Internet Engineering Task Force
INTERNET-DRAFT
draft-fu-ccamp-traceroute-00.ps

X. Fu, R. Soltwisch
University of Goettingen

June 23, 2003
Expires: Dec 2003

A Proposal for Generic Traceroute Over Tunnels

Status of this Memo

Copyright Notice

Abstract

We identify some issues for generic traceroute for tunnels (tunneltrace): (1) it is possible that some IP hops do not support tunneltrace, (2) for each tunnel wishing to be traced, at least the two end points should support the tunneltrace, (3) tracing message should be able to bypass firewalls and NATs. One possible solution, based on the CASP signaling protocol (stateless mode), is proposed to support generic routetracing over tunnels.

Contents

1 Introduction

UDP is the transport mechanism recommended as the basis for the IETF CCAMP WG towards a generic traceroute tool that can also verify tunnel paths and diagnose tunnel failures. Some protocols, e.g., GTTP [1], are being developed based on UDP.

This draft identifies some issues concerning generic route tracing over tunnels and proposes a solution based on a generic signaling protocol.

2 Some Issues with Transport Support for Generic Traceroute

In addition to the requirements for traceroute over generic tunnels proposed in [2], we find there are some other issues for a generic traceroute tool should consider:

Transition requirements: it would be difficult to have all IP routers support the generic traceroute tool at the same time, thus it can be quite possible some of IP hops do not support the generic traceroute tool. Nevertheless, to enable tunnel tracing, at least tunnel entry and exit points should support the traceroute functionality.

Firewall traversal: some network administrators deploy packet filters which discard UDP or ICMP packets or packets with IP options. The decision for dropping packets these packets might be based on past security incidents. Thus, traceroute based on UDP, ICMP or UDP/raw IP with router alert option may fail.

Transport of generic traceroute messages: it is possible (and adopted by most of existing proposals) to use UDP end-to-end addressing for the traceroute messages. However there are some potential problems, for instance:

1) collecting information about tunnels and nodes along the path might exceed the path MTU size. This might cause fragmentation and reassemply.

2) Routing assymmetry requires that tunnnel tracing to be initiated by both end points to have information about the path in both directions.

3 Traceroute based on CASP (CASP-T)

3.1 CASP Introduction

The Cross-Application Signaling Protocol (CASP) [3] is a generic signaling protocol for path-coupled (and path-decoupled signaling) between two nodes. Particular path-coupled signaling is attractive for this application.

CASP splits signaling message transport and application specific information. This allows to support different signaling applications to reuse the same underlying transport mechanism. Furthermore it allows to next-hop discovery from signaling message delivery, as shown in Figure 1. The messaging layer is responsible for delivering signaling messages from the initiator to the responder, typically the data source and the data sink, respectively, or the reverse way. The CASP messaging layer is built on existing reliable or unreliable transport protocols, such as TCP, SCTP or UDP, depending on the needs of the application.

The client layer consists of a specialized client, namely next-peer discovery, and any number of specific signaling client protocols, which perform the actual signaling functions, e.g., QoS resource reservation, firewall and configuration, code distribution for active networks, and network diagnostics. Each node can choose its own next-hop discovery mechanism, relying on manual configuration, router advertisements, link state routing protocols, scout protocol [3], or server discovery solutions such as DNS or SLP. A CASP message is simply forwarded to next CASP hop if a CASP node doesn't support the requested client type.

The stateful mode of CASP relies on the soft-state maintained for signaling clients and messaging layer. The stateless mode of CASP, however, does not establish state in IP devices by using record-route objects that enable to traverse the response message to follow the same path in the reverse direction.

Traceroute based on CASP (CASP-T) works as a signaling client layer protocol upon the stateless mode operation of CASP. Depending on the need of transport support, TCP or SCTP is recommended in CASP-T, but a CASP node can decide individually to use UDP if it knows there is no UDP firewall problem and the message size is not larger than MTU to next CASP node.

```
 +- - - - - - - - - - - - - - - - + - - - - - - - - - - - - - - +
 |            +--------------+  |                              |
  Application+--------------+|
 |Signaling+--------------+||   |  +------------------------+ |
  Protocols| CASP Clients ||+      |CASP Next-hop Discovery |
 |         |eg.,CASP-T,QoS|+    |  | Clients (e.g. Scout)   | | CASP
           +--------------+        +------------------------+   Client
 |                ^             |              ^        ^   ^   | Layer
 +- - - - - - - - -| - - - - - -+              |        |   |
=|=================|=========================|========|===|===|=========
                  v                          v        |   |
 |            +-----------------------------------+   |   |   | CASP
              |        CASP  Messaging Layer      |   |   |    Messaging
 |            +-----------------------------------+   |   |   | Layer
                                ^                ^    |   |
 +- - - - - - - - - - - - - - - |- - - - - - - - |- - |- -| - +
                                v                v    v   |
              +-----------------------------+ +-------+ |
              |Reliable Transport(TCP,SCTP...)| |  UDP  | |
              +-----------------------------+ +-------+ |
                                |                  |     v
              +----------------------------------------------+
              |                      IP                      |
              +----------------------------------------------+
```

Figure 1: Cross-Application Signaling Protocol

3.2 CASP-T Overview

In this section, first we describe how the traceroute client for CASP (CASP-T) works, then present its general working environment.

A CASP-T message traverses and records information in all intermediate nodes between a source (initiator) and a destination node in a hop-by-hop way. CASP-T returns information about the entire path with a single roundtrip instead of iteratively requesting more and more information. With regard to tunnels CASP-T works incrementally, which means, an intermediate tunnel immediately. First the signaling message discovers the end points of the tunnel, which then serves as a new source and destination for a subsequent CASP-T session. Once the response message is built the tunnel information is added. Essentially, CASP-T can be used to identify the top-level hops (without looking into tunnels or clouds do not support CASP-T), or all IP hops (including those within tunnels and non-CASP-T clouds) in between the initiator and the destination. Once the destination has been reached the results are sent back to its initiator. When an examination into a tunnel is performed, the results are not directly sent back to the tunnel entrance but forwarded from the tunnel end to the tier-1 next CASP hop for examination. When an error occurs, a CASP_TraceResponse message set with corresponding error code is sent back (in a hop-by-hop fashion) to the initiator on the same level, which is the CASP-T initiator or tunnel entrance in case of tunnel examination.

When the destination on top level is reached a response message is sent back to the top level initiator. This message includes all captured information and provides the entire route information. Classical traceroute [4] can be incorporated within CASP-T to look into each IP hop between two CASP nodes, as described in Section 3.3. This allows CASP-T (and CASP) to be deployed incrementally.

Figure 2 shows an example to illustrate how CASP-T can trace IP hops information inside a tunnel. At node B, CASP uses node D as destination of the next-hop discovery to learn the next CASP hop supporting CASP-T. Then it adds its local information into the traceroute payload of the CASP message, and forwards the message to the next hop. The tunnel exit point, node B, determines that it is not the final destination of the signaling message, and repeats the discovery process (if the next hop is not already known).

```
                 end-to-end addressed signaling message
     +----------------------------------------------------------+
     |                       (a tunnel: B-->D)                  |
     |               %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%           |
  A  V           B   %          C              D   %           E|
+----------+   +-----%----+   +----------+   +-----%----+   +---------+
|  CASP-T  |   |  CASP-T  |   |  CASP-T  |   |  CASP-T  |   |  CASP-T |
+----------+   +----------+   +----------+   +----------+   +---------+
|  CASP-M  |-->|  CASP-M  |-->|  CASP-M  |-->|  CASP-M  |-->|  CASP-M |
+----------+   +----------+   +----------+   +----------+   +---------+
     |            ^    |        ^   |           ^   |           ^
     |            |    |        |   |           |   |           |
     +------------+    +--------+   +-----------+   +-----------+
         hop-by-hop addressing and signaling message delivery
```

Figure 2: Tunnel routetracing using CASP-T

CASP-T is not necessary to be supported in all IP nodes; rather, it can be configured in a more flexible way. Figure 3 shows an example where protocol stacks regarding CASP are configured differently (next-hop discovery is necessary for all CASP nodes; here we omit it in the figure). CASP-T works on the Client Layer of the two-layer CASP architecture. It is based on the functionalities provided by the messaging layer of CASP, which supports secure, congestion-controlled delivery of signaling messages (of any size and for any purpose) between each two CASP aware nodes, while the next CASP node can be discovered by a special discovery client. Figure 3 illustrates a path from node A to B where node A, C and E support CASP-T but node B does not. Some nodes, for instance node D in our example, do not support CASP at all. Here, various nodes reuse the same CASP messaging layer (CASP-M) and different client layers (e.g. CASP-QoS and CASP-T). Interworking with CASP-QoS (e.g. Query message) is possible.

```
                 end-to-end addressed signaling message
     +---------------------------------------------------------+
  A  V               B            C          D            E  |
+------+--------+   +---------+  +------+  +---------+  +------+--------+
|CASP-T|CASP-QoS|   |CASP-QoS |  |CASP-T|  | IP hop, |  |CASP-T|CASP-QoS|
+------+--------+   +---------+  +------+  | no CASP |  +------+--------+
|  CASP-M       |-->| CASP-M  |-->|CASP-M|-------------->|  CASP-M       |
+---------------+   +---------+  +------+  +---------+  +---------------+
     |               ^    |       ^    |                    ^
     +---------------+    +-------+    +--------------------+
        hop-by-hop addressing and signaling message delivery
```

Figure 3: A Possible Configuration

3.3 Incorporating with classical traceroute

It is quite possible not all nodes along the path are CASP aware, therefore, a classical traceroute based on ICMP responses (classical way of traceroute) is incorpated in CASP-T, to trace into such non-CASP-aware clouds along the path. Figure 4 illustrates an example where node A and D speak CASP but node B and C do not. When node A is reached, (classical) traceroute is performed to discover IP addresses and delays for intermediate IP hops between A and D. These results, in addition to information like path MTU and RTT between A and D measured by CASP-T, are added as new trace objects to the CASP-Trace message in node A. This CASP-Trace message in A will be forwarded to node D by CASP-T, and node D will perform a similar operation as node A, until the destination is reached.

3.4 Error Handling

Errors might be caused due to various reasons. One error reason might be a broken tunnel along the path and another reason might be caused by a lost ICMP packet or a firewall dropping packets.

In such cases, an error has to be reported back to the initiator by a CASP-T node along the path discovering the error situation.

3.5 CASP-T Messages

Currently, two types of CASP-T messages are defined: CASP_Trace and CASP_TraceResponse messages.

As shown in Figure 5, a CASP_Trace message consists of one Initiator object and several Trace objects collected along the path.

```
  A              B                     C                 D
+-------+     +--------------+       +------+         +------+
|CASP-T |     |normal IP node|       |  IP  |         |CASP-T|
+-------+     +--------------+       +------+         +------+
 |   ^              ^                   ^               ^   ^
 |   +--------------+                   |               |   |
 |   +----------------------------------+               |   |
 |   +--------------------------------------------------+   |
 |     classical traceroute                                 |
 +----------------------------------------------------------+
          CASP_Trace
```

Figure 4: CASP-T: Using Classical Traceroute

Figure 6 shows the CASP_TraceResponse message format. Trace objects collected along the path are encapsulated as the CASP_TraceResponse payload.

Type: Identify the type of the message. Here, 2 for CASP_TraceResponse. (Later version might include a reverse trace message.)

Version: 4 bit
Identifies the Version of the Protocol. Currently Version = 1.

Error Code: 8 bit to identify protocol errors.
0 - no error
1 - access denied
2 - broken tunnel
3 - destination unreachable

Length: 16 bit
The total length of the message

Tunnel depth: 8 bit
The maximum tunnel depth for the analyses.
0 - top level analysis only.
1 - 255 for the tunnel depth

Classical traceroute flag: 1 bit
0 - CASP node examination only
1 - Classical traceroute enabled

Session ID: 128bit
An identifier to identify the CASP_Trace message. It can be a random number selected by the originator node.

Source and destination ID: 128bit
IPv4 address + identifier or IPv6 address

```
 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Type |Version| Error Code     | Length                        |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| Tunnel depth  |C|             Reserved                         |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Session ID                                                    |
|   //                                                           |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Source ID                                                     |
|   //                                                           |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Destination ID                                                |
|   //                                                           |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Timestamp (date, seconds)                                     |
|            (milliseconds)                                      |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| Initiator Object                                               |
|   //                                                           |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| Authorization Token  |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|nextObj +  Object 1                                             |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|nextObj +  Object 2                                             |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|nextObj +  Object 3                                             |
|   //                                                           |

Type: The message type (here, 1 for CASP_Trace)
C=Classical traceroute flag
```

Figure 5: CASP_Trace message format

Timestamp: 64bit

32 bit for date in seconds since 01/01/1970

32 bit for milliseconds

3.6 CASP-T Objects

```
Trace Object := <ObjectType> <ObjectLength> <ObjectValue>
```

```
 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Type |Version| Error Code    | Length                          |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Session ID                                                     |
|   //                                                            |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Source ID                                                      |
|   //                                                            |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Destination ID                                                 |
|   //                                                            |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| Trace objects as payload                                        |
|   //                                                            |

Type: The message type (here, 2 for CASP_TraceResponse)
```

Figure 6: CASP_TraceResponse message format

```
<ObjectType> := <IPVersion> | <IPAddress> | <LocalHop> | <ErrorCode> |
<delay> | <MTU> | <level> | <tunnel type> | <ClassicalTracerouteObject>
```

Depending on the ObjectType, ObjectValue can be one of the following:

IPVersion: can be either 4 or 6.

IPAddress: the node's IP address.

LocalHop: an increasing number and counts hops. 0 for the first hop and all tunnel entrances

ErrorCode: indicates what kind of error occurs.
0 if there was no error.
1 for timeout,
2 for destination unreachable,
3 for connection interruption,
4 for explicit rejection of the measurement response,
5 authorization required,
6 access denied Other codes are reserved.

Delay: (in ms) shows the time to reach the node

MTU: is the Maximum Transportation Unit which indicates the payload size of IP pages

Level: shows the depth of the tunnel
0 indicates top level
1 one level of tunnel
2+ several level of tunnel in tunnel.

Tunnel type: indicates what kind of tunnel the node belongs to. (For example, IP-in-IP encapsulation, IPsec, GRE, MPLS, L2TP or others)

Classical_Tracroute_Object: The correspondent ObjectValue can be expressed as follows:

```
Value := <IPAddress> <Next CASP_hop> <Number_Of_Hops> <result>
```

The Classical_Traceroute_Object is used for the case that no CASP aware node are in between. In that case the classical traceroute is used. IPAddress is the classical traceroute initiator.

Next CASP_hop is the destination hop is the number of hops the reach the next CASP hop the result is the classical traceroute result.

4 Security Considerations

CASP uses security mechanisms described in [3]. Generic traceroute over tunnels introduces some security threats, such as source authentication, trust relationships between neighboring nodes and between neighboring network domains.

Authorization tokens are suggested in CASP-T to provide protection against such threats. The details are to be investigated in a future version of this document.

5 Open Issues and Discussions

Third-party Tracing: CASP-T can support third-party tracing, by using the tracing source address different from the tracing initiator (with certain changes in operations).

Overhead and Operational Time: Stateless mode of CASP does not introduce significant overhead in the nodes it traverses. As CASP is a generic protocol for various signaling purposes as well, it is possible to reduce the overall overhead if other signaling client protocols are supported. The way that results are forwarded over reliable connections makes that approach more robust against packet losses, and allows it to carry larger size of messages. However, this has to tradeoff with the possibly longer operational time because of the connection establishment. Nevertheless, due to the possibility of reusing existing TCP/SCTP connections (which can be used for both CASP-T and other signaling clients) between CASP hops, the average time for CASP-T can be, on average, low.

Extensibility: use of TCP or SCTP allows larger size of traceroute message, avoiding fragmentation and defragmentation for delivery of the traced data. For example, CASP can be also used for discovery of more information, such as flow-based measurement information in IP nodes, if desired.

6 Acknowledgement

We would like to acknowledge Hannes Tschofenig for his useful comments and discussions with Henning Schulzrinne.

7 Authors' Address

Xiaoming Fu, Rene Soltwisch
University of Goettingen
Institute for Informatics
Lotzestr. 16-18
37083 Goettingen
Germany
EMail: [fu,soltwisch]@cs.uni-goettingen.de

References

[1] R. Bonica, E. Rosen, K. Kompella, D. Meyer, and R.Dube, "Generic tunnel tracing protocol (GTTP) specification," internet draft, Internet Engineering Task Force, Mar. 2003. Work in progress.

[2] R. Bonica, K. Kompella, and D. L. Meyer, "Tracing requirements for generic tunnels," internet draft, Internet Engineering Task Force, June 2003. Work in progress.

[3] H. Schuzrinne, H. Tschofenig, X. Fu, J. Eisl, and R. Hancock, "Casp – cross-application signaling protocol." Internet draft, work in progress, Sept. 2002.

[4] G. Kessler and S. Shepard, "A primer on internet and TCP/IP tools and utilities," RFC FYI 30, 2151, Internet Engineering Task Force, June 1997.