

NAME

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

COPYRIGHT

Copyright (c) 2000-2012 QoSient. All rights reserved.

SYNOPSIS

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

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 preserves 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 /tmp/ra.out
      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  0.000706  0.000055  0.000806  0.000625  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  10.22.96.1  2  0.008560  0.001136  0.009946  0.006942  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  208.59.246.1  3  0.009302  0.002173  0.013861  0.007692  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  207.172.15.92  4  0.010273  0.004480  0.020175  0.007444  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.71.190.9  5  0.008902  0.001298  0.011437  0.007695  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.138.222  6  0.010983  0.005946  0.024175  0.007698  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.132.89  7  0.014322  0.000833  0.015722  0.013444  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.134.158  8  0.014032  0.000412  0.014444  0.013621  2
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.134.154  8  0.013686  0.000000  0.013686  0.013686  1
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.134.150  8  0.014186  0.000254  0.014440  0.013932  2
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.134.146  8  0.016197  0.000000  0.016197  0.016197  1
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.134.129  9  0.013877  0.000204  0.014181  0.013686  4
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.134.133  9  0.013947  0.000000  0.013947  0.013947  1
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.134.141  9  0.015189  0.000000  0.015189  0.015189  1
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.69.135.241  10 0.024223  0.009878  0.046167  0.018183  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  4.49.108.46  11 0.020230  0.000223  0.020685  0.019940  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  128.2.255.249  12 0.023140  0.001388  0.025183  0.021183  6
192.168.0.68  192.168.0.68  ->      128.2.42.10  128.2.255.205  13 0.033801  0.013120  0.046921  0.020682  2
192.168.0.68  192.168.0.68  ->      128.2.42.10  128.2.255.212  13 0.021994  0.000115  0.022193  0.021911  4
```

This sample invocation of **rapath(1)** prints out a graph of the path, suppressing the output of the actual node information (-q). The '[' and ']' (brackets) deliniate AS's and the '{' and '}' contain set of nodes at the same distance.

```
% rapath -qA -r /tmp/ra.out
A -> B -> [C -> D] -> [E -> F -> G -> {H,I,J,K}] -> {L,M,N} -> O -> P] -> [Q -> {R,S}]
```

This sample invocation of **rapath(1)** prints out a graph of the ASpath, suppressing the output of the actual node information (-q). 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 -r /tmp/rapath.out -qA -M aspath
A -> B -> AS6079 -> AS3356 -> 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 -r /tmp/rapath.out -qA -M aspath addr
192.168.0.1 -> 10.22.96.1 -> AS6079 -> AS3356 -> 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 -r /tmp/rapath.out -qA -M aspath dist addr
192.168.0.1:1 -> 10.22.96.1:2 -> AS6079:3-4 -> AS3356:5-11 -> AS9:12-13
```

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 -r /tmp/rapath.out -qA -M asnode
A -> B -> AS6079:[C -> D] -> AS3356:[E -> F -> G -> {H,I,J,K}] -> {L,M,N} -> O -> P] -> AS9:[Q -> {R,S}]
```

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

```
% rapath -r ~/rapath.out -A -m saddr/16 daddr/24 -srcid 192.168.0.68
A -> B -> [C -> D] -> [E -> F -> G -> H -> I -> J -> K] -> [L -> M]
Node  SrcAddr  Dir      DstAddr      Inode sTtl      Mean      StdDev      Max      Min  Trans
A  192.168.0.0/16  -> 128.2.42.0/24  192.168.0.1  1  0.000706  0.000055  0.000806  0.000625  6
B  192.168.0.0/16  -> 128.2.42.0/24  10.22.96.1  2  0.008560  0.001136  0.009946  0.006942  6
C  192.168.0.0/16  -> 128.2.42.0/24  208.59.246.1  3  0.009302  0.002173  0.013861  0.007692  6
D  192.168.0.0/16  -> 128.2.42.0/24  207.172.15.92  4  0.010273  0.004480  0.020175  0.007444  6
E  192.168.0.0/16  -> 128.2.42.0/24  4.71.190.9  5  0.008902  0.001298  0.011437  0.007695  6
F  192.168.0.0/16  -> 128.2.42.0/24  4.69.138.222  6  0.010983  0.005946  0.024175  0.007698  6
G  192.168.0.0/16  -> 128.2.42.0/24  4.69.132.89  7  0.014322  0.000833  0.015722  0.013444  6
H  192.168.0.0/16  -> 128.2.42.0/24  4.69.134.144  8  0.014387  0.000872  0.016197  0.013621  6
I  192.168.0.0/16  -> 128.2.42.0/24  4.69.134.128  9  0.014108  0.000512  0.015189  0.013686  6
J  192.168.0.0/16  -> 128.2.42.0/24  4.69.135.241  10 0.024223  0.009878  0.046167  0.018183  6
```

| | | | | | | | | | | |
|---|----------------|----|---------------|---------------|----|----------|----------|----------|----------|---|
| K | 192.168.0.0/16 | -> | 128.2.42.0/24 | 4.49.108.46 | 11 | 0.020230 | 0.000223 | 0.020685 | 0.019940 | 6 |
| L | 192.168.0.0/16 | -> | 128.2.42.0/24 | 128.2.255.249 | 12 | 0.023140 | 0.001388 | 0.025183 | 0.021183 | 6 |
| M | 192.168.0.0/16 | -> | 128.2.42.0/24 | 128.2.255.192 | 13 | 0.025930 | 0.009400 | 0.046921 | 0.020682 | 6 |

SEE ALSO

ra(1), rarc(5), argus(8),

FILES**AUTHORS**

Carter Bullard (carter@qosient.com).

BUGS