NAME

straightenRV − straighten University of Oregon RouteViews table

SYNOPSIS

straightenRV rvtable

straightenRV [-c cutoff:peerfile] [-i|-e ip] ... [-i|-e ip] [-f large-fraction] [-l log-interval] [-m all|2links|paths|peerpaths] [-a] [-s] [-d debug-level] rvtable

PARAMETERS

rvtable The RouteViews table to process. The format of the table is the output of the Cisco *show ip bgp* command. Tables in this format can be downloaded from http://archive.routeviews.org/. *rvtable* should follow one of the following naming conventions:

1. oix-full-snapshot-YYYY-MM-DD-HHMM.dat (e.g. oix-full-snapshot-2001-04-20-1200.dat)

2. bgp_dump_YYYYMMDD (e.g. bgp_dump_20000808)

3. YYYY-MM-DD_HH:MM:SS_PST (e.g. 2001-03-01_03:24:00_PST)

Each of these can optionally be followed by an extenstion indicating what compression was used: ’.gz’ for gzip, ’.bz2’ for bzip2, or no extension for an uncompressed table.

OPTIONS

−c *cutoff:peerfile*

−i *ip*

−e *ip* The -c, -i, and -e options determine the selection of peers straightenRV will consider while parsing the RV table. Any routes in the RV table that are announced by a peer that is not in the selection are ignored. In the absence of any -c, -i, and -e options, all peers are selected. The -i and -e options may appear any number of times. All -e options are processed after -i and -c options.

−c *cutoff:peerfile*

peerfile is a .peer file obtained from a previous run of this script or a handwritten file. If a .peer file from a previous run is used, straightenRV will include those RV peers in the .peer file that have a <=/24 prefix count (number of prefixes of length 24 or less) of at least *cutoff*. straightenRV -c will include at most one RV peer per AS from *peerfile*. To generate a .peer file for use with -c, this script is usually first run without any -i, -e or -c options.

In the case of a handwritten *peerfile*, not all fields need to be included in each line. It is possible to (1) omit the ’len<=24’ and subsequent fields (i.e. only the ’peer ip’ field is present), or (2) omit the ’peer as’ and subsequent fields. In case (1), the RV peer is included in the selection regardless of its <=/24 prefix count or its AS. In case (2) the ’len<=24’ field is present but ’peer as’ is missing. straightenRV -c will only apply the <=/24 prefix count criterion, and omit checking the AS of the RV peer (thus allowing more than one peer per AS).

See the .peer file description further down.

−i *ip* Include RV peer with IP address ’ip’.

−e *ip* Exclude RV peer with IP address ’ip’.

−f *large-fraction*

Defines what a large peer is. An RV peer is considered ’large’ if its <=/24 prefix count is at least ’large-fraction’ of the highest <=/24 prefix count of any one of the RV peers. The default is 0.9.

−l *log-interval*
straightenRV prints some info on stderr every log-interval lines while reading rvtable. The default is 125000.

−m all|2links|paths|peerpaths
conserve memory usage by disabling one or all of the following: generation of the dual AS graph (**2links**), statistics on AS paths (**paths**), statistics on combinations of RV peers and AS paths (**peerpaths**), or all of the aforementioned (**all**). None of these are essential for atom computation. Only **all** has been tested. Without **-m**, a large amount of memory is needed, around 600000 Kbytes on a FreeBSD 4.2 system, while analysing a RouteViews snapshot of February 2003. Using **-m all** reduces memory usage to around 180000 Kbytes. Any data that is unavailable due to these options is replaced by the word "disabled". Furthermore, the .asp output file will not be generated if **-m all** or **-m paths** is passed, and the .as2lk file will not be generated if **-m all** or **-m 2links** is passed.

−a turns on assertions

−s skip lines containing an AS path loop that starts at the first AS (for backward compatibility). See AS Path Normalisation below.

−d *debug-level*
verbosity of debugging output, 0 for no debugging output, 1 (default) for normal debugging output, 2 for all debugging output. Level 2 is not recommended for large data sets!

DESCRIPTION

straightenRV massages a RouteViews table for further processing by findBgpAtoms. In addition it produces a number of files containing statistics, and a ’full’ version of the RV table. The full RV table is a standard, easy-to-parse version of the RV table.

straightenRV is often run twice in succession. Among the files produced by the first run is a .peer file containing a prefix count for each RV peer. The second run uses the .peer file of the first run to select only those RV peers that have a certain minimum prefix count (see ’straightenRV -c’):

Finally, findBgpAtoms uses the .peer and .pfasp.gz files produced by straightenRV as its input. If straightenRV was run twice in succession, the output files of the second run should be passed to findBgpAtoms.

To summarise the above:

```
rvtable -> straightenRV -> .peer + .pfasp.gz + ...
rvtable + .peer -> straightenRV -c -> .peer + .pfasp.gz + ...
.peer + .pfasp.gz -> findBgpAtoms -> ...
```

PROCESSING

This section describes processing of ’rvtable’ by straightenRV.

Peer Selection

If -c, -i or -e options were passed, peer selection is used. Any ’rvtable’ lines that do not have a peer in the peer selection as their Next Hop are skipped.

Unparsable Lines

An ’rvtable’ line must have the following properties, or it will be skipped:

The Status Code, Next Hop, Weight and Path fi elds must be present.

The Status Code fi eld must not contain ’i’.

The Network fi eld, if present, must be a well-formed IPv4 prefi x. Its IP address part must not be 0.0.0.0. The prefi x length part may be omitted (i.e. it may be a classful network number).

The Next Hop fi eld must be a well-formed IPv4 address and not be 0.0.0.0.

The Weight fi eld must be ’0’.

Components in the Path fi eld must not be ’0’.

Any repeated ASes in the Path fi eld must either (1) be consecutive (prepending) or (2) be a loop that can be removed by normalisation (see AS Path Normalisation below).

The AS path must be non-empty after normalisation (see AS Path Normalisation below).

AS Path Normalisation

Any prepending (repeated consecutive ASes) is removed. For example, the AS path "1 2 2 5" becomes "1 2 5".

If, after removing prepending, any repeated ASes in the AS path remain, then the AS path contains a loop. Loops are illegal, and are often the result of typos. The next step is to attempt to fi x the AS path by removing the loop. If the AS path cannot be fi xed, the line containing the AS path is skipped. To fi x the AS path, fi rst the looping part of the AS path is identifi ed. This is the part starting at the fi rst instance of any repeated AS up until the last instance of any repeated AS. The line is skipped if the looping part begins and ends with a different AS number or, if the -s option was given, the looping part starts at the fi rst AS in the AS path. The AS path is fi xed by replacing the looping part by the fi rst AS in the looping part.

For example, the AS path "1 2 3 2 5" is fi xed and becomes "1 2 5". The following AS paths are rejected: "1 2 3 2 3 4" (looping part begins with 2 and ends with 3), "1 2 3 2 4 5 4" (looping part begins with 2 and ends with 4).

Finally, any AS set in the AS path is normalised to become an ordered, comma-separated list of set elements in which each element appears only once. Any brackets (’()’) or braces (’{}’) are removed. Elements that are ’0’ are also removed. If the AS set is empty after normalisation, the AS set is deleted from the AS path. For example, the AS set {3,2,1,0,2} becomes 1,2,3, the AS set {0,0} becomes empty and is therefore deleted from the AS path.

Adding Prefix Length

A prefi x length is added to Network fi elds that do not have one (classful network numbers). There are two ways to determine the prefi x length of a classful network number: 1) look at the network class, and 2) look at the number of trailing ’0’ bytes. We take the longer prefi x length of the two.

FILES

straightenRV generates several output fi les. All output fi les begin with the same string, which summarises the date of the RV table and how many peers were included (-c and -i) or excluded (-e). The format ’bgpYYMMDDf’ is used if none of the -c, -i or -e options appeared (e.g. bgp010420f’; f stands for ’full’). Otherwise the string used is of the form ’bgpYYMMDDpP’ (e.g. ’bgp010901p37’) or

’bgpYYMMDDpPeE’ (e.g. bgp010901p37e1). ’P’ is the number of peers that were included by -c and -i. ’E’ is the number of peers excluded by -e. If only one peer results from the -c, -i and -e options, the string is extended with the peer’s IP address (e.g. bgp010901p1.1.2.3.4).

.as

The .as fi le contains the following counts for each AS: origin, transit and peer and degree. The origin count of an AS is the number of times the AS appears in the origin (last) position of AS paths. The peer count of an AS is the number of times the AS appears in the peer (fi rst) position of AS paths longer than one (see below). The transit count of an AS is the number of times an AS appears in a transit (any but fi rst or last) position of AS paths.

For example, in the AS path W X Y Z, W is the peer AS, Z is the origin AS, and X and Y are transit ASes. In the path X Y, X is the peer AS, Y is the origin AS; there is no transit AS. In the path X, X is both origin and peer AS; there is no transit AS. However, to avoid double counting it is only counted as the origin AS.

Note that given an AS path of length greater than one, the AS in peer position is not counted as transit, even though in reality it may act as transit. Counting the AS in peer position as transit would lead to inflated transit counts for peer ASes, since RouteViews learns all AS paths as a customer of these ASes.

The indegree, outdegree and degree of an AS are defi ned by the directed AS graph formed by the AS links in all AS paths. An AS path X-Y-Z contributes the following edges to the AS graph: X->Y and Y->Z. The degree of an AS is the sum of its indegree and outdegree. For example, the AS path X-Y-Z contributes 1 to the outdegrees of X and Y, and 1 to the indegrees of Y and Z. It contributes 2 to the degree of Y and 1 to the degrees of X and Z. The AS path X does not contribute to indegree, outdegree or degree of X at all.

Note that an AS link X-Y contributes only 1 to the indegree and outdegree of Y and X (respectively), no matter how many AS paths it appears in. Therefore, the outdegree of an AS is NOT the same as the sum of its peer and transit counts (see .as fi le). Similarly, the indegree of an AS is NOT the same as the sum of its transit and origin counts (see .as fi le).

Each line in the .as fi le corresponds to one AS. The header line below shows the fi elds in each line. The AS fi eld contains the AS number. The Transit+Origin fi eld is the sum of the origin and transit counts of the AS. The remaining fi elds are the counts described above.

```
AS|Transit+Origin|Peer|Transit|Origin|Degree|Indegree|Outdegree
```

.asdeg

The .asdeg fi le contains four AS degree distributions: an indegree distribution, an outdegree distribution, a (total) degree distribution, and a stub AS indegree distribution. See the .as fi le description for a defi nition of the indegree, outdegree and degree of an AS. A stub AS is an AS that has positive indegree, but a zero outdegree. In other words it is an origin-only AS that announces at least one prefi x. Note that stub ASes can be multihomed.

Format Each distribution is formatted as follows. Each line corresponds to one degree. The header line below shows the fi elds in each line. An example line for degree 0 is also shown.

```
# degree|      asCn|% all ASes|    N(>=X)|P(>=X)*100
      0      11928    83.6700      14256   100.0000
```

Field: degree

The degree for this line.

Field: asCn
The number of ASes that have this degree.

Field: % all ASes
The percentage of all ASes that have this degree. In the case of the stub AS indegree distribution, it is the percentage of all stub ASes that have this degree.

Field: N(>=X)
The number of ASes that have this degree or a higher degree.

Field: P(>=X)*100
The percentage of all ASes that have this degree or a higher degree. In the case of the stub AS indegree distribution, it is the percentage of all stub ASes that have this degree or a higher degree.

The full distribution is followed by a number of statistics about the distribution. Finally, for convenient use with tools such as xmgrace, an extract of the distribution is given in which only the degree and N(>=X) columns are shown.

.aslk

The .aslk file contains a count of each AS link. In AS path X Y Z, there are two links: X-Y and Y-Z.

Each line corresponds to one AS link. The header line below shows the fields in each line. Also below is an example of AS link 702-12371. This AS link occurs 188 times in all AS paths.

```
# From|To    |Count
702    12371  188
```

.asp

The .asp file contains the counts of each (distinct) AS path encountered. Prepending and looping have been eliminated from the paths.

Each line corresponds to one AS path. The header line below shows that the AS path 293 1 7260 26284 has a count of 7.

```
# count|path
     7 293-1-7260-26284
```

.full.gz

The .full.gz file contains the ’rvtable’ BGP table dump in a standard, easy-to-parse format. Whereas the original BGP table dump has a number of ambiguities, in this file those ambiguities have been removed. All potentially useful information from the original BGP table is preserved. Information that should probably not appear in such a table is omitted, e.g. internal information such as local preference.

Format Each line consists of one BGP route. The header line below gives an overview of the fields in each line. Fields are separated by a space. Also below is an example line containing some fictitious (and illegal) values.

```
linenum |stat |prefix              |peer            |metric    |path + origin
     71 *     255.255.255.255/32 255.255.255.255 8          1 2 {3,4} i
```

Field: linenum (width 11)

Line number of the original BGP table dump that this line was derived from.

Field: stat (width 5)

The status code field, a sequence consisting of any of the following characters:

*: valid
>: best
s: suppressed
d: damped
h: history

So far we have only witnessed status code fields of length 1 or 2.

Field: prefix (width 18)

The CIDR prefix of this route, of the form: IPv4/length. In contrast with the ’rvtable’ format the prefix field cannot be empty, nor can the length part be omitted.

Field: peer (width 15)

The next hop of this route, an IPv4 address. In most cases the next hop corresponds to the BGP peer that announced the route.

Field: metric (width 10)

The metric of this route, or ’-’ if missing.

Field: path + origin (variable width)

The AS path and origin code of this route. The AS path *before* normalisation is used (see ’AS Path Normalisation’). The AS path is a space separated sequence of AS numbers and AS sets. Each AS set is a comma separated sequence of AS numbers enclosed by curly braces (’{}’). (Any brackets ’()’ in the original ’rvtable’ file are replaced by curly braces.) As in the ’rvtable’ file, components of the AS path are ordered from peer AS (AS of the peer) to origin AS.

The origin code (separated from the AS path by a space) is one of the following characters:

i: IGP
e: EGP
?: incomplete
-: missing

.log

The .log file consists of messages that may be relevant to interpreting the data in the other files. In particular, any input lines that were skipped and non-obvious modifications that were made to input lines are recorded in this file.

.peer

The .peer file contains a summary of per-peer statistics. Full per-peer statistics are in the .peerstat file. All AS path related statistics are based on distinct AS paths after removing prepending and looping. Each line corresponds to one peer.

Field: peer ip

The IP address of this line’s peer.

Field: len<=24

The number of prefi xes shorter than (or equal to) /24 seen for this peer.

Field: len>24

The number of prefi xes longer than /24 seen for this peer.

Field: tot.pfs

Total number of prefi xes seen for this peer.

Field: peer as

The fi rst AS in AS paths carried by this peer. Usually this is the AS to which the peer belongs; however in some cases (such as exchange points) this need not be so.

Normally, the fi rst AS in any AS path carried by a particular peer should be the same. If this is not the case, 'peer as' is taken from the fi rst AS path encountered.

Field: #as paths

The number of AS paths seen for this peer.

Field: av.plen

The average length of the AS paths seen for this peer.

Field: std.plen

The standard deviation of path lengths seen for this peer.

Field: ent.plen

The entropy of path lengths seen for this peer.

Field: #origins

The number of distinct origin ASes reached by this peer.

Field: #paths/orig

The average number of AS paths per origin AS for this peer.

.peerstat

The .peerstat contains per-peer statistics. There are several sections. All AS path related statistics are based on distinct AS paths after removing prepending and looping.

.peerstat: Per-peer prefi x length distribution

Each line in a peer's prefi x length distribution corresponds to one prefi x length. The format is as follows:

```
# prefi x length|perc<=24|prefi x count|accumulated prefi x count
  24        55.5016     62966    113449
  25              0        94    113543
```

The 'perc<=24' fi eld is 0 in lines for prefi x lengths > 24. It is the prefi x count of this line as a percentage of this peer's total <= 24 prefi x count.

.peerstat: Per-peer path length distribution

Each line corresponds to one path length. The format is as follows:

```
# as path length|as paths of this length
  4             6018
  5             1926
```

.peerstat: Peer-to-origin path diversity distribution

Each line corresponds to one AS path count. The format is as follows:

```
# as path count |origin ases with this path count
  1            10922
  2             2289
```

The second fi eld gives the number of origin ASes that have the path count for that line.

.peerstat: Prefi x length distribution of global prefi xes

Global prefi xes are those carried by all peers. ’All peers’ means the peers selected by means of the -c, -i, and -e options, or, if no such selection took place, it means all peers that were encountered in the RouteViews table. Each line corresponds to one prefi x length. The header line below shows the fi elds in each line.

```
# prefi x length|percentage|prefi x count|accumulated|accum percentage
  8             0.0135       15          15           0.0135
```

.peerstat: Distribution of peer counts of prefi xes

All prefi xes are considered in this section; there is no selection on global prefi xes or on prefi x length. The distribution is formatted as follows. Each line corresponds to one peer count. The header line below shows the fi elds in each line.

```
# peer count|prefi x count
  1          5117
  2          1012
```

The prefi x count gives the number of prefi xes that are carried by the given number of peers.

.pfasp.gz

The .pfasp.gz fi le contains a list of prefi xes, peers and paths used for BGP atoms computation. Each line corresponds to one BGP route. The header line below gives an overview of the fi elds in each line. Fields are separated by a space. Also below is an example line containing some fi ctitious (and illegal) values.

```
# prefi x          |peer           |path
255.255.255.255/32 255.255.255.255 1-2-3,4
```

Field: prefi x

The CIDR prefi x of this route, of the form: IPv4/length. The prefi x fi eld cannot be empty, nor can the length part be omitted.

Field: peer

The next hop of this route, an IPv4 address. In most cases the next hop corresponds to the BGP peer that announced the route.

Field: path

The AS path of this route in canonical format. The AS path is a hyphen separated sequence of AS numbers and AS sets. Each AS set is a comma separated sequence of AS numbers in numerical order. The component of the AS path are ordered from peer AS (AS of the peer) to origin AS. Repeated components (due to prepending or loops) have been removed from the AS path.

.pref

The .pref file contains per-prefix statistics. Each line corresponds to a prefix encountered in the BGP table dump. The header line below gives an overview of the fields in each line. Fields are separated by a space. Also below is an example line containing some fictitious (and illegal) values.

```
# prefix          |origin ases|nPeers|nOriLks|oriLkstats|oriAsStats
255.255.255.255/32 1_2          15     2       3-1:5˜4-2:10 1:5 2:10
```

Field: prefix

The CIDR prefix of the form: IPv4/length. The prefix field cannot be empty, nor can the length part be omitted.

Field: origin ases (note: variable width field)

An underscore separated, sorted list of distinct origin ASes found for this prefix.

Field: nPeers

The number of peers that carry this prefix.

Field: nOriLks

The number of distinct origin links found for this prefix. (In AS path "1 2 3 4", "3-4" is the origin link.)

Field: oriLkstats (note: variable width field)

A tilde ('˜') separated breakdown of counts per origin link for this prefix.

Field: oriAsStats (note: variable width field)

A space separated breakdown of counts per origin AS for this prefix.

.stats

The .stats file contains an overview of all statistics, and statistics on multi-origin prefixes.

.as2lk

Note: the information in this file has not yet been integrated into our research.

The .as2lk file contains the count of each AS 2-link encountered in the AS paths. The AS paths from which AS 2-links are drawn have been stripped of prepending and looping. A 2-link is a combination of three ASes (two links) which appears in some AS path. For example, in AS path V W X Y Z we have the following 2-links: V->W->X, W->X->Y and X->Y->Z.

From the set of 2-links we can construct a 'dual graph'. The vertices of the dual graph consist of all (directed) AS links. A directed edge is drawn from vertex [X->Y] to vertex [Y->Z] if a 2-link X->Y->Z exists in some AS path. Thus, the dual graph shows how AS links are connected by AS paths.

The dual graph can be used to determine connectivity more accurately than a straightforward AS graph constructed from AS links. In the latter, many paths can be followed which do not correspond to AS paths in the BGP data. For example, consider the following typical scenario with three peers P1, P2 and P3:

```
P1-P2-P3
 | |
  RV
```

P1 and P2 are connected to RouteViews (RV). (Note that RouteViews acts as a customer to BGP routers.) In a typical peering relationship, traffic may flow from P1 (and its customers) to P2 (and its customers) and vice-versa, and from P2 (and its customers) to P3 (and its customers) and vice-versa, but not from P1 (and its customers) to P3 (and its customers) or vice-versa. Since RouteViews is a customer of P1 and P2 it will see the following AS paths:

```
P1->P2
P2->P1
P2->P3
```

Since these AS paths have length one, they are also edges in the AS graph. When walking the AS graph, there now appears to be connectivity from P1 to P3 (since P1->P2->P3 is a path in the graph). However as mentioned, in a typical peering relationship, traffic cannot flow that way.

In contrast, the dual graph contains the vertices [P1->P2], [P2->P1] and [P2->P3], but these vertices will not be connected by edges (since the 2-links P1->P2->P3, and P3->P2->P1 do not exist in any AS path).

As an example of edges that do exist in the dual graph, suppose that P2 has a customer C, and C has a customer CC:

```
P1-P2-P3
   |
   C
   |
   CC
```

The following edges will (typically) exist in the dual graph:

```
[P1->P2]->[P2->C]
[P2->C]->[C->CC]
```

Paths that exist in the dual graph are more likely to correspond to actual connectivity, since common constraints for provider/customer and peer/peer relationships are incorporated. However, inferences about how traffic flows remain speculative since:

The dual graph may still contain many paths that are not present in the AS paths in the BGP data.

No distinction is made between different prefixes that are announced.

Not all policy is visible in BGP AS paths.

In addition a number of vertices and edges called ’terminators’ are added. For each AS link X->Z, we add the vertices [0-X] and [Z-0], as well as the following edges:

```
[0-X] -> [X->Z]
[X->Z] -> [Z-0]
```

These terminators allow paths to be followed through the dual graph, starting at a particular AS (e.g. [0-X] for AS X) and ending at a particular AS (e.g. [Z-0] for AS Z). The count for each terminating edge in the data below is the count of the AS link it was derived from.

Each line in the .as2lk file corresponds to an AS 2-link. The header line below shows the fields in each line. An example line in which 2-link 1-3561-19030 has a count of 33 is also shown. Note that 2-link X-Y-Z is represented as the two links of which it is composed: 'X-Y Y-Z'.

```
AS 2-link        |count
1-3561 3561-19030 33
```

SEE ALSO

findBgpAtoms(1)

NOTES

Throughout, the Next Hop field in rvtable is assumed to be the IP address of the RV peer.

Throughout, the first AS in an AS path is assumed to be the AS to which the RV peer that carries the AS path belongs. However in some cases (such as exchange points) this need not be so.

BUGS and TODO

AS sets size 1

These are converted to AS numbers during repeated-AS removal. E.g. {1} is treated as AS 1. However, in the .full.gz file we should preserve AS sets (e.g. not print {1} as '1', but as {1}). For other purposes it is OK to treat {1} as '1'.

Apply to 'show ip bgp' in general

Making straightenRV suitable for 'show ip bgp' in general, rather than just RouteViews. That means local preference and weight should be included in the .full.gz file, and that in the original table, weight might be non-zero (which affects parsing).

AUTHORS

Patrick Verkaik (patrick@caida.org)
Andre Broido: algorithms and scripts before rewrite
Young Hyun: code review

REFERENCES

Atoms web page:
http://www.caida.org/projects/routing/atoms/

Andre Broido, kc claffy, ‘Analysis of RouteViews BGP data: policy atoms’, Proceedings of the Network-Related Data Management workshop, Santa Barbara, May 23, 2001.
http://www.caida.org/publications/papers/2001/NdrmBgp/

Andre Broido, Evi Nemeth, kc claffy, ‘Internet Expansion, Refinement, and Churn’, European Transactions on Telecommunications, January 2002.
http://www.caida.org/publications/papers/2002/EGR/

Andre Broido, kc claffy, ‘Complexity of global routing policies’.

http://www.caida.org/publications/papers/2001/CGR/