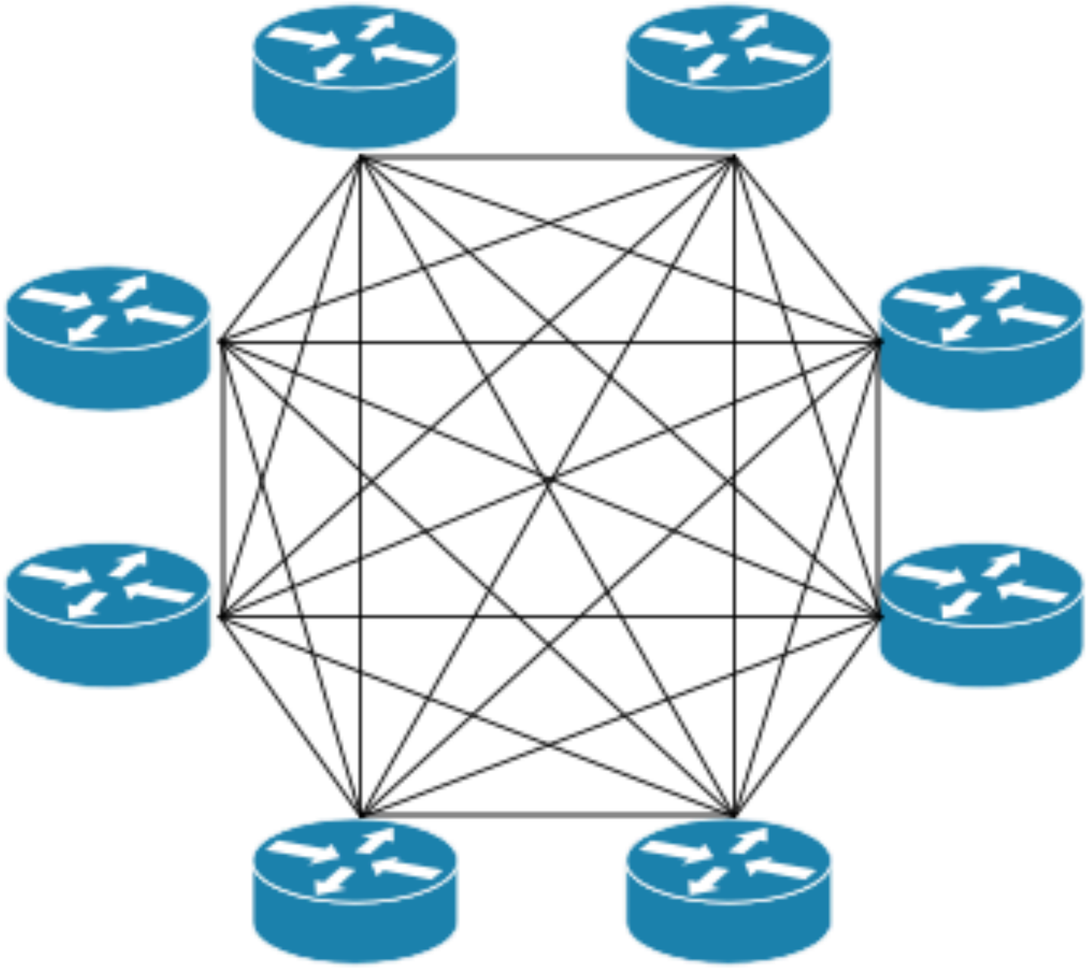
— a third-party brokering system providing multilateral interconnection via BGP

— BGP sessions required for full mesh peering:

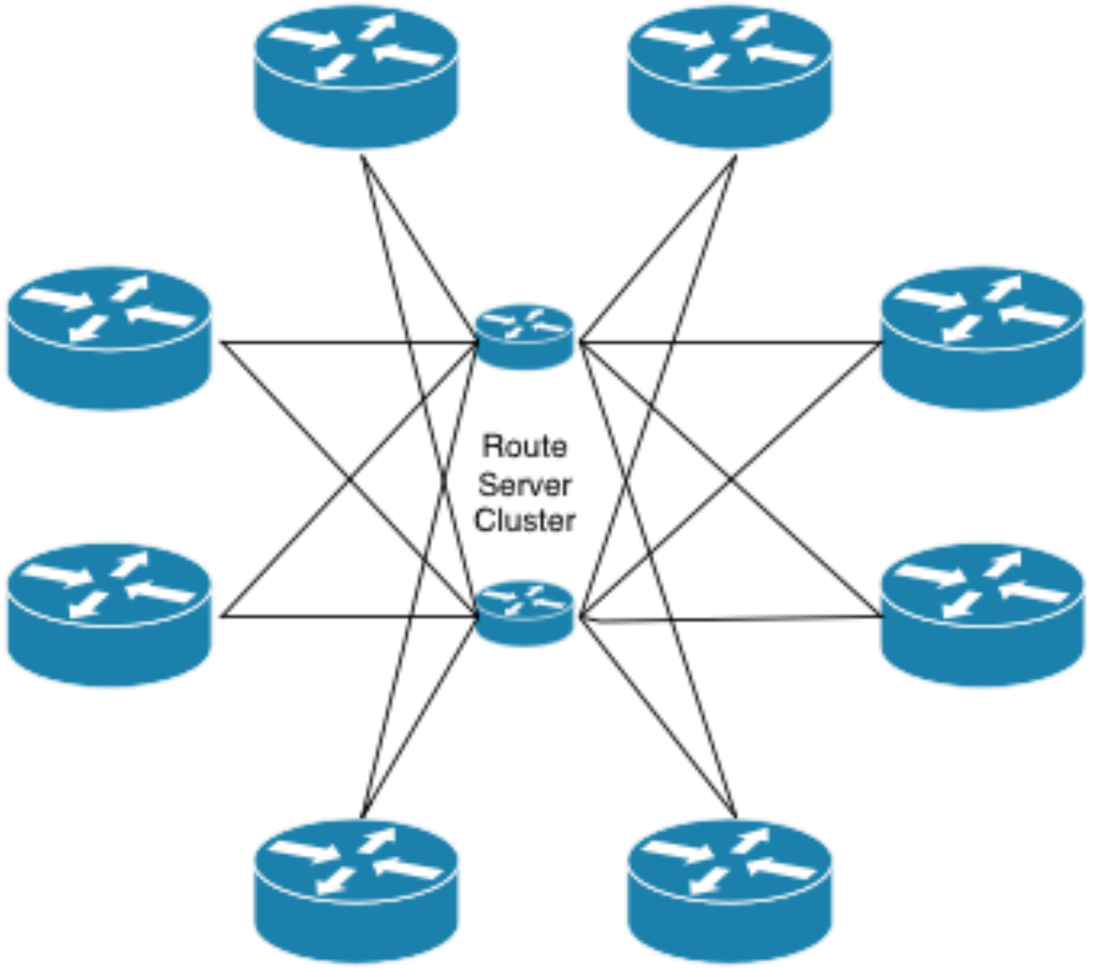
$$\frac{n(n-1)}{2}$$

— 10 participants: 45 sessions

— 100 participants: 4,950 sessions



IXP full mesh peering relationships



IXP route server peering relationships

Route Servers

— routes announced by one route server client are forwarded to all other participating clients

— **does not** forward traffic and is not a router

— rfc7947 and rfc7948, Hilliard et al:

"the overhead associated with dense interconnection can cause substantial operational scaling problems for participants of larger IXPs"

Route Servers

- Considered a production level service at IXPs
  - Stability, reliability, consistency
- Threats can be malicious or accidental
- Threats include:
  - Route leak (DFZ or targeted network)
  - Next hop hijacking
  - Route server software bugs

Route Servers & IXP Manager

— INEX has operated route servers since 2007
— INEX CTO co-authored the RFCs
— We have given presentations, workshops, tutorials
— Always automated - never deployed manually
— Always secured with prefix filtering

This experience and knowledge has been distilled into IXP Manager

Subject: [inex-tech] Route server system now in beta

Date: Fri Nov 23 12:20:17 GMT 2007
From: Nick Hilliard <email@inex.ie>

Following the announcement at the last INEX members meeting that we were looking into running a route server system, we are now pleased to announce that we now have a route server system which is in stable beta.

As a brief summary, the route server system offers the following advantages:

- dramatically reduces the number of BGP sessions required to peer with other INEX members
- strict route filtering on inbound announcements means that only prefixes registered at RIPE by exchange members will be visible
- dual-hosted system offers high reliability
- community based filtering allows route server users to control which INEX members their prefixes are sent to

Well-Known Community Filters

Allowing members to control prefix propagation is a critical feature to drive route server usage.

```
router bgp 65503
  address family ipv4
    neighbor 192.0.2.8 remote-as 65501
    neighbor 192.0.2.8 route-map ix-router-server-out out
    neighbor 192.0.2.8 send-community
!
route-map ix-router-server-out
  set community 65501:65502
```

More information on https://docs.ixpmanager.org/

Well-Known Community Filters - Standard

Description	Community
Prevent announcement of a prefix to a peer	`0:peer-as`
Announce a route to a certain peer	`rs-asn:peer-as`
Prevent announcement of a prefix to all peers	`0:rs-asn`
Announce a route to all peers	`rs-asn:rs-asn`

— No one except X, Y and Z: `0:rs-asn rs-asn:X rs-asn:Y rs-asn:Z`

— Everyone but X and Y: `0:X 0:Y`

Well-Known Community Filters - Large Communities

Description	Community
Prevent announcement of a prefix to a peer	`rs-asn:0:peer-as`
Announce a route to a certain peer	`rs-asn:1:peer-as`
Prevent announcement of a prefix to all peers	`rs-asn:0:0`
Announce a route to all peers	`rs-asn:1:0`

— Standard filtering does not work with 32-bit ASNs

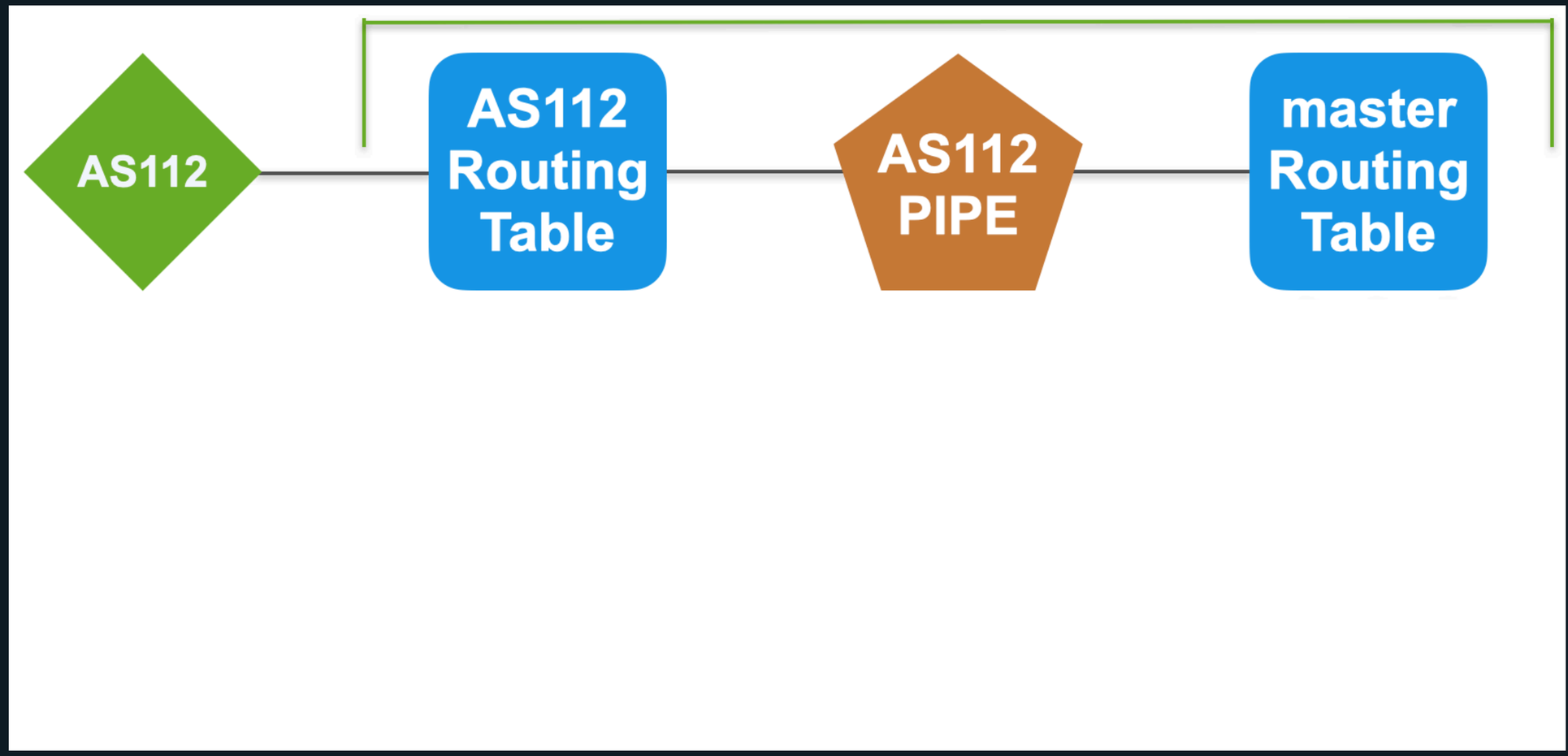
— A route server client should not mix standard 16-bit communities and large communities – please choose one or the other

Well-Known Community Filters - Large Communities Bonus

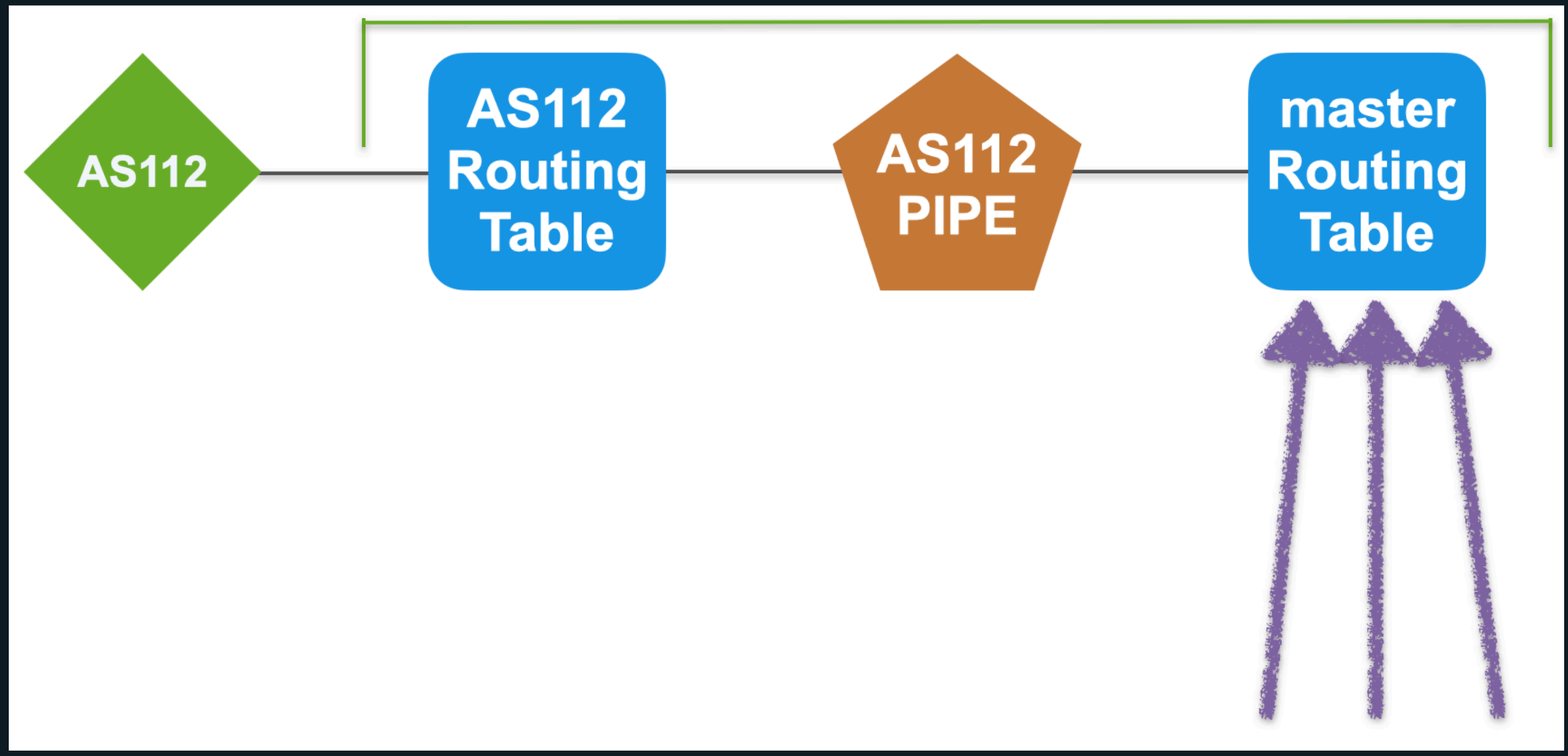
Description	Community
Prepend to peer AS once	`rs-asn:101:peer-as`
Prepend to peer AS twice	`rs-asn:102:peer-as`
Prepend to peer AS thrice	`rs-asn:103:peer-as`

NB: communities control propagation of a client's routes to other clients. Clients responsible for filtering inbound themselves.

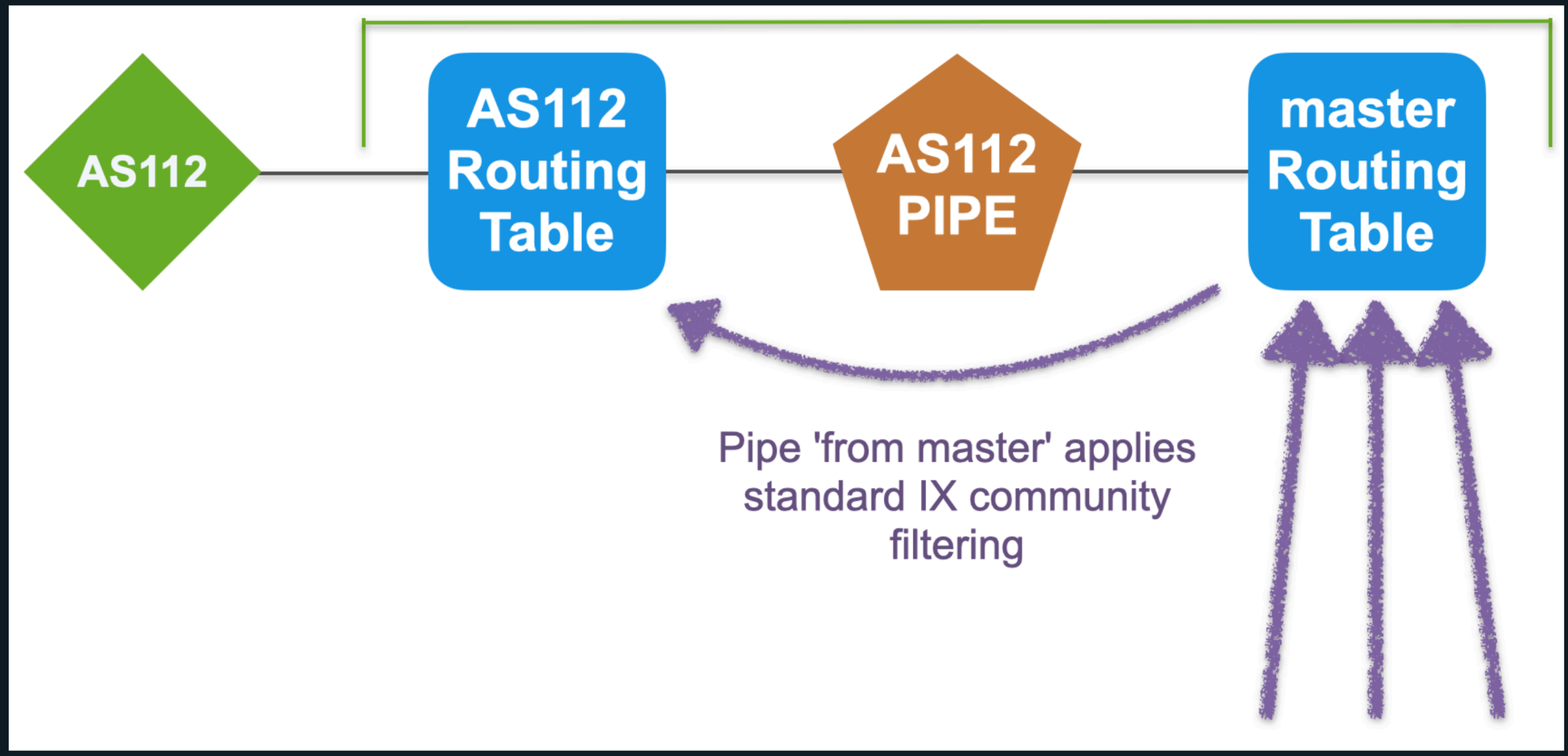
IXP Manager Bird Topology - Export to Client



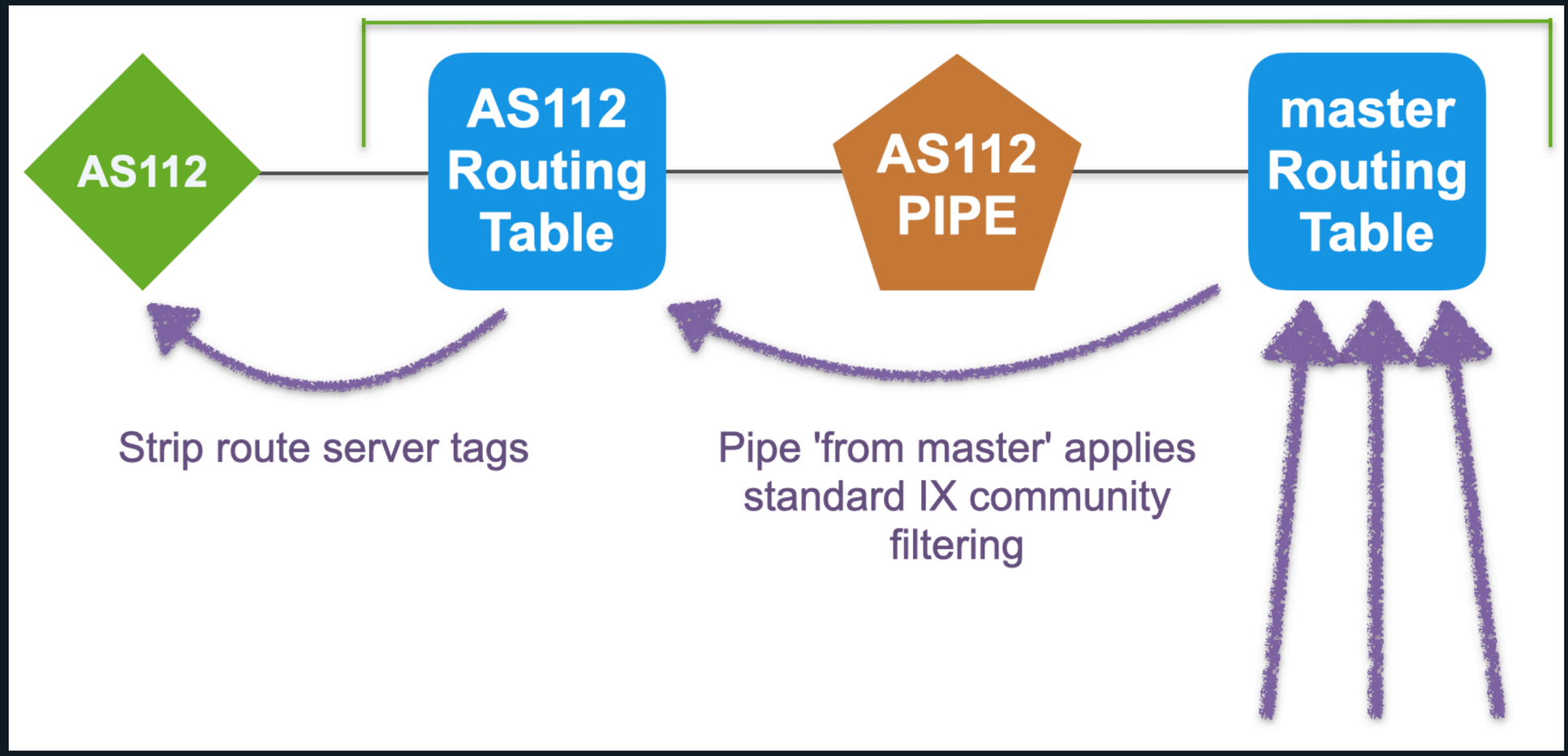
IXP Manager Bird Topology - Export to Client



IXP Manager Bird Topology - Export to Client



IXP Manager Bird Topology - Export to Client



Configuring Route Servers with IXP Manager

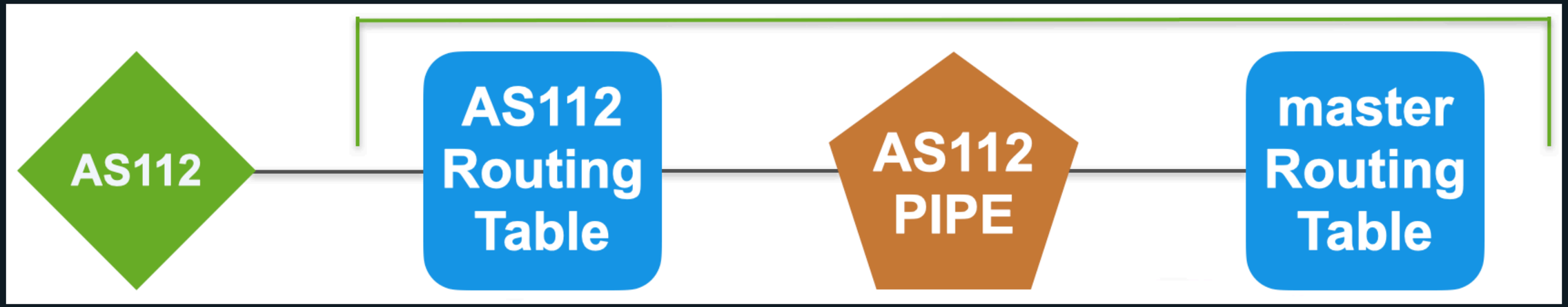
Required Parameters

— AS Number (should be dedicated for the route servers)
— Peering IPv4 and IPv6 address
— Router ID (Usually the peering IPv4 address)
— BGP software (IXP Manager has Bird v1 and v2 baked in)
— Support: MD5? Large communities?
— RPKI?
— Looking glass?

Then add the router in the IXP Manager API...

Examine Client Configuration

Remember:



Examine Client Configuration - Protocol (Peer Defn)

```
protocol bgp pb_0001_as1213 from tb_rsclient {
        description "AS1213 - HEAnet";
        neighbor 192.0.2.32 as 1213;
        ipv4 {
            import limit 100 action restart;
            import filter f_import_as1213;
            table t_0001_as1213;
            export filter f_export_as1213;
        };
        # enable rfc1997 well-known community pass through
        interpret communities off;
        password "yxtRJmDvTYNh";
}
```

Examine Client Configuration - Import Filter

```
filter f_import_as1213 {
    # Filter small prefixes
    if ( net ~ [ 0.0.0.0/0{25,32} ] ) then {
      bgp_large_community.add( IXP_LC_FILTERED_PREFIX_LEN_TOO_LONG );
        accept;
    }

    if !(avoid_martians()) then {
        bgp_large_community.add( IXP_LC_FILTERED_BOGON );
        accept;
    }

    # Peer ASN == route's first ASN?
    if (bgp_path.first != 1213 ) then {
        bgp_large_community.add( IXP_LC_FILTERED_FIRST_AS_NOT_PEER_AS );
        accept;
    }
```

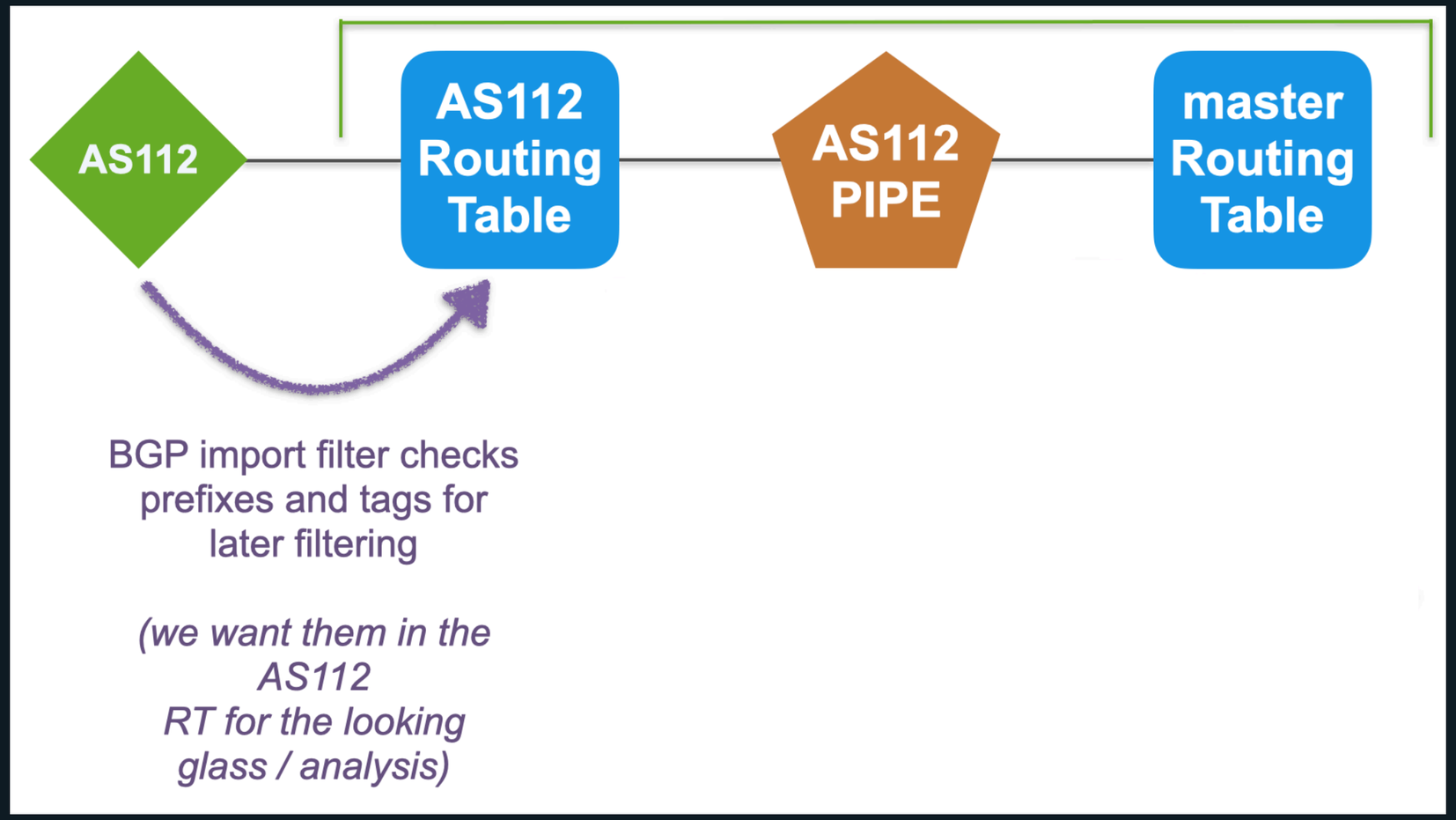
Examine Client Configuration - Import Filter

```
# set of all IPs this ASN uses to peer with on this VLAN
allips = [ 192.0.2.32 ];

# Prevent BGP NEXT_HOP Hijacking
if !( from = bgp_next_hop ) then {

    # need to differentiate between same ASN next hop or actual next hop hijacking
    if( bgp_next_hop ~ allips ) then {
        bgp_large_community.add( IXP_LC_INFO_SAME_AS_NEXT_HOP );
    } else {
        # looks like hijacking (intentional or not)
        bgp_large_community.add( IXP_LC_FILTERED_NEXT_HOP_NOT_PEER_IP );
        accept;
    }
}
```

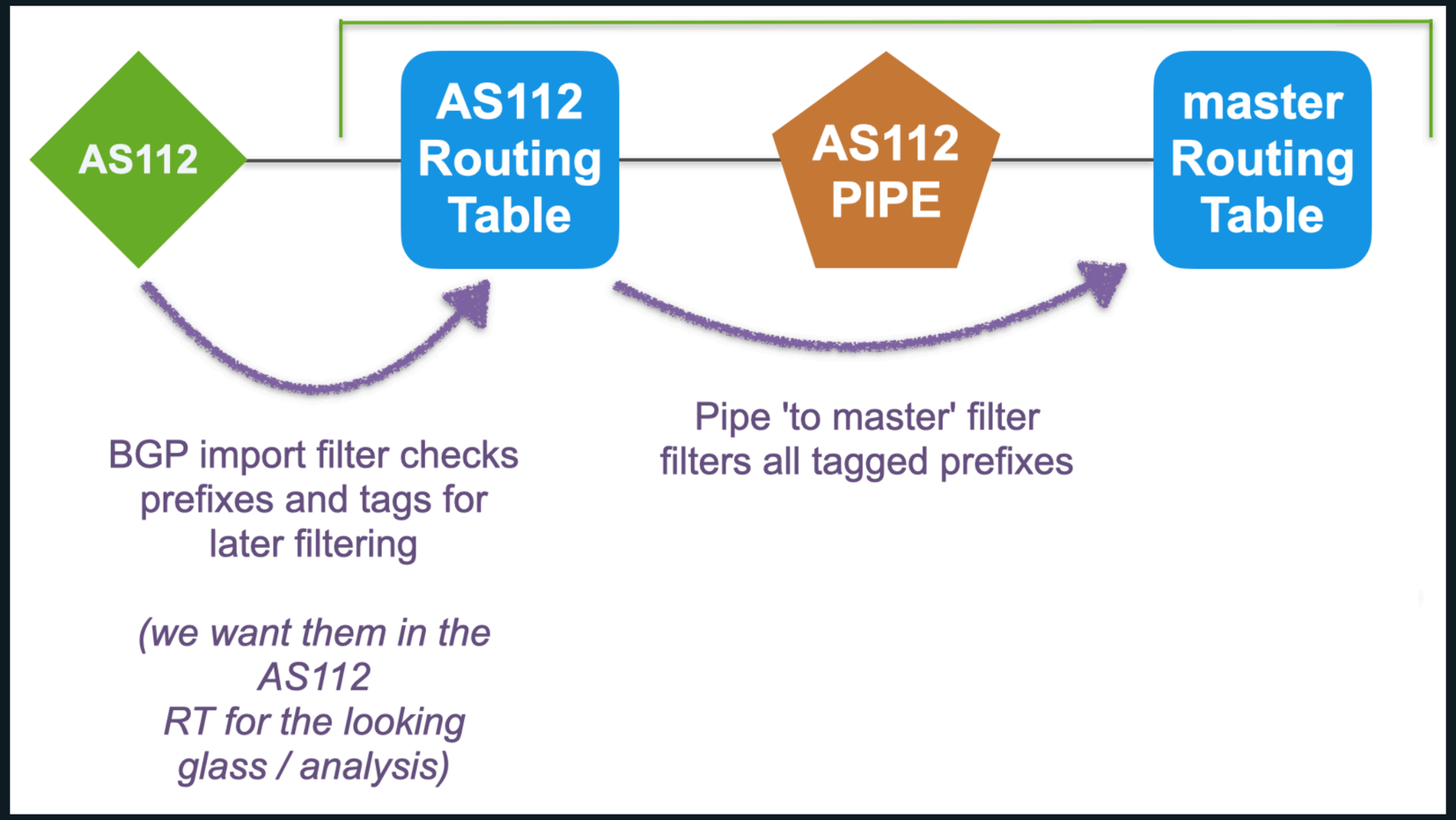
Examine Client Configuration - Import Filter

```
    # Filter Known Transit Networks
    if filter_has_transit_path() then accept;

    # Belt and braces: no one needs an ASN path with > 64 hops, that's just broken
    if( bgp_path.len > 64 ) then {
        bgp_large_community.add( IXP_LC_FILTERED_AS_PATH_TOO_LONG );
        accept;
    }

    # Skipping RPKI check -> RPKI not enabled / configured correctly.
    bgp_large_community.add( IXP_LC_INFO_RPKI_NOT_CHECKED );

    # This ASN was configured not to use IRRDB filtering
    bgp_large_community.add( IXP_LC_INFO_IRRDB_NOT_CHECKED );

    accept;
}
```

IXP Manager Bird Topology - Import from Client



IXP Manager Bird Topology - Import from Client



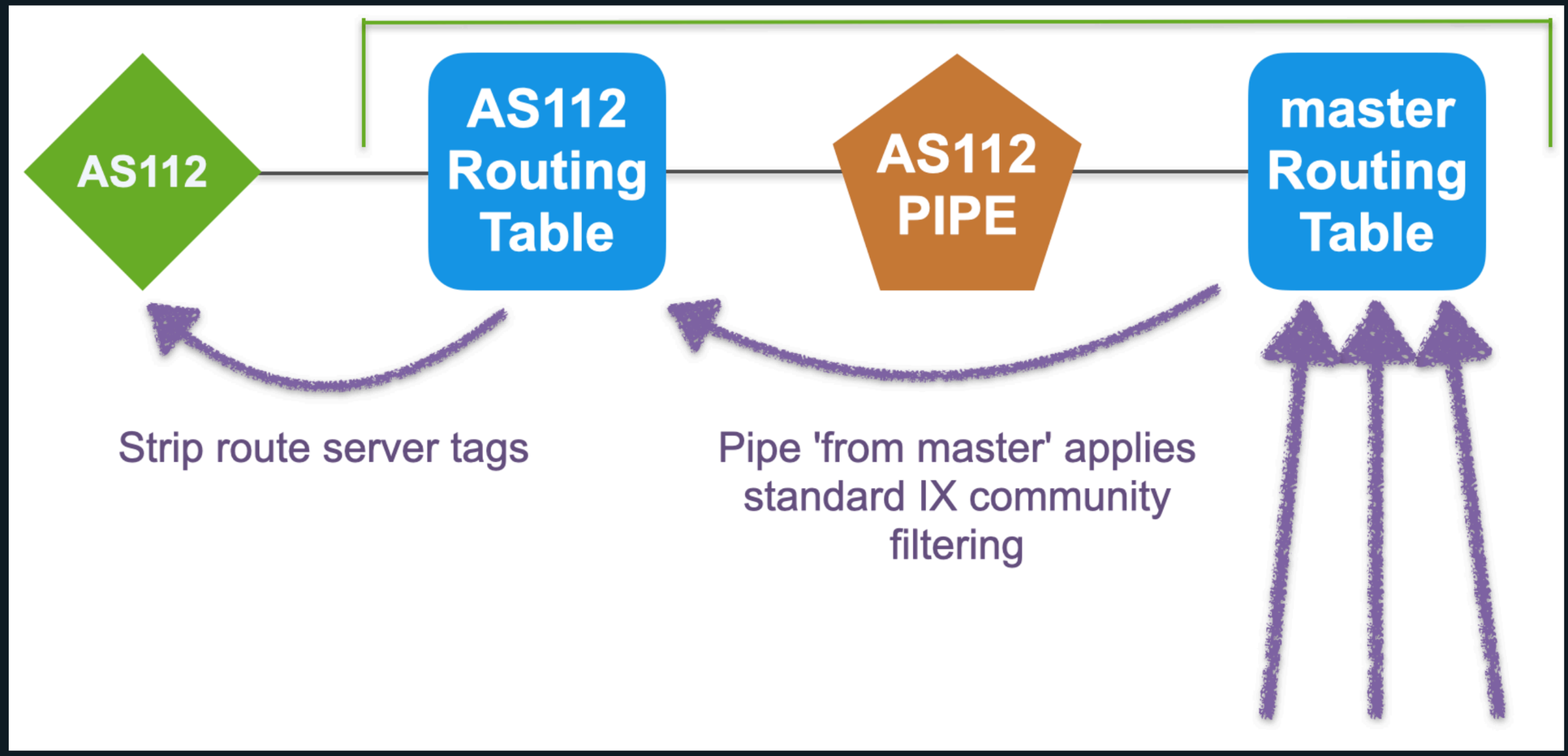
Examine Client Configuration - Protocol (Peer Defn)

```
protocol bgp pb_0001_as1213 from tb_rsclient {
        description "AS1213 - HEAnet";
        neighbor 192.0.2.32 as 1213;
        ipv4 {
            import limit 100 action restart;
            import filter f_import_as1213;
            table t_0001_as1213;
            export filter f_export_as1213;
        };
        # enable rfc1997 well-known community pass through
        interpret communities off;
        password "yxtRJmDvTYNh";
}
```

Examine Client Configuration - Export Filter

```
filter f_export_as1213 {

    # strip internal communities
    bgp_large_community.delete( [( routeserverasn, *, * )] );
    bgp_community.delete( [( routeserverasn, * )] );

    accept;
}
```

Examine Client Configuration - Export Filter



Examine Client Configuration - Pipe

```
protocol pipe pp_0001_as1213 {
        description "Pipe for AS1213 - HEAnet";
        table master4;
        peer table t_0001_as1213;
        import filter f_export_to_master;
        export where ixp_community_filter(1213);
}
```

Examine Client Configuration - Pipe

```
...
define IXP_LC_FILTERED_BOGON = ( routeserverasn, 1101, 3  );
...


filter f_export_to_master
{

    if bgp_large_community ~ [( routeserverasn, 1101, * )]
        then reject;

    accept;
}
```

Examine Client Configuration - Pipe

```
protocol pipe pp_0001_as1213 {
        description "Pipe for AS1213 - HEAnet";
        table master4;
        peer table t_0001_as1213;
        import filter f_export_to_master;
        export where ixp_community_filter(1213);
}
```

Examine Client Configuration - Pipe

```
function ixp_community_filter(int peerasn)
{
    # AS path prepending
        ...
    } else if (routeserverasn, 101, peerasn) ~ bgp_large_community then {
        bgp_path.prepend( bgp_path.first );
    }
    ...
}
```

— Client AS65503 tagged their prefix: routeserverasn:101:1213

— Route server will prepend the route from AS65503 with 65503 one extra time when advertising to AS1213

Let's Regroup!

- We know what route servers are and why they are needed
- We have an understanding of why they need to be secure
- We have seen how IXP Manager generates configuration
- We've examined the basic structure of that config
- We've examined a client configuration
  - Examined import rules and internal tagging
  - Examined standard route server communities
  - Basic understanding of Bird topology
- And we have a working route server configuration!

Demonstration System

— VirtualBox on my own laptop
— Ubuntu 20.04 VM for IXP Manager (inc. database, etc)
— Ubuntu 20.04 VM for a route server
— 3 x Ubuntu 20.04 VMs for route server clients

Build a Route Server in ~10 Minutes

- `apt install bird2`
- Download IXP Manager's reconfiguration script
- Edit to add URL and API key
- Execute

What are skipping?

- Full commissioning should include:
  - Firewall, monitoring, documentation, ...

Demonstration

- [x] Build the route server
- [x] Show clients connected and routes
- [ ] IPv6 instance
- [ ] Looking glass
- [ ] Community filtering
- [ ] IRRDB filtering
- [ ] RPKI filtering

Demonstration

- [x] Build the route server
- [x] Show clients connected and routes
- [x] IPv6 instance
- [x] Looking glass
- [x] Community filtering
- [ ] IRRDB filtering
- [ ] RPKI filtering

IRRDB Filtering

IRRDB Filtering

IRRDB filtering (and RPKI) used to ensure that a route server participant can only advertise routes that they should be able to advertise.

- LIRs register routes with routing registries.
  - APNIC, RIPE, etc. but also commercial such as RADB
- Quality of records vary greatly
- IRRDB based filtering has been - and is - the standard

IRRDB Filtering - Example Records

```
route:     192.0.2.0/24
descr:     Packet Loss Ltd
origin:    AS65501
mnt-by:    JOE-MNT
source:    APNIC

route:     2001:db8::/32
descr:     Packet Loss Ltd
origin:    AS65501
mnt-by:    JOE-MNT
source:    RIPE
```

IRRDB Filtering - Generating Prefix Lists

```
$ bgpq3 -j as58372
{ "NN": [
    { "prefix": "103.29.204.0\/22", "exact": true },
    { "prefix": "103.29.204.0\/24", "exact": true },
    { "prefix": "103.29.205.0\/24", "exact": true },
    { "prefix": "103.29.206.0\/24", "exact": true },
    { "prefix": "103.29.207.0\/24", "exact": true }
] }

$ bgpq3 -6j as58374
{ "NN": [
    { "prefix": "2402:9100::\/32", "exact": true }
] }
```

IRRDB Filtering - AS Sets

— Important for members with downstream networks
— Currently a gap in RPKI functionality (AS Cones?)
— BGPQ3 and IXP Manager will recursively unwrap AS sets

```
$ whois AS-HEANET
as-set:          AS-HEANET
descr:           Autonomous Systems routed by HEAnet
members:         AS1213, AS2128, AS112, AS42310, AS2850, AS-IEDR
...
```

IRRDB Filtering - IXP Manager

— Database updated every 6 hours via the scheduler
— Route server config updated via the scripts
— Transaction safe - won't trip over each other
— Manually via the UI
— Manually via Artisan (command line tool)

```
$ ./artisan irrdb:update-asn-db -vvv
Aptus: [IPv4: 1 total; 0 stale; 0 new; DB updated] [IPv6: 1 total; 0 stale; 0 new; DB updated]
    Time for net/database/processing: 0.921408/0.010303/0.000834 (secs)
$ ./artisan irrdb:update-prefix-db -vvv
Aptus: [IPv4: 7 total; 0 stale; 0 new; DB updated] [IPv6: 1 total; 0 stale; 0 new; DB updated]
    Time for net/database/processing: 1.100500/0.014494/0.000373 (secs)
```

IRRDB Filtering - IXP Manager

Previous route server filter config when IRRDB was disabled:

```
# Skipping RPKI check -> RPKI not enabled / configured correctly.
bgp_large_community.add( IXP_LC_INFO_RPKI_NOT_CHECKED );

# This ASN was configured not to use IRRDB filtering
bgp_large_community.add( IXP_LC_INFO_IRRDB_NOT_CHECKED );
```

What does it look like now?

IRRDB Filtering - IXP Manager

```
allas = [ 49567 ];

# Ensure origin ASN is in the neighbors AS-SET
if !(bgp_path.last_nonaggregated ~ allas) then {
    bgp_large_community.add( IXP_LC_FILTERED_IRRDB_ORIGIN_AS_FILTERED );
    accept;
}

# Skipping RPKI check -> RPKI not enabled / configured correctly.
bgp_large_community.add( IXP_LC_INFO_RPKI_NOT_CHECKED );

allnet = [ 31.217.240.0/21, 45.154.100.0/22, ... ];

if ! (net ~ allnet) then {
    bgp_large_community.add( IXP_LC_FILTERED_IRRDB_PREFIX_FILTERED );
    bgp_large_community.add( IXP_LC_INFO_IRRDB_FILTERED_STRICT );
    accept;
} else {
    bgp_large_community.add( IXP_LC_INFO_IRRDB_VALID );
}
```

Demonstration

- [x] Build the route server
- [x] Show clients connected and routes
- [x] IPv6 instance
- [x] Looking glass
- [x] Community filtering
- [x] IRRDB filtering
- [ ] RPKI filtering

RPKI

RPKI Filtering

— An IRRDB entry was a prefix and an origin ASN
— RPKI is a cryptographically secure replacement
    — Adds maximum prefix length
— Yields route origin triplets that have been validated

```
( Origin AS, Prefix,          Max Length )
( AS65500,   2001:db8::/32,   /48        )
( AS65501,   192.0.2.0/24,    /24        )
```

RPKI Validators

An RPKI Validator (aka Relying Party software- rely: depend on with full trust) downloads and verifies the global RPKI data set and can be used to feed the resultant data to our route servers.

— NLnetLabs Routinator
— Cloudflare's OctoRPKI
— FORT Validator, OpenBSD's rpki-client, rpki-prover, RPSTIR2

RPKI and IXP Manager

1. You need **two** RPKI validators
   — See https://docs.ixpmanager.org/features/rpki/
2. Add simple config to IXP Manager's `.env`:

```
# IP address and port of the first RPKI local cache:
IXP_RPKI_RTR1_HOST=192.168.140.211
IXP_RPKI_RTR1_PORT=3323
```

3. Enable RPKI for the router(s) in IXP Manager

RPKI and Origin ASNs

- RPKI provides a prefix and an origin AS
- It **does not** provide any information about whether a particular peer should be able to advertise such a prefix and origin ASN
- E.g. if a peer accidentally advertised a Netflix prefix with Netflix's ASN as the origin, it would pass RPKI's test!
- **You cannot have RPKI without the IRRDB origin AS check**

RPKI Filtering and IXP Manager

We had:

```
# IRRDB origin ASN check
...

# Skipping RPKI check -> RPKI not enabled / configured correctly.
bgp_large_community.add( IXP_LC_INFO_RPKI_NOT_CHECKED );

# IRRDB prefix check
```

RPKI Filtering and IXP Manager

We now have:

```
# IRRDB origin ASN check
...

# RPKI test - if it's INVALID or VALID, we are done
if filter_rpki() then accept;

# IRRDB prefix check
```

RPKI Filtering and IXP Manager

```
function filter_rpki()
{
    if( roa_check( t_roa, net, bgp_path.last_nonaggregated ) = ROA_INVALID ) then {
        bgp_large_community.add( IXP_LC_FILTERED_RPKI_INVALID );
        return true;
    }

    if( roa_check( t_roa, net, bgp_path.last_nonaggregated ) = ROA_VALID ) then {
        bgp_large_community.add( IXP_LC_INFO_RPKI_VALID );
        return true;
    }

    bgp_large_community.add( IXP_LC_INFO_RPKI_UNKNOWN );
    return false;
}
```

Demonstration

- [x] Build the route server
- [x] Show clients connected and routes
- [x] IPv6 instance
- [x] Looking glass
- [x] Community filtering
- [x] IRRDB filtering
- [x] RPKI filtering

Recap on "Securing Route Servers"

1. Small prefixes (default is > /24 for ipv4 and /48 for ipv6)
2. Martians / bogons
3. Ensure at least 1 ASN and <= 64 ASNs in path
4. Ensure peer AS is the same as first AS in the prefix’s AS path
5. Prevent next-hop hijacking
6. Filter known transit networks
7. Ensure origin AS is in set of ASNs from member AS-SET
8. RPKI:
   - Valid -> accept
   - Invalid -> drop
   - Unknown -> revert to standard IRRDB prefix filtering

Thanks for listening!

- https://www.ixpmanager.org/
- https://docs.ixpmanager.org/
- https://www.barryodonovan.com/
- @barryo79 on Twitter
- barry.odonovan@inex.ie

Questions?