BGP Communities:
A Guide for Service Provider Networks

Richard A Steenbergen <ras@nlayer.net> nLayer Communications, Inc.
Tom Scholl <tscholl@att.com> AT&T

What are BGP Communities?

- Defined by RFC1997 (August 1996)
- Quite simply, a 32-bit integer which is attached to a BGP route as an optional transitive attribute.
  - AKA: Not required, and exportable between neighbors.
  - Multiple communities can be attached to one route.
- Well-known (hard-coded) communities exist
  - No-Export, No-Advertise, etc.
  - But mostly, the communities and how they are interpreted are defined by each individual network.

How We Use BGP Communities

- To add additional information to a BGP route
  - Any data you can encode into a 32 bit integer
  - From you to others (providing information)
  - From others to you (requesting actions)
- To take action based on that information
  - Alter route attributes on demand
    - Both globally and within your own network
  - Control the import/export of routes

Why use BGP Communities?

- A scalable network ***needs*** them for its own use
  - Be able to identify customers, transits, peers, etc
  - To perform traffic engineering and export controls
  - There is no other truly acceptable implementation
- But customers love using them as well
  - “Power user” customers demand this level of control.
  - Many make purchasing decisions accordingly.
  - Having self-supporting customers doesn’t hurt either.
  - The more powerful you make your communities, the more work it will save you in the long run.

How are BGP Communities used?

- A 32-bit integer isn't always easy to work with
  - More common convention is to split into two 16-bit values
  - First value is intended to define the scope or "target"
    - So you know if this community is "for you" or someone else
    - So two networks don't do conflicting things with the same data
  - Second value is arbitrary data for the targeted network
    - Whatever data you're trying to encode
- For example: 701:1234
    - Intended for AS701
    - Community value is "1234"

Practical Considerations of Communities

- Most routers parse BGP communities as strings rather than integers, using Regular Expressions.
  - Design your community system with this in mind.
  - Think strings and character positions, not numbers.
  - For Example, 1239:1234 can easily be parsed as
    - Field #1, Value 1
    - Field #2, Value 23
    - Field #3, Value 4
  - But can't easily be parsed numerically
    - For example as "larger than 1233".
  - Remember not to exceed 65535 as a 16-bit value.

Types of BGP Communities

- Practical BGP Communities can essentially be classified into two types:
  - Informational tags
    - Communities set by and sent from a provider network, to tell their customers (or other interested parties) something about that route.
  - Action tags
    - Communities set by and sent from a customer network, to influence the routing policies of the provider network.

Informational Communities

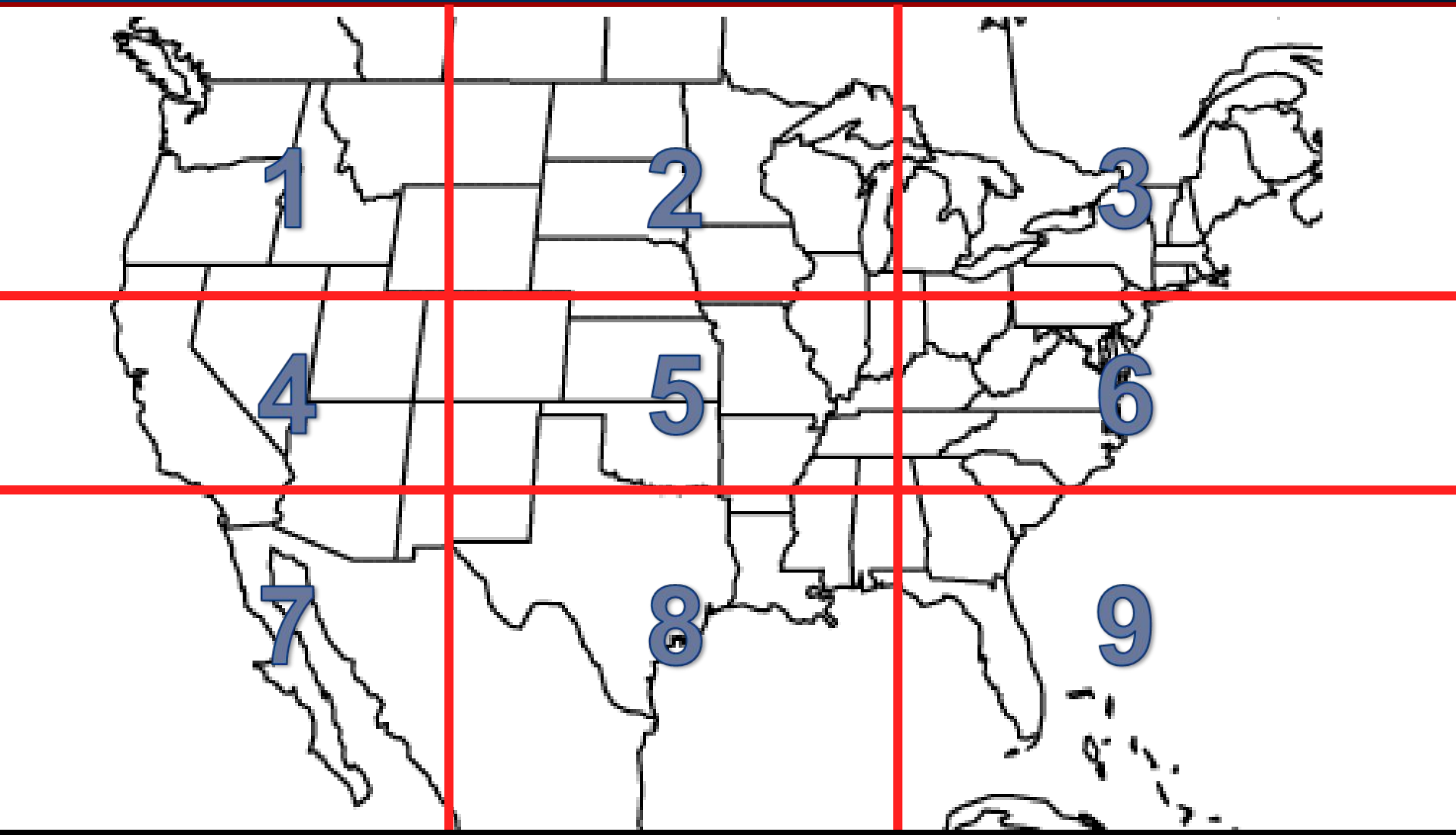
Informational Tags

- Information communities typically focus on
  - Where the route was learned
    - AKA Geographic data (continent, country, region, city, etc)
  - How the route was learned
    - AKA Relationship data (transit, peer, customer, internal, etc)
  - There is no other good way to pass on this data
- This data is then used to make policy decisions
  - Either by you, your customer, or an unknown third party.
  - Exporting this data to the Internet can provide invaluable assistance to third party networks you may never even know about. This is usually a good thing for everyone.

Ways to Encode Community Information

- Encode simple arbitrary data
  - No standards, each network defines its own mapping
    - Which you must then publish somewhere for others to use
  - Ex: Continent (1 = North America, 2 = Europe, etc)
  - Ex: Relationship (1 = Transit, 2 = Public Peer, etc)
- Provide a 9-sectioned overlaid map “region” field
  - Useful for mapping large countries like the US where a region size between state and country is required.
- Other standards based encoding
  - Ex: ISO 3166 encodes Country Codes into 3 digits

Example Region Definitions



Providing Information – Cities/Countries

- As always, the exact design decision depends on your specific network and footprint.
  - Networks in only a few major cities may want to focus on enumerating those cities in a short list.
  - Networks in a great number of cities may want to focus on regional aggregation specific to their scope.
  - European networks may be more focused on enumerating countries than North American networks.
- But whatever you do, ***plan for the future***!
  - Changing your community design after it is already being used by customers may prove impossible.

Ways to Provide Information, Examples

- For example: 1234:TCRPP
  - T Type of Relationship
  - C Continent Code
  - R Region Code
  - PP POP Code
- The community 1234:31311 could be parsed as:
  - Private Peer
  - North America
  - Mid-Atlantic Region
  - Ashburn VA POP Code

Practical Use of Informational Tags

- Make certain you can easily distinguish your Informational Tags from your Action Tags
  - Ex: Make Informational Tags always 5 characters in length, and all other tags to be 4 characters or less.
  - This allow you to easily match Info tags: “1234:.{5}”
- Be prepared to filter communities from neighbors
  - Don’t let ***anyone*** send you Informational tags, these should only be set by you, and you should strip them from all BGP neighbors (customers, transits, peers, etc).
  - Otherwise you have a massive security problem.

Practical Use of Informational Tags

- Consider your company politics first
  - Some networks (think large, with lots of lawyers) consider “relationship” data to be strictly confidential.
    - Some peering contracts do prohibit ANY disclosure of data or even confirming if you are or are not a peer of network X.
    - The vast majority of the Internet just ignores this anyways.
  - Others may be willing to send data to customers only.
- Be prepared to design around them if necessary
  - If there is some data you can’t export, you may want to use two separate Info community tags so you can strip the secret data without impacting non-secrets.

Action Tags

Influencing Routing Policies

- Primarily two main types of actions
  - Export control (do or do not announce the prefix to X)
  - BGP Attribute manipulation
- Typical BGP attributes to be influenced include
  - AS-PATH
  - Local-Preference
  - Multi-Exit Discriminator (MED)
  - Next-Hop Address
  - Anything else you can set in policy (color/weight/etc)

Influencing Routing Policies, Part 2

- Export control actions are typically targetable
  - Apply action only to a specific geographic region
    - E.g. No-export to neighbors in North America, etc.
  - Apply action only to a specific relationship
    - E.g. Prepend to Peers, No-export to Transits, etc.
  - Apply action only to a specific neighbor ASN
    - E.g. Prepend to AS1234
- The most powerful actions are combinations
  - E.g. No-export to Customers in North America

BGP Community Features

Location Specific Actions

Location Specific Actions

- Fairly straightforward, just allow customers to add location codes to their actions
  - Hopefully the same way they were used in Info tags
  - This isn't always possible due to length restrictions
    - But you can often recreate portions of the location info.
  - Reserving 0 for "all locations of this type" works well.
- Examples:
  - 1234:1013 = Action code 1, Continent 1, Region 3
  - 1234:1121 = Action code 1, City code 21
  - 1234:2010 = Action code 2, Continent 1, All Regions

Location Specific Regular Expressions

- Location Specific parsing must be done in regexp
  - You can't break it up into multiple community definitions and then do a logical AND, due to a requirement that you are matching these all in ONE single community.
- A good example:

```
community MATCH_3356_PREPEND_ONE members "^3356:1((000)|(010)|(012)|(116))$";
community MATCH_3356_PREPEND_TWO members "^3356:2((000)|(010)|(012)|(116))$";
community MATCH_3356_PREPEND_THREE members "^3356:3((000)|(010)|(012)|(116))$";
community MATCH_3356_PREPEND_FOUR members "^3356:4((000)|(010)|(012)|(116))$";
community MATCH_3356_MED_ZERO members "^3356:5((000)|(010)|(012)|(116))$";
community MATCH_3356_DENY_EXPORT members "^3356:6((000)|(010)|(012)|(116))$";
community MATCH_3356_FORCE_EXPORT members "^3356:9((000)|(010)|(012)|(116))$";
```

Location Specific Action Caveats

- Many peering requirements specifically require the consistent announcement of prefixes across all locations.
  - Allowing location-specific actions to peers may result in inconsistent announcements.
    - Example: Customer sets No-Export to Peers in Dallas.
    - You are no longer advertising consistently.
- Many networks choose to offer this anyways
  - Tradeoff between customer functionality and “rules”.
  - Not many people seem to be complaining so far.

BGP Community Features

Null Route Community

Null Route Community

- Allow customers to create a more-specific null route via a BGP route tagged with a community
  - One of the more popular “self-help” communities.
- Best implemented with an “anycast null-route”
  - Pick a reserved IP address for null route traffic
  - Static route that IP to null0/discard on every router
    - Or at least most routers, as close to the borders as possible
  - In the Customer import policy:
    - IF the null-route community is set
    - THEN rewrite BGP next-hop to the null-route IP

How to Implement Null Route Community

- In-line with the regular customer BGP sessions
  - Automatically configured/available, no extra work required
  - Harder to manage more-specifics within a prefix-list
    - E.g. only allow up to /24 for normal routes, /32s for null routes
  - Some routers require eBGP-Multihop to rewrite BGP next-hop
  - Easier to accidentally null route something by mistake.
- From a dedicated route-server / blackhole-server
  - No one remembers to configure the extra sessions until an attack
    - Panic call to the NOC ensues, no “self-help” advantage
  - Extra complexity for the customer to manage 2 BGP sessions
  - May be easier for the operator to manage prefix-lists this way
  - May be more flexible for future use of protocols like FlowSpec

BGP Community Features

Per-ASN Communities

Important Caveats – Per-ASN Actions

- A powerful community system allows you to target an action towards a specific ASN
  - For example, No-Export to AT&T/7018
  - This requires matching all of the previous fields, plus a new field for the specific ASN

Per-ASN Implementation Techniques

- Use private ASNs and an arbitrary lookup table
  - E.g. 65001:xxxx = Level(3), 65002:xxxx = Sprint, etc
  - Difficult to deploy, limits the number of target ASNs to 1023
  - Is usually only applied to a handful of the largest peers
  - Changes over time, may impact customers if you recycle #'s
- Use the first half to specify target ASN
  - E.g. 1239:1234 = Apply code "1234" to neighbor ASN 1239
  - Solves the problems above, but introduces some new problems
  - Creates conflicts with communities in use within 1239 itself
    - You can strip 1239:.* before export to 1239 to avoid conflicts
    - But this may impact *your* ability to send 1239 communities
  - May not be obvious to customers which ASNs are supported

Per-ASN Caveats – BGP Replication

- Router control-plane CPUs are much slower and more expensive than their server/desktop cousins
  - 10x the price, and a generation (or more) older tech.
- BGP is made more efficient with update replication
  - Example: 50 BGP neighbors with the same export policy
  - These will be automatically grouped together, the export policy evaluated once, and the results copied 50 times.
- **Per-ASN communities destroy update replication**
  - Now you must evaluate 50 “different” policies 50 times
  - These policies now include string-based RegExps too.
  - CPU suffers accordingly (that is to say, ouch).

BGP Replication and CPU impact

- How harmful is breaking BGP update replication?
  - The increase in policy evaluation CPU is linear
    - 50 peers without update replication = 50x the CPU of 1 peer
  - But this is a huge percentage of the BGP CPU use
    - Even exporting 1 prefix, the entire table must be evaluated
    - Every routing change requires 50 re-evaluations
  - Memory usage also increases linearly
    - Every neighbor now has its own Adjacent RIB Out
- Brand new RP CPUs are able to handle this load
  - But older HW may require upgrades or slow to a crawl
  - May also accelerate the need for upgrades in the future

Advanced Features

Dynamic/Automatic Policies

Important Caveats – Dynamic Policies

- Under both IOS and JUNOS, Per-ASN actions require a unique route-map/policy per ASN
  - Creating a unique policy per ASN can be time-consuming and introduces potential for mistakes.
- There are currently two major approaches to automating Per-ASN BGP Communities
  - Script-based creation of the individual policies
  - Dynamic policies which can evaluate properties of the neighbor they are currently being applied to

Juniper – Automatic Policies

- “Commit scripts” introduced in JUNOS 7.4
  - Essentially small XSLT programs which transform the configuration at commit-time, allowing users to write “programs” and “macros” in their configuration.
  - SLAX alternative syntax introduced in JUNOS 8.2
    - Much more C/Perl-like, less XML-ish. Easy to convert.
- This turns out to be a great way to automate Per-ASN BGP communities.
- Also great for automating “location tags”.

Juniper - Commit Script Example

- In our example we define a “location” macro:
  - system {
    location {
    apply-macro bgp {
    city 16;
    continent 1;
    region 2;
    …
    }
    }
  - And use this data to automatically create the communities and referencing policies.

Juniper – Commit Script Example

- Simple example SLAX pseudo-code:
  - var $location = system/location/apply-macro[bgp];
    var $cont = $location/data[‘continent’]/value;
    var $region = $location/data[‘region’]/value;
    var $city = $location/data[‘city’]/value;

    for-each (protocols/bgp/group/neighbor[peer-as]) {
        call example($asn, $name, $cont, $region, $city);
        …
    }

- Calls this “function” for every configured neighbor

Juniper – Commit Script Examples

- Dynamically creates community expressions that look like this:

```
community MATCH_3356_PREPEND_ONE members "^3356:1((000)|(010)|(012)|(116))$";
community MATCH_3356_PREPEND_TWO members "^3356:2((000)|(010)|(012)|(116))$";
community MATCH_3356_PREPEND_THREE members "^3356:3((000)|(010)|(012)|(116))$";
community MATCH_3356_PREPEND_FOUR members "^3356:4((000)|(010)|(012)|(116))$";
community MATCH_3356_MED_ZERO members "^3356:5((000)|(010)|(012)|(116))$";
community MATCH_3356_DENY_EXPORT members "^3356:6((000)|(010)|(012)|(116))$";
community MATCH_3356_FORCE_EXPORT members "^3356:9((000)|(010)|(012)|(116))$";
```

- And the policies which reference these expressions

```
 term PREPEND_ONE {
    from community MATCH_3356_PREPEND_ONE;
    then as-path-prepend “1234”;
}
term PREPEND_TWO {
    from community MATCH_3356_PREPEND_TWO;
    then as-path-prepend “1234 1234";
}
```

Juniper – Commit Script Example

- Insert the newly created policies into the import/export policy chains

```
for-each (protocols/bgp/group/neighbor[peer-as]) {
        var $import = jcs:first-of(import, ../import, ../../import);
        var $export = jcs:first-of(export, ../export, ../../export);
        var $in_first = $import[position() = 1];
        var $out_first = $export[position() = 1];

        call jcs:emit-change($tag = 'transient-change') {
           with $content = {
              <import> $import;
              <import insert="after" name=$in_first> "AUTOCOMM-" _ peer-as _ "-IN";
              <export> $export;
              <export insert="after" name=$out_first> "AUTOCOMM-“ _ peer-as _ "-OUT";
           }
        }
}
```

IOS XR – Dynamic Policies

- Cisco takes the opposite approach of Juniper, by using dynamic policies in IOS XR

- IOS-XR will let you do:

```
route-policy PEER-OUT
        if  community matches-any (peeras:1, peeras:1111) then
                prepend as-path 7132 1
        endif
        if  community matches-any (peeras:2, peeras:2111)  then
                prepend as-path 7132 2
        endif
```

IOS XR – Dynamic Policies

- Unfortunately, IOS XR will not currently do:

```
route-policy PEER-OUT
        if  community matches-any ( ios-regex '$peeras:1', ios-regex '$peeras:1...1')
                then    prepend as-path 7132 1
        endif
```

- Unable to evaluate “$peeras” variable in Regexp
- This results in a limited feature-set, or some number of unique policies still being required

Important Caveats – Practical Usage

- Be sure to watch for greedy RegExp evaluations
  - E.g. “1239:123.” matches 1239:1234 and 1239:12345
- Protect your expressions with ^ and $ or ()
  - Cisco/IOS
    - ip community-list expanded NAME permit “_(1234:123.)_”
  - JUNOS
    - community NAME members “^1234:123.$”

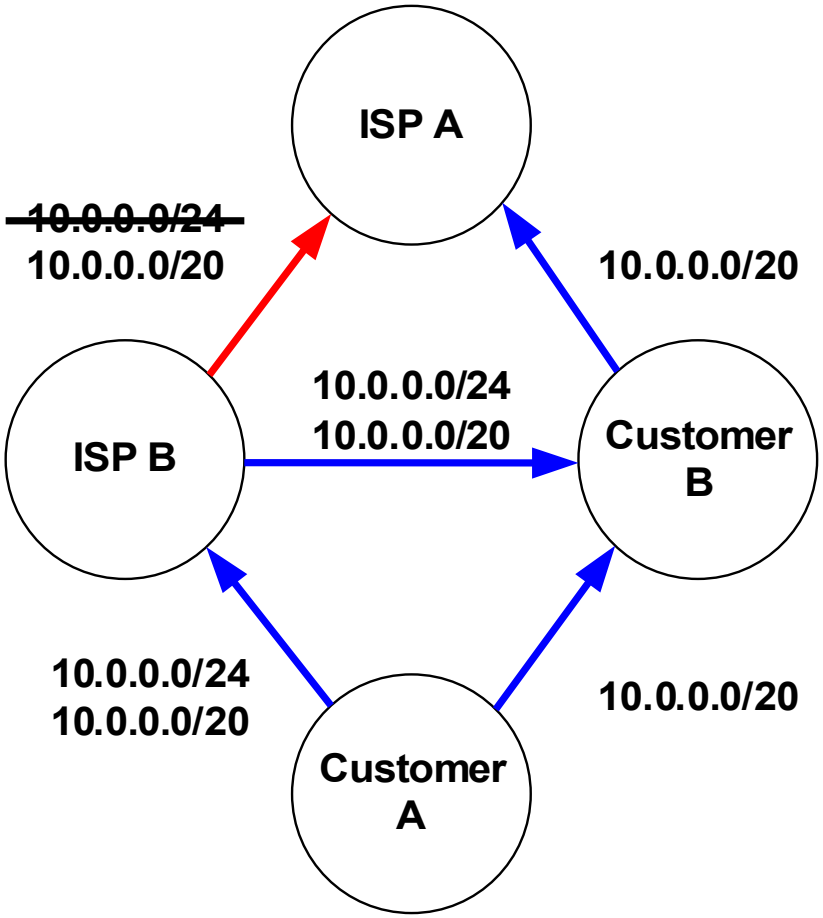
Extra Caveats

Selective Advertisements

Important Caveats – Selective Advertisements

- Allowing for selective policy options can result in bad consequences
  - Customer A is a customer of ISP A and Customer B
  - ISP A and ISP B are settlement-free peers
  - Customer A transmits 10.0.0.0/20 to ISP A and Customer B.
  - Customer A transmits 10.0.0.0/24 to ISP B to not be exported to peers, only customers.
  - Customer B advertises 10.0.0.0/20 to ISP A

Important Caveats – Selective Advertisements (2)



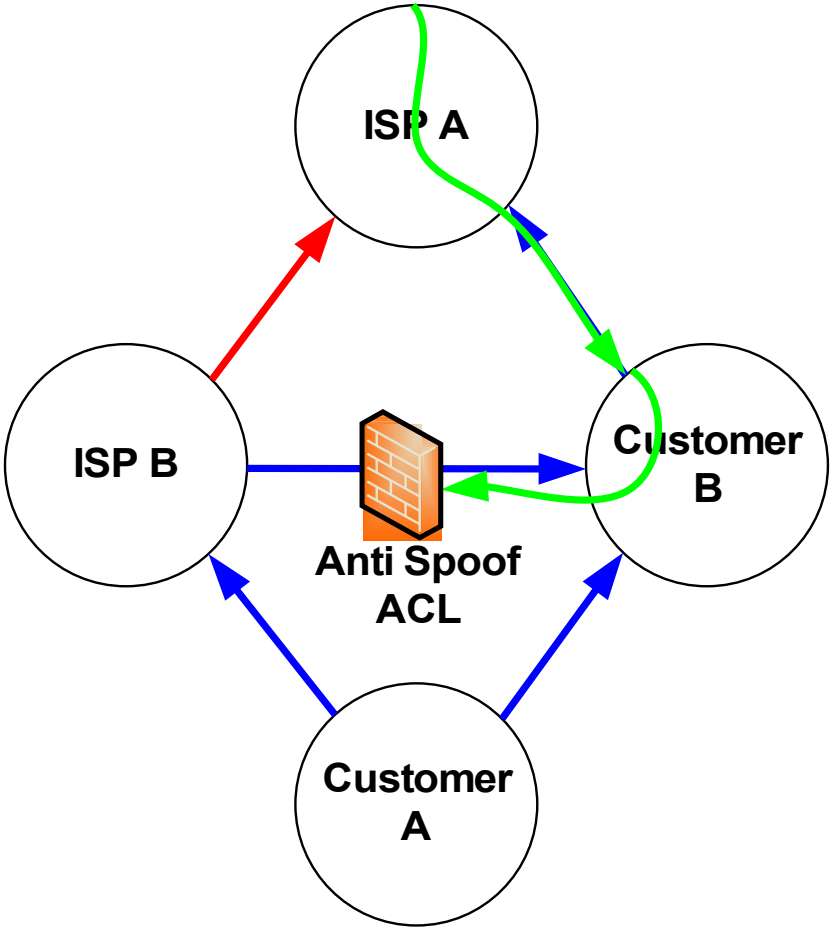
RED – Peering Relationship

BLUE – Customer Relationship

Important Caveats – Selective Advertisements

- ISP A prefers 10.0.0.0/20 via Customer B (default to prefer customers over peer routes)
- Packets from ISP A -> Customer A are routed via Customer B.
- ISP B performs anti-spoofing filters on Customer B, resulting in traffic to Customer A being dropped.

Important Caveats – Selective Advertisements



RED – Peering Relationship
BLUE – Customer Relationship

A Complete Example

Donated by nLayer Communications

Complete Example: Informational Tags

- 5 Digit Informational Tag Format
- 4436:TCRPP
  - T Type of Relationship
  - C Continent Code
  - R Region Code
  - PP POP Code (City Code)

Complete Example: Informational Tags

- Type of Relationship
  - 1 Transit
  - 2 Public Peer
  - 3 Private Peer
  - 4 Customer
  - 5 Internal

- Note 5 is the practical limit due to 0-65535 range.

Complete Example: Informational Tags

- Continent Code (Subcontinents too)
  - 1 North America
  - 2 Europe
  - 3 Asia
  - 4 Australia
  - 5 South America
  - 6 Africa
  - 7 Middle East

Complete Example: Informational Tags

- Region Code (North American Example)
  - 1 North-West (ex: Seattle)
  - 2 North (ex: Chicago)
  - 3 North-East (ex: New York, Boston)
  - 4 West (ex: San Francisco, San Jose)
  - 5 Central (ex: Denver, Kansas City)
  - 6 East (ex: Washington DC)
  - 7 South-West (ex: Los Angeles)
  - 8 South (ex: Dallas, Houston)
  - 9 South-East (ex: Atlanta, Miami)

Complete Example: Informational Tags

- POP Code (City Code)
  - 11 Ashburn VA US
  - 12 New York NY US
  - 13 San Jose CA US
  - 14 Palo Alto CA US
  - 15 San Francisco CA US
  - 16 Chicago IL US
  - 17 Dallas TX US
  - 18 Los Angeles CA US
  - 19 Newark NJ US
  - etc, etc, etc

Complete Example: Action Tags

- Import/Export Action Tag Format
- ASN:A0CR *or* ASN:A1PP
  - ASN Target Autonomous System
    - 65001 Transits
    - 65002 Peers
    - 65003 Customers
  - A Action Code
  - C Continent Code
  - R Region Code
  - PP POP Code (City Code)

Complete Example: Action Tags

- Export Action Codes
  - 1 Prepend AS-PATH with 4436
  - 2 Prepend AS-PATH with 4436 4436
  - 3 Prepend AS-PATH with 4436 4436 4436
  - 4 Prepend AS-PATH with 4436 4436 4436 4436
  - 5 Reset Multi-Exit Discriminator (MED) to 0
  - 6 Set Do-Not-Export
  - 7 Override Do-Not-Export
  - 8 Set Do-Not-Import (applied at IBGP level)

Complete Example: Action Tags

- Local Preference Control Communities
  - 4436:50 Local-pref 50 (Backup Only)
  - 4436:100 Local-pref 100 (Transit preference)
  - 4436:150 Local-pref 150 (Between Transit/Peer)
  - 4436:200 Local-pref 200 (Peer preference)
  - 4436:250 Local-pref 250 (Between Peer/Customer)
  - 4436:300 Local-pref 300 (Customer preference)
  - 4436:350 Local-pref 350 (Above Customer)

Complete Example: Action Tags

- Other Actions
  - 4436:666 Set Next-Hop to Null (Blackhole)
  - 4436:999 Do not export outside of current region

- Code 999 is applied at the IBGP export level, to prevent a route from leaving the region where it was learned.

Complete Example: Some Examples

- 4436:1000 Prepend 1x Globally
- 4436:2010 Prepend 2x in North America
- 4436:3111 Prepend 3x in Ashburn
- 3549:6000 No-Export to AS3549 Globally
- 65001:6020 No-Export to Transits in Europe
- 4436:50 Set Local-Preference to 50
- 4436:999 Do not export out of current region

Links to Implementation Examples

- Example Implementation for Cisco (IOS-XR)
  - http://spf.is-is.ca/xr
- Example Implementation for Juniper
  - http://www.e-gerbil.net/ras/communities/
- Archive of BGP community documentation
  - http://www.onesc.net/communities/

Areas for Future Work

- BGP Extended Communities
- Standardization of Blackhole Community
- Standardization of Geolocation Data
  - See RFC4384

Send questions, comments, complaints to:

Richard A Steenbergen <ras@nlayer.net>

Tom Scholl <tom.scholl@att.com>