The Traceroute zoo

Lukas Schwaighofer
Advisors: Dirk Haage, Johann Schlamp
Seminar course Innovative Internet Technologies and Mobile Communication, SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: schwaigh@in.tum.de

ABSTRACT

The well known traceroute utility can be used to gather information about the path packets take through a network. Since traceroute is very error-prone, the results may be inaccurate or even unusable. This paper will introduce and discuss some of the different traceroute implementations available.

Keywords

traceroute, load balancers, nmap, traceroute-nanog, Paris traceroute, DisCarte

1. INTRODUCTION

Especially for system administrators, it is very useful to discover the route to a specific destination in order to locate network problems. This task is mostly achieved with the well-known traceroute tool. The Internet Protocol (IP) does not provide any sufficient functionality for route discovery, although two IP features will be discussed in Sections 4.3 and 4.5.1. As illustrated in Figure 1 the only available information is necessarily the first hop of the path. In order to unravel the subsequent hops, the Time To Live (TTL) header field is exploited. While this approach works good enough within small and simple networks, the complexity of the Internet causes inconsistencies. The main reason are non-standard router implementations, firewalls and load balancers [1]. Approaches to deal with these problems and their success will be discussed in this paper.

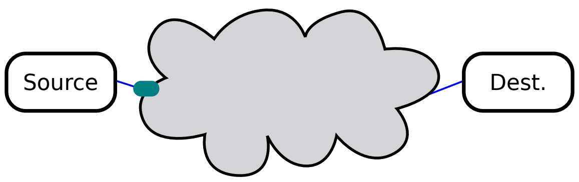

Figure 1: The path to the destination is unknown

2. STANDARD TRACEROUTE

The Time To Live (TTL) field in the IP header plays a central role for traceroute. Each IP packet has a TTL that is decremented by every router before forwarding it. In case the TTL reaches the value of zero, the router will drop the packet instead of forwarding it and send an error message (Time Exceeded, TTL Expired) using the Internet Control Messages Protocol (ICMP) back to the original sender. Dropping packets after they have stayed in the network too long is crucial – otherwise corrupt routing tables could cause a packet to be endlessly routed in a loop, thereby congesting the network.

2.1 Discovering paths

In order to discover the path to a destination D, traceroute exploits the TTL field with the following algorithm:

1. $n := n + 1$
2. Send a packet with TTL n to D
3. Wait for an ICMP error message
4. Save the error message's source IP

The above procedure starts with $n = 0$ and is repeated until a reply from D is received.

The first packet sent by the initiator will have TTL 1. That means the TTL will expire on the first router on the path to D. Assuming this first router isn't already the desired destination D, it will drop the message as described above and return an ICMP error (Time Exceeded, TTL Expired) to the initiator. Now, knowing the first router on the way to D, the initiator sends the next packet (according to the algorithm) with TTL 2. This will yield the IP address of the second router on the way to D.

The whole process goes on, until the received ICMP message actually comes from D. This will usually also be an ICMP error message (Port Unreachable), because most likely D won't be listening on the destination port we were using for that packet.

2.2 Small variations

The above algorithm to discover paths isn't completely specified yet. Most notably, the following details were left out:

- The transport layer protocol used for the package sent to D in the first step
- The destination port used in the first step

Typical traceroute implementations use the User Datagram Protocol (UDP) together with a non-reserved, arbitrary destination port. This port is increased for each subsequent packet in order to relate the sent packets to their responses.

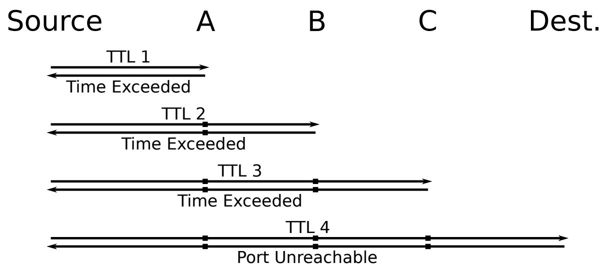

Figure 2: A traceroute example

Technically speaking, it does not matter if the Transmission Control Protocol (TCP) or ICMP is used instead of UDP for sending the packets. The port number may also be chosen from the pool of the well-known ports and doesn't need to be varied for each different packet. However, the packets have to be different somewhere within the first 8 octets of the transport protocol's header, otherwise establishing a relation between the sent packet and the ICMP response (holding only the first 8 octets of the transport protocol's header) can't be established.

Varying transport protocols and ports may seem insignificant, but they play a very important role, for example, when trying to trace through a firewall (discussed in section 3.2).

2.3 An example

Figure 2 shows a trace of four hops to the destination. The traceroute output for that example would look like this:

```
Traceroute to Dest (Dest.IP)
 1   A (A.IP)           x_A ms      y_A ms      z_A ms
 2   B (B.IP)           x_B ms      y_B ms      z_B ms
 3   C (C.IP)           x_C ms      y_C ms      z_C ms
 4   Dest (Dest.IP)  x_Dest ms   y_Dest ms   z_Dest ms
```

In this example output, the letters A, B and C denote the DNS names of the 3 routers that were discovered. With .IP appended, they denote the respective IP address. For each of the routers, three times in milliseconds (ms) are given. To reduce the chance of failure, traceroute actually sends three packets with the same TTL (thus reaching the same router) and measures their round trip time. These three times are given as x, y and z.

3. PROBLEMS

Since traceroute wasn't really the intended use of the TTL field, this approach of finding paths introduces some problems.

3.1 Non-standard routers

One major problem for traceroute is caused by routers that disregard RFC 1812 [3]. In this paper, three common problems caused by such non-standard routers will be discussed.

3.1.1 Unknown routers

This problem exists due to routers discarding packets, without sending an ICMP Time Exceeded message back to the origin. When traceroute tries to discover such a router, it won't receive a response. The result in the output of traceroute will be a star (*) instead of the DNS name and IP address of the router in question.

3.1.2 Missing routers

Routers not decrementing the TTL field at all cause this problem. When the TTL reaches the appropriate value to discover such a router, the next router in the path will instead be discovered. This will result in a router completely missing in the trace. Even worse: There is no way of knowing that the packet has passed an additional router.

3.1.3 Loops in trace

We will call the appearance of the same router twice in the same trace a loop. Routers causing this problem decrement the TTL as they should but fail to discard packets that have reached a TTL of zero. When the TTL of the probe packet reaches the right value to discover this router, the packet will be forwarded once more and the next router on the path (N) will be discovered instead. When traceroute increases the TTL by one, it will again discover the router N, which correctly drops the packet once again. In the resulting trace, the router N appears twice in a row while the non-standard router causing this problem is missing.

3.2 Firewalls

Firewalls play an important role in today's Internet. Almost every corporate network will operate at least one firewall as gatekeeper to their network. As operating systems and applications are – due to their complexity – error prone, using firewalls plays a vital role in security. Even though traceroute itself should not be regarded as an attack, it may give a potential attacker valuable insight into the network internals. Since this information can be used for planning and carrying out attacks, some network administrators deliberately block traceroute with their firewall policy.

Two types of problems when dealing with firewalls will be discussed.

3.2.1 Blocked protocols and ports

A very common way of protecting a network is to limit incoming traffic to certain combinations of protocol and port depending on the destination. A company operating a web server within their corporate network, for example, needs to allow incoming traffic to that particular host on TCP port 80.

This problem can be circumvented by trying different protocol and port combination, as described in Section 2.2. Eventually a combination allowed by the firewall policy will be found and a path to the destination within the firewall protected network can be detected.

3.2.2 Blocked ICMP time exceeded

Firewalls blocking ICMP packets of the type time exceeded deliberately block attempts to gain information about the network with traceroute. Since the ICMP time exceeded packets generated from within the network can't reach the initiator of the trace, there is no way to gain information about the network beyond such a firewall.

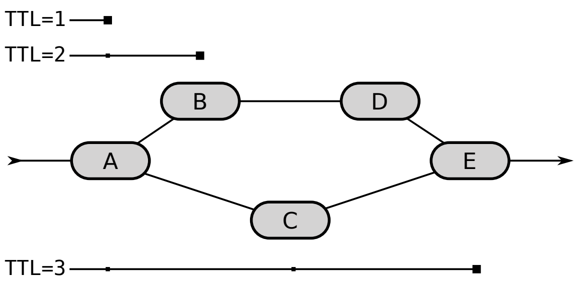

Figure 3: Problem caused by load balancers

3.3 Load balancers

Networks are usually designed to deal with failures of single points without loosing connectivity. This implies, that in most cases multiple possible paths between two distinct nodes exist. In the Internet, routers automatically update their routing tables when they realize that a certain connection is no longer available.

In order to increase performance of the network, backup routes are not only used to avoid fatal failures but also to provide extra bandwidth by distributing the packets over all available routes to the desired destination. [2, 11]. This behavior makes traceroute's life hard: As one trace consists of multiple packets that are individually balanced over the available paths, traceroute may actually detect non-existent links.

An example is given in Figure 3: Node A is a load balancer and in order to reach our desired destination we have to pass router E. For simplicity reasons we assume that A is already our next hop router, reachable with TTL 1. For the second packet (sent with TTL 2), the load balancer decided to forward the packet to router B. The third packet (TTL 3) was sent via router C. The resulting route to E looks like this:
$A \to B \to E$
We will wrongly assume that there exists a link between the routers B and E.

Dealing with this problem is difficult. Traceroute uses 3 packets for each single router to discover. This ensures that it will likely be noticed that something unexpected is happening since packets sent with the same TTL result in responses from different IP addresses. Unfortunately there is no way to decide on a correct path based on the available information. Nevertheless, some traceroute-like implementations that will be discussed in Section 4 have come up with ideas to deal with this problem.

4. DIFFERENT APPROACHES

In this section different route discovery approaches will be introduced. Most important, the differences from standard traceroute will be pointed out and discussed.

4.1 Nmap traceroute

The well known port scanner nmap (Network MAPper) brings it's own traceroute implementation. That way, portscanning and tracing target hosts can be conveniently combined. While nmap uses basically the same approach as standard traceroute, it starts the trace with a high TTL decreasing it step by step to 1 instead of starting with 1 and increasing. This feature is meaningful when tracing whole subnetworks: Once the performed backward trace reaches a point that was passed in a previous trace, there is no need to complete the whole trace any more since the path from that point to the initiator is already known. Thus, for tracing whole networks, the amount of packets required is significantly reduced. For single hosts, this attempt is a little bit slower, because the correct highest TTL value has to be discovered.

4.2 Traceroute-nanog

Traceroute-nanog is a route discovery application from the North American Network Operators' Group (NANOG). This is just another standard traceroute implementation with the following additions:

- Option to lookup Autonomous System (AS) numbers for each hop.
- Change the Type Of Service (TOS) field in the IP header of sent packages to arbitrary values.
- Option to perform a Maximum Transmission Unit (MTU) discovery along the path being traced.
- Can detect the use of the Multiprotocol Label Switching (MPLS) protocol (application described in [4])
- Offers a parallel mode: In order to speed up the trace, multiple packets are sent at the same time. Since some routers limit the amount of ICMP packets per minute, setting this value too high may result in packet loss.

4.3 Traceroute using an IP Option

In an attempt to simplify traceroute, the Internet Engineering Task Force published RFC 1393 [6], introducing an IP option for traceroute. Using this IP option, traceroute is performed in the following way:

1. The initiator sends a packet to the desired destination with the IP traceroute option set.
2. Every router receiving the packet will not only forward it but additionally send a newly generated extra packet back to the initiator.

This approach has two advantages. Firstly, the amount of packets needed for the whole trace is $n + 1$ packets (with n denoting the length of the path). The total number of hops taken by all the packets is $n + \sum_{i=1}^{n} i = \frac{(n+1)^2+n-1}{2}$. Using traceroute, the number of packets required is $2n$ (without doing multiple measurements per router) and the number of total hops is $2 \cdot \sum_{i=1}^{n} i = (n+1)^2 - n + 1$. This comparison is actually in favor of traceroute, because the main reason for doing multiple measurements is not averaging the round trip times. It's all about at least detecting problems described in Chapter 3.

Secondly, the one packet traveling from the source to the destination while triggering an extra packet from each router travels through the network on just one (naturally consistent) path. While there's no way to notice load balancers on the way, the resulting path is always valid and all detected links actually exist.

Unfortunately, this option isn't implemented on any public router. This is mainly due to the security risk such an option represents: A malicious user could trigger a large number of packets sent to a particular host by sending just one packet with forged source address to a far away destination. It is very unlikely that we will see such a convenient way to discover paths in the future.

4.4 Paris traceroute

Paris traceroute's [1] main goal is to do better in the presence of load balancers. The programmers use the fact, that most load balancers use a per-flow approach in their load balancing decision: packets from the same flow are forwarded along the same path. This implementation's main achievement is to control the packet header fields of the probe packets, thereby enabling them to be detected as part of the same flow and routed along the same path by per-flow load balancers.

Through experimentation, the scientists discovered that, apart from some fields in the IP header – namely the TOS, protocol, source address and destination address – the first four octets of the transport layer header are used to identify a packet as being part of a flow. On the other hand, only the first eight octets of that header are encapsulated in the ICMP time exceeded message. So, in order to establish a relation between the probe packets and the ICMP errors, we need to perform modifications in octets 5 to 8 of the transport layer header.

This is easiest when using the TCP protocol: the first 4 octets of the TCP header consist of the source and destination port, the second 4 octets hold the 32 bit sequence number. Thus, varying only the sequence number while keeping source and destination port constant allows all the packets to be regarded as the same flow.

Using UDP things get slightly more difficult: The first four octets also hold source and destination port, but the second four octets hold the length and checksum of the packet. Since neither length nor the checksum can be changed independently from the payload (otherwise the packet is liable to be discarded because of an incorrect checksum), Paris traceroute actually varies the payload in order to produce different checksums. The value of those checksums is ultimately the information used to relate the ICMP time exceeded messages with the sent probe packets.

Using ICMP for sending the probes is the most challenging: The header's first four octets hold the type, code and checksum header fields. The second four octets consist of the identifier and sequence number fields. In order to allow the packets to be identified as part of the same flow, Paris traceroute varies both the Identifier and Sequence number fields in such a way, that the computed checksum remains constant.

According to the authors, Paris traceroute does significantly better than traditional traceroute. According to their paper [1], the number of loops observed was reduced by approximately 84%. Also, this variant of traceroute is less likely to report non-existent links as explained in Section 3.3. The number of faulty links reported is reduced by about 64%.

4.5 DisCarte

DisCarte [8] – standing for Disjunctive Internet Cartographer – is the most advanced route discovery program covered in this paper. It is really more than just another traceroute utility: It can be used to generate topology maps of networks. DisCarte makes good use of traditional traceroute together with an additional feature of the IP protocol: the Record Route (RR) option.

4.5.1 The record route option

The record route option as specified in RFC 791 [7] is older than the IP traceroute option discussed in Section 4.3. It works as follows:

1. The initiator sends a packet with the RR option set to the desired destination.
2. Every router on the path will, while forwarding the packet, add it's own IP address to the IP header.

This approach has a major drawback: The amount of space in the IP header is limited – only up to 9 addresses may be recorded with the record route option. After nine addresses have been added, subsequent routers will just forward the packet without performing any manipulation. This is not as bad as it may seem at first, since using a geographically distributed set of starting points usually offers one point within (or at least close to) the distance of nine hops from the destination.

Surprisingly, only very few routers – according to [8], just a little bit more than 1% – actually filter packets with the RR option set. At the same time, due to the fact that RR is under-standardized, router implementations vary in the way they treat packets with RR set:

- NotImpl: About 9% of the routers don't implement this option at all.
- Departing: About 62% update the RR array at the time packets leave the router, adding the IP address of the router's outgoing interface. This implies that packets arriving with TTL 1 will be dropped before the RR array is updated.
- Arriving: A rather small number of the routers, approximately 7%, already update the RR array at the time the packet arrives, putting the IP address of either the router's outgoing interface or the router's internal loopback interface into the RR array. Those routers also update the RR array of packets that are about to expire.

[8] introduces even more classes of different behavior in respect of the RR option, but introducing them all would go beyond the scope of this paper.

Probe TTL	ICMP source IP	RR Array
1	A	-
2	B	X
3	C	X, Y, Z

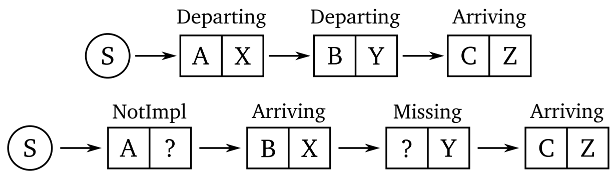

Figure 4: Example taken from [8] showing how the combination of different RR implementations and wrong behavior with respect to the TTL parameter may yield ambiguous results.

4.5.2 *Combining RR and traceroute*

The idea of combining record route with traceroute seems to be simple enough. In practice, combining the information in a meaningful way becomes quite challenging: ICMP time exceeded messages usually contain the address of the router's incoming interface while the RR array mostly contains the address of the router's outgoing interface (see Section 4.5.1). Moreover, due to some abnormalities caused by non-standard routers as described in Section 3.1, the routers discovered by RR and traceroute may differ. For that reason, the actual architecture of the path often becomes ambiguous.

The example in Figure 4 illustrates such a case: The table on top shows the information gained by both the IP address of the returning ICMP packet (traditional traceroute) and the content of the RR array. A, B, C denote the router's incoming and X, Y, Z the router's outgoing interface respectively. From the information given in the table, at least two different topologies for the path can be inferred, illustrated by the two paths drawn. Arriving, Departing and NotImpl refer to the different possible record route implementations as described in Section 4.5.1. Missing refers to the non-standard router implementation introduced in Section 3.1.2.

In order to cross-reference the results, DisCarte uses Disjunctive Logic Programming: Using the results from both the traceroute and the RR trace, all possible paths that do not contain the same router twice in a row are generated. The most likely path (according to the likeliness of each router type) is then selected. In Figure 4 it is easy to see that the upper (smaller) path would be selected, because the combination (Departing, Departing, Arriving) has a higher probability than (NotImpl, Arriving, Departing).

It should be noted, that only a small part of DisCarte was described – the discovery of one single route. The authors of DisCarte evaluate their program against other cartographers (namely Passenger [9] and Rocketfuel [10]) and not against any of the pure traceroute utilities explained in this paper. Therefore no direct comparison and evaluation can be offered at that point. However, according to [8], DisCarte performs significantly better than Rocketfuel, which relies on pure traceroute for route discovery.

5. CONCLUSION

Many scientists have tried dealing with the problem of finding routes and many different implementations for route discovery are available. Nevertheless, taking a closer look, most of these implementations are not very different from standard traceroute. And, just like standard traceroute, all of these implementations have problems when dealing with non-standard situations. Still – over 20 years after the first traceroute application was implemented by Van Jacobson [5], traceroute remains a difficult problem.

6. REFERENCES

[1] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 153–158, New York, NY, USA, 2006. ACM.

[2] B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the internet. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 149–160, New York, NY, USA, 2007. ACM.

[3] F. Baker. Requirements for IP version 4 routers. RFC 1812, Internet Engineering Task Force, 1995. Available from: `http://www.ietf.org/rfc/rfc1812.txt`.

[4] R. Bonica, D. Gan, D. Tappan, and C. Pignataro. ICMP Extensions for Multiprotocol Label Switching. RFC 4950, Internet Engineering Task Force, 2007. Available from: `http://www.ietf.org/rfc/rfc4950.txt`.

[5] V. Jacobson. Traceroute [online]. 1989. Available from: `ftp://ftp.ee.lbl.gov/traceroute.tar.gz`.

[6] G. Malkin. Traceroute using an IP option. RFC 1393, Internet Engineering Task Force, 1993. Available from: `http://www.ietf.org/rfc/rfc1393.txt`.

[7] J. Postel. Internet Protocol. RFC 791, Internet Engineering Task Force, 1981. Available from: `http://www.ietf.org/rfc/rfc791.txt`.

[8] R. Sherwood, A. Bender, and N. Spring. Discarte: a disjunctive internet cartographer. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 303–314, New York, NY, USA, 2008. ACM.

[9] R. Sherwood and N. Spring. Touring the internet in a TCP sidecar. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 339–344, New York, NY, USA, 2006. ACM.

[10] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. *SIGCOMM Comput. Commun. Rev.*, pages 133–145, 2002.

[11] D. Veitch, B. Augustin, R. Teixeira, and T. Friedman. Failure control in multipath route tracing. In *INFOCOM*, pages 1395–1403. IEEE, 2009.