Report for “Reverse traceroute”


7th USENIX conference on Networked Systems Design & Implementation 2010

Introduction

The authors of this paper wanted to complement the already existing traceroute tool, which at the time of the publishing of this paper, until today, is the most widely used tool for network debugging, as well as researching purposes, such as Internet mapping and topology discovery. Specifically, the major motivation to conduct their research and to implement and propose their own system, was traceroute's lack of reverse path information, which, among others, hinders operators and researchers.

The paper's goal, other than addressing traceroute's lack of reverse path information, was to design, implement and propose their own distributed reverse traceroute system. Their system aims to retrieve reverse path information from an arbitrary destination back to the source, without requiring any control of the destination. Their experiments showed that their system achieves an 87% coverage compared to the classic traceroute and its main characteristic is that it builds the reverse path incrementally, hop by hop.

A more precise description of the reverse traceroute system follows. The system is distributed and consists of 10 to 100 vantage points. It does not require any sort of control of the destination and the only requirement is for the destination to respond to probe packets, like classic traceroute. A key feature of the system, is that it relies on the fact that the Internet is destination-based, allowing for the reverse path to be built incrementally. Another key feature of the system, is that an atlas is built beforehand by issuing standard tracertoutes from the vantage points to the source, which is later on used to help guide when to stop building the reverse path. Also, like classic traceroute relies on TTL expired responses, reverse traceroute relies on the support of IP options. In case the destination or an intermediate hop does not support IP options, symmetry is assumed, meaning the next hop is identified based on a standard traceroute issued from the source to the destination. The IP Options that the authors employ to identify hops on the reverse path, are Record Route (RR) and Timestamp-Query (TS) options. In combination with the IP options, the system also uses source spoofing to overcome several obstacles that might occur. Finally, the system operates in a network friendly manner and has no extra requirements from the Internet or the destination itself whatsoever.

IP Options & Techniques

The first technique the system uses to identify the next hop, or even hops, is the Record Route IP option. If this option is enabled, the probe packet records 9 interfaces that it visits, so it starts with 9 empty slots and each interface along the way fills in one of these slots. If the destination is at most 8 hops away from the source, then the first hop on the reverse path will fill in the remaining slot. It is clear that if the destination is closer to the source, then more than one hops on the reverse path will fill in the remaining slots. The whole procedure can be seen in Figure 1 below.
We can easily observe that this technique, while quite useful and straightforward, is quite restricting and limiting. If the destination is further than 8 hops away from the source, this would not work. The way that this limitation is overcome will be explained later on.

The second technique, the Timestamp-Query option, allows a probe to specify up to 4 addresses in a specific order. If the probe visits those interfaces in the exact same order specified, each interface will place a timestamp on the probe. If the order of the interfaces differs, or they do not follow each other, then not all specified interfaces will stamp the probe. This technique involves a bit of guessing, because the system should somehow know which interfaces might follow the given destination. This is solved by using existing network topology to extract candidate sets of routers. The procedure can be clearly seen in Figure 2 below.

Finally, the last technique used, in combination with the ones explained above, is source spoofing. Spoofing is a powerful measurement tool, since it allows to choose the most suitable among the available vantage points of the system to issue probes to the destination, whose response will follow the path back to the source \( S \). The authors note that another alternative could be loose source spoofing, meaning that a vantage point \( V \) would issue probes to the source \( S \), but specifying the destination \( D \) as an intermediate hop. The system does not make use of loose source spoofing though, because such packets are frequently filtered and dropped. In the following examples, it is
presented how source spoofing helps overcome limitations to the techniques discussed above.

In Figure 3, we can see that the source is far away from the destination, so it cannot directly issue a RR option enabled probe to the destination. Instead, a vantage point close to the destination is picked and it issues the probe instead of the source, while spoofing as the source. The result is that the destination will respond to the source, which will see that the interface $R$ is indeed on the reverse path.

Another example is shown in Figure 4. It is clearly depicted that the IP options enabled packets issued by the source hit a probe filter and are dropped. Source spoofing is the solution to this problem as well. A vantage point that can issue such packets is picked and, once again, spoofing as the source, it will manage to make the destination respond to the source, which will reveal the wanted reverse path information.

**System design, architecture & deployment**

The proposed system's design is driven by accuracy, coverage and scalability. In more detail, the system should be accurate, meaning it should yield results as close to a standard traceroute from the destination to the source as possible, it should be able to achieve maximum coverage, in a sense that it should work for any arbitrary destination, independently from ISPs and other factors and, finally, it should be scalable, meaning that it should be selective with its vantage points to be more...
efficient and to cause as little traffic as possible.

Regarding the architecture of the system, it consists of three main components: the vantage points which issue probes, a central controller which coordinates the various vantage points and the source which requests paths. More precisely, when a vantage point starts it connects to the controller, which can then pass probe requests to it. Also, a vantage point is responsible for issuing standard traceroutes back to the source, in order to construct the atlas. The controller receives requests from the source, coordinates the vantage points and collects their measurements to finally report reverse path information. The source itself, issues standard traceroutes, RR and TS-Query pings and it receives the responses.

The current deployment of the system uses 200 active PlanetLab sites as vantage points to construct the atlas back to the source. Because not all of these sites are capable of spoofing, around 60 of them are used as spoofers, along with 14 more spoofing nodes from Measurement Lab machines. Finally, the system makes use of 1200 public traceroute servers for atlas expansion.

**Evaluation**

In order to evaluate the reverse traceroute system, there needs to be some comparison between the results it yields and the results of direct traceroutes from the destinations back to the source. Due to the lack of control of arbitrary destinations, the authors tested their system against hosts they already had control of (PlanetLab sites) and also against public traceroute servers.

The evaluation results showed, that in the median case 87% of hops are measured for the PlanetLab dataset and in 80% of the cases, 78% of the whole path was measured without assuming symmetry. About the public traceroute servers dataset, 74% of the hops are measured compared to standard traceroutes, but 28% of the hops revealed by reverse traceroute, did not appear in the direct traceroute. Finally, for this dataset 98% of the whole path was measured without having to assume symmetry.

The reasons that might cause the paths yielded by the standard and the reverse traceroute to differ vary. To begin with, reverse traceroute, as explained, is sometimes forced to assume symmetry, which is not always a correct assumption. Another reason that might cause inconsistencies between the two methods, is incomplete alias information. This means that the two techniques might discover IP addresses that might appear different, but in reality are aliases for the same router, so are falsely recorded as different hops. Load balancing and contemporaneous path changes during the measurements, is another reason that could yield different but equally valid hops. Finally, another main reason, is the different assumptions about routing. Reverse traceroute assumes destination-based routing, that means if the path from D to S passes through R, from there on it is the same as R to S. Traceroute on the other hand, uses the same (source, destination) pair for each probe, which might cause further discrepancies between the two methods.

Regarding the overhead of the system, it was observed that standard traceroute required around 5 seconds, while reverse traceroute requires around 41. In addition, traceroute on average sends out 45 probes (for a hop count of 15), while reverse traceroute issues roughly 2-3 times more probes.

**Applications of reverse traceroute**

The proposed reverse traceroute system, is intended to be used for the same reasons as traceroute, with a few being debugging path information, topology discovery, measuring one-way link latency etc. The authors present a case study, in order to prove that their system is indeed useful. Briefly, they observed an unjustified latency between two hops in a standard traceroute from one of their machines to a destination. The standard traceroute did not provide any useful
information to justify this latency. In order to investigate the problem further, they used their own system, reverse traceroute, and measured the reverse path back to their machine. The reverse path revealed, showed some circuitousness which seemed like a routing misconfiguration. After private communication with an operator, they verified that, indeed, the detour taken in the reverse path, which caused the unjustified latency, was unintentional. Without the use of reverse traceroute the identification of the problem would not be possible.

Conclusions

The standard traceroute tool certainly is powerful and useful, but its lack of reverse path information hinders network operators and researchers, since they cannot answer important questions. The authors of this paper proposed a distributed system, reverse traceroute, to resolve this issue and provide information about the reverse path from any arbitrary destination back to the source. Its key feature is that it constructs the reverse path hop-by-hop, relying on the fact that the Internet is destination based, until a known node or path is intersected. To actually achieve this, the system employs several auxiliary vantage points and specific techniques, such as certain IP options and limited source spoofing. As the case study described above suggested, the system is quite effective and useful, allowing operators to identify path inflation issues and more. Overall, the system achieves relatively high coverage and accuracy, which can continue to grow as more vantage points are added.