NAME

rapath – print traceroute path information from **argus(8)** data.

SYNOPSIS

rapath [**–A**] [**–M [aspath [dist] | asnode]**] [**–m fields**] [**raoptions**] [**--** *filter-expression*]

DESCRIPTION

Rapath reads **argus** data from an *argus-data* source, and generates the path information that can be formulated from flows that experience ICMP responses. When a packet cause the creation of an ICMP response, for whatever reason, the intermediate node that generates the ICMP packet is, by definition, on the path. Argus data perserves this intermediate node address, and **rapath** uses this information to generate path information, for arbitrary IP network traffic. **Rapath** is principally designed to recover traceroute.1 traffic, so that if a trace is done in the network, argus will pick it up and record the intermediate nodes and the RTT for the volleys. However the method is generalized such that it also picks up routing loop conditions, when they exist in the observed packet stream.

Rapath will generate argus flow records that have the src address, dst address and src ttl of the transmitted packet, aggregated so that the average duration, standard deviation, max and min rtt's are preserved. The most accurate estimate of the actual Round-Trip Time (RTT) between a src IP address and an ICMP based intermediate node is the MinDur field. As the number of samples gets larger, the MinDur field approaches the theoretical best case minimum RTT. RTT's above this value, will include variations in network and device delay.

When using the optional racluster.1 style flow descriptors, path information to and from CIDR based network addresses can be calculated, so that traces from and to multiple machines in the subnets can be grouped together.

The output of rapath can be piped into ranonymize.1, in order to share path performance information without divulging the actual addresses of intermidate routers.

RAPATH SPECIFIC OPTIONS

Rapath, like all ra based clients, supports a number of **ra options** including filtering of input argus records through a terminating filter expression. See **ra(1)** for a complete description of **ra options**. **rapath(1)** specific options are:

–A Draw a description of the path with a legend.

–M *pathmodes*

Supported pathmodes are:

node - print a series of nodes that represent the path (default).
addr - print the IP addresses, instead of node labels.
aspath [dist] - print the series of origin AS's along the path. Optional 'dist' adds the ttl range.
asnode - print the series of nodes, preceded with their AS's along the path.

–m *fields*

Specify modifications to the default flow identifiers. Supported fields are:

srcid - the observation domain source identifier.
saddr[/len] - the source address, optionally as a CIDR address.
daddr[/len] - the destination address, optionally as a CIDR address.

INVOCATION

A sample invocation of **rapath(1)**. This call reads **argus(8)** data from **inputfile** and generates any path information, based on src and dst IP addresses, and writes the results to stdout.

```
% rapath -r inputfile

      SrcId      SrcAddr  Dir     DstAddr          Inode sTtl     Mean    StdDev        Max       Min  Trans
192.168.0.68 192.168.0.68  ->  128.2.42.10    192.168.0.1    1  0.000686  0.000037  0.000764  0.000627     18
```

```
192.168.0.68  192.168.0.68  ->  128.2.42.10     10.22.96.1    2  0.009329  0.002719  0.019935  0.007435    18
192.168.0.68  192.168.0.68  ->  128.2.42.10    208.59.246.2   3  0.010686  0.002619  0.020175  0.007698    18
192.168.0.68  192.168.0.68  ->  128.2.42.10   207.172.15.85   4  0.013988  0.007116  0.032652  0.008923    11
192.168.0.68  192.168.0.68  ->  128.2.42.10   207.172.15.67   4  0.010188  0.000218  0.010676  0.009932     7
192.168.0.68  192.168.0.68  ->  128.2.42.10  198.32.118.161   5  0.010865  0.003557  0.019436  0.007937    18
192.168.0.68  192.168.0.68  ->  128.2.42.10    64.57.20.251   6  0.044649  0.008916  0.076137  0.039844    18
192.168.0.68  192.168.0.68  ->  128.2.42.10    64.57.21.146   7  0.056345  0.003985  0.065643  0.053371    18
192.168.0.68  192.168.0.68  ->  128.2.42.10  147.73.16.120    8  0.052594  0.003037  0.061770  0.050151    18
192.168.0.68  192.168.0.68  ->  128.2.42.10  128.2.255.249    9  0.055147  0.002541  0.064620  0.053151    18
192.168.0.68  192.168.0.68  ->  128.2.42.10  128.2.255.212   10  0.051835  0.000326  0.052362  0.051392     9
192.168.0.68  192.168.0.68  ->  128.2.42.10  128.2.255.205   10  0.054236  0.000658  0.055198  0.053028     9
```

The output of rapath is an argus data stream, and can be written to a file, or piped to other programs for processing. The resulting stream is a clustered data stream ordered by the unique " saddr -> daddr " paths.

The next sample invocation of **rapath(1)** prints out a graph of the path information using letters as index, with the node information provided as reference.

```
% rapath -Ar inputfile

192.168.0.68(192.168.0.68::128.2.42.10) A -> B -> C -> {D,E} -> F -> G -> H -> I -> J -> {K,L}
 Node         SrcId       SrcAddr  Dir      DstAddr           Inode sTtl       Mean     StdDev        Max        Min  Trans
   A    192.168.0.68  192.168.0.68   ->  128.2.42.10      192.168.0.1    1   0.000686   0.000037   0.000764   0.000627     18
   B    192.168.0.68  192.168.0.68   ->  128.2.42.10       10.22.96.1    2   0.009329   0.002719   0.019935   0.007435     18
   C    192.168.0.68  192.168.0.68   ->  128.2.42.10     208.59.246.2    3   0.010686   0.002619   0.020175   0.007698     18
   D    192.168.0.68  192.168.0.68   ->  128.2.42.10    207.172.15.85    4   0.013988   0.007116   0.032652   0.008923     11
   E    192.168.0.68  192.168.0.68   ->  128.2.42.10    207.172.15.67    4   0.010188   0.000218   0.010676   0.009932      7
   F    192.168.0.68  192.168.0.68   ->  128.2.42.10   198.32.118.161    5   0.010865   0.003557   0.019436   0.007937     18
   G    192.168.0.68  192.168.0.68   ->  128.2.42.10     64.57.20.251    6   0.044649   0.008916   0.076137   0.039844     18
   H    192.168.0.68  192.168.0.68   ->  128.2.42.10     64.57.21.146    7   0.056345   0.003985   0.065643   0.053371     18
   I    192.168.0.68  192.168.0.68   ->  128.2.42.10    147.73.16.120    8   0.052594   0.003037   0.061770   0.050151     18
   J    192.168.0.68  192.168.0.68   ->  128.2.42.10    128.2.255.249    9   0.055147   0.002541   0.064620   0.053151     18
   K    192.168.0.68  192.168.0.68   ->  128.2.42.10    128.2.255.212   10   0.051835   0.000326   0.052362   0.051392      9
   L    192.168.0.68  192.168.0.68   ->  128.2.42.10    128.2.255.205   10   0.054236   0.000658   0.055198   0.053028      9
```

the path. Because network paths can be divergent, due to routing changes, load balancing, or redirects, multiple nodes can be observed at the same distance along the path. **rapath(1)** uses '{' and '}' to delimit the set of nodes that are observed at the same distance in the path. Letters in the path are references to inode addresses contained in the actual node records.

The next sample invocation of **rapath(1)** prints out just a graph of the path information in two sets of argus data; today's and last month, to highlight how paths change. ASN information is added to the records, to show how **rapath(1)** depicts ASN relationships, using a **-f ralabel.conf(5)** option.

The -q option suppresses the default output of the actual argus record data compiled for each node along the path. The '[' and ']' (brackets) deliniate AS's and will contain the set of nodes that were observed within the same AS.

```
% rapath -f ralabel.conf -qA -r inputfile
192.168.0.68(192.168.0.68::128.2.42.10) A -> [B] -> [C -> {D,E}] -> [F] -> [G -> H] -> [I] -> [J -> {K,L}]

% rapath -f ralabel.conf -qA -r inputfile.last.month
192.168.0.68(192.168.0.68::128.2.42.10) A -> [B] -> [C -> D] -> [E -> F -> G -> {H,I,J,K} -> {L,M,N} -> O -> P] -> [Q -> {R,S}]
```

This next sample invocation of **rapath(1)** prints out a graph of the ASpath, the set of AS's that the network path traversed. The -q option, again is used to suppress the output of the actual node information. Where there is no AS number, possibly due to a private network or an unregistered address space, letters are used to denote the node.

```
% rapath -f ralabel.conf -r inputfile -qA -M aspath
192.168.0.68(192.168.0.68::128.2.42.10) A -> AS30496 -> AS6079 -> AS1257 -> AS11164 -> AS5050 -> AS9
```

This sample invocation of **rapath(1)** prints out a graph of the ASpath, suppressing the output of the actual node information (-q), and printing actual IP addresses, rather than node labels.

```
% rapath -f ralabel.conf -r inputfile -qA -M aspath addr
192.168.0.68(192.168.0.68::128.2.42.10) 192.168.0.1 -> AS30496 -> AS6079 -> AS1257 -> AS11164 -> AS5050 -> AS9
```

This sample invocation of **rapath(1)** prints out a graph of the ASpath, with distance information, suppressing the output of the actual node information (-q). This is the aspath output, but with distances in TTL's for each entry specified.

```
% rapath -f ralabel.conf -r inputfile -qA -M aspath dist addr
192.168.0.68(192.168.0.68::128.2.42.10) 192.168.0.1:1 -> AS30496:2 -> AS6079:3-4 -> AS1257:5 -> AS11164:6-7 -> AS5050:8 -> AS9:9-10
```

This sample invocation of **rapath(1)** prints out a graph of the AS nodal path, suppressing the output of the actual node information (-q).

```
% rapath -f ralabel.conf -r inputfile -qA -M asnode
192.168.0.68(192.168.0.68::128.2.42.10) AS30496:[A -> B] -> AS6079:[C -> {D,E}] -> AS1257:[F] -> AS11164:[G -> H] -> AS5050:[I] -> AS9:[J

% rapath -f ralabel.conf -r inputfile.last.month -qA -M asnode
192.168.0.68(192.168.0.68::128.2.42.10) A -> AS30496:[B] -> AS6079:[C -> D] -> AS3356:[E -> F -> G -> {H,I,J,K} -> {L,M,N} -> O -> P] ->
```

This sample invocation of **rapath(1)** demonstrates how to use CIDR address aggregation, using the -m option, to generate path performance data from a class B subnet, to a class C subnet.

```
% rapath -f ralabel.conf -r inputfile -A -m saddr/16 daddr/24 - srcid 192.168.0.68

192.168.0.68(192.168.0.0/16::128.2.42.0/24) A -> [B] -> [C -> {D,E}] -> [F] -> [G -> H] -> [I] -> [J -> {K,L}]
 Node         SrcId           SrcAddr   Dir           DstAddr            Inode sTtl       Mean     StdDev        Max        Min  Trans
  A    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24        192.168.0.1    1   0.000686   0.000037   0.000764   0.000627     18
  B    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24         10.22.96.1    2   0.009329   0.002719   0.019935   0.007435     18
  C    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24       208.59.246.2    3   0.010686   0.002619   0.020175   0.007698     18
  D    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24      207.172.15.85    4   0.013988   0.007116   0.032652   0.008923     11
  E    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24      207.172.15.67    4   0.010188   0.000218   0.010676   0.009932      7
  F    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24     198.32.118.161    5   0.010865   0.003557   0.019436   0.007937     18
  G    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24       64.57.20.251    6   0.044649   0.008916   0.076137   0.039844     18
  H    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24       64.57.21.146    7   0.056345   0.003985   0.065643   0.053371     18
  I    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24      147.73.16.120    8   0.052594   0.003037   0.061770   0.050151     18
  J    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24      128.2.255.249    9   0.055147   0.002541   0.064620   0.053151     18
  K    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24      128.2.255.212   10   0.051835   0.000326   0.052362   0.051392      9
  L    192.168.0.68     192.168.0.0/16    ->      128.2.42.0/24      128.2.255.205   10   0.054236   0.000658   0.055198   0.053028      9
```

COPYRIGHT

SEE ALSO

ra(1), rarc(5), ralabel.conf(5), argus(8),

FILES

AUTHORS

Carter Bullard (carter@qosient.com).

BUGS