NAME

sc_analysis_dump — dump of traceroute data in a format that is easily parsed.

SYNOPSIS

sc_analysis_dump [**−cCdeghHilMopQrstT**] [**−D** *debug-count*] [**−G** *geo-server*]
[**−S** *skip-count*] [*file ...*]

DESCRIPTION

The **sc_analysis_dump** utility provides a dump of traceroute data in a format that is easily parsed by scripts. Each line output contains a summary of a single trace, and includes the interfaces visited and the delay of each response. The output format is identical to that of sk_analysis_dump from CAIDA, except that it uses the scamper file API to read both arts++ files produced by skitter and warts files produced by scamper. The **sc_analysis_dump** utility only outputs traceroute data; for parsing other types of measurement, use sc_warts2json(1) instead. The options are as follows:

−c disables printing the cycle number in each line of output.

−C disables printing the comments about the output at the top of the output.

−d disables printing the destination address in each line of output.

−D *debug-count*
for each input file stop reading after the specified number of traces.

−e adds the response from the destination to each line of output. Please read the bugs section below.

−g use geographical data from netacuity. Not all builds of **sc_analysis_dump** support this option.

−G *geo-server*
specifies the name of the netacuity server to use.

−h prints a help message and then exits.

−H disables printing the halt fields: why traceroute halted and data for that reason.

−i disables printing the RTT to each hop, and how many tries were required.

−l disables printing the list id in each line of output.

−M prints any MPLS label stack objects embedded in ICMP responses.

−o prints each line of output using the old format from sk_analysis_dump 1.0.

−p disables print path data in each line of output.

−Q prints the IP-TTL from inside the ICMP quotation.

−r disables printing the data associated the response from a destination: the RTT, the TTL of the probe, and the TTL of the response.

−s disables printing the source IP address in each line of output.

−S *skip-count*
skips the defined number of traces from each input file.

−t disables printing the timestamp of when the traceroute began.

−T prints the IP-TTL of the response packet.

OUTPUT

There is one trace per line. Fields are separated by a tab character. The output is structured into header fields (2 to 6), reply fields (7 to 10) corresponding to the response received from the destination, halt fields (11 and 12), and hop fields (beginning at index 13).

1. Key

   Indicates the type of line and determines the meaning of the remaining fields. This will always be 'T' for an IP trace.

2. Source

   Source IP of skitter/scamper monitor performing the trace.

3. Destination

   Destination IP being traced.

4. ListId

   ID of the destination list containing this destination address. This value will be zero if no list ID was provided. A ListId is a 32 bit unsigned integer.

5. CycleId

   ID of current probing cycle. A cycle is a single run through a given list. A CycleId is a 32 bit unsigned integer. For skitter traces, cycle IDs will be equal to or slightly earlier than the timestamp of the first trace in each cycle. There is no standard interpretation for scamper cycle IDs. This value will be zero if no cycle ID was provided.

6. Timestamp

   Timestamp when trace began to this destination.

7. DestReplied

   Whether a response from the destination was received. The character R is printed if a reply was received. The character N is printed if no reply was received. Since skitter sends a packet with a TTL of 255 when it halts probing, it is still possible for the final destination to send a reply and for the HaltReasonData (see below) to not equal no_halt. Note: scamper does not perform this last-ditch probing at TTL 255 by default.

8. DestRTT

   The RTT (ms) of first response packet from destination. This value is zero if DestReplied is N.

9. RequestTTL

   TTL set in request packet which elicited a response (echo reply) from the destination. This value is zero if DestReplied is N.

10. ReplyTTL

    TTL found in reply packet from destination. This value is zero if DestReplied is N.

11. HaltReason

    A single character corresponding to the reason, if any, why incremental probing stopped. S is printed if the destination was reached or there is no halt data. U is printed if an ICMP unreachable message was received. L is printed if a loop was detected. G is printed if the gaplimit was reached.

12. HaltReasonData

    Extra data about why probing halted. If HaltReason is S, the zero is output. If HaltReason is U, the ICMP code of the unreachable message is printed. If HaltReason is L, the length of the loop is printed. If HaltReason is G, the length of the gap is printed.

13. PathComplete

    Whether all hops to destination were found. C is printed if the trace is complete, all hops are found. I is printed if the trace is incomplete, at least one hop is missing (i.e., did not respond).

14. PerHopData

    Response data for each hop. If multiple IP addresses respond at the same hop, response data for each IP address are separated by semicolons:

    IP,RTT,numTries (for only one responding IP) IP,RTT,numTries;IP,RTT,numTries;... (for multiple responding IPs)

    where IP is the IP address which sent a TTL expired packet, RTT is the RTT of the TTL expired packet, and numTries is the number of tries before a response was received from the TTL.

    This field has the value ’q’ if there was no response at a hop.

    If the **-M** option is specified, any MPLS label stack objects embedded in the ICMP response will be included in the following format, and the four fields correspond to each of the fields in a MPLS header.

    ```
    M|ttl|label|exp|s
    ```

    If the ICMP response embeds more than one MPLS header, they are given one at a time, each starting with an M.

    If the **-Q** option is specified, the TTL value found in a quoted IP packet is included with the following format:

    ```
    Q|ttl
    ```

    If the **-T** option is specified, the TTL value of the response packet is included with the following format:

    ```
    T|ttl
    ```

EXAMPLES

The command:

```
sc_analysis_dump file1.warts file2.warts
```

will decode and print the traceroute objects in file1.warts, followed by the traceroute objects in file2.warts.

The command:

```
gzcat file1.warts.gz | sc_analysis_dump
```

will decode and print the traceroute objects in the uncompressed file supplied on stdin.

BUGS

When the **-e** option is used, any unresponsive hops between the last responding router and the destination are not printed, which could imply an IP link where none exists. Use `sc_warts2json`(1) instead.

SEE ALSO

`scamper`(1), `sc_wartsdump`(1), `sc_warts2json`(1)

AUTHORS

`sc_analysis_dump` was written by Matthew Luckie <mjl@luckie.org.nz>. It was derived from CAIDA's sk_analysis_dump program and should behave in an identical manner.