WHITE PAPER

The Shortfalls of Traceroute in Modern Multi-Path Networks

solarwinds msp

THE SHORTFALLS OF TRACEROUTE IN MODERN MULTI-PATH NETWORKS

INTRODUCTION

Traceroute is a fantastically popular network troubleshooting tool. It is second in popularity only to ping. The reason is simple. Traceroute quickly shows you what network devices your traffic is going through to reach a destination, and gives an indication of what each of those devices' performance is. Theoretically, you can quickly tell where your traffic stops, where your traffic is slowed, and what devices are important for this connectivity. Not only is this super helpful, but it's also intuitive. It aligns with our mental model of how networks work. It just makes sense.

If you've been using traceroute though, you know this cheery picture doesn't match reality. Often times, traceroute doesn't work. Or it doesn't show you results that make sense. Traceroute was invented in 1987 and has not kept up with the changes in networking. Today's networks have far more stringent security, consistently have redundancy, and complex hardware architectures that break or confuse traceroute. Understanding these limitations is the key to deriving correct conclusions from the data traceroute presents.

In this document, we will review how traceroute works, take a look at some common problems, learn how to make the most of traceroute, and introduce an alternative solution.

HOW TRACEROUTE WORKS

Routers were not designed specifically to support traceroute. Instead, traceroute found a clever way to use the loop prevention mechanism built into IPv4 to derive its results. Before we can understand how traceroute works, we must understand that loop prevention mechanism.

A router routes by taking a packet from one interface, inspecting its IP header, and forwarding it on through another interface. That's great, but what if the next router decides to take the packet and send it back to the first router? This is called a routing loop. Routing loops can be short, between two routers, or much longer, along a set of many routers. The result is the same; one or several routers have to process the same packet multiple times. Without any controls in place, a router could end up processing the same packet an infinite number of times. Although each processing takes very little effort from the router, if you multiply it by infinity, the router won't be able to handle it. As a result, the router will start dropping packets that it can't handle. This is a big problem and is very dangerous.

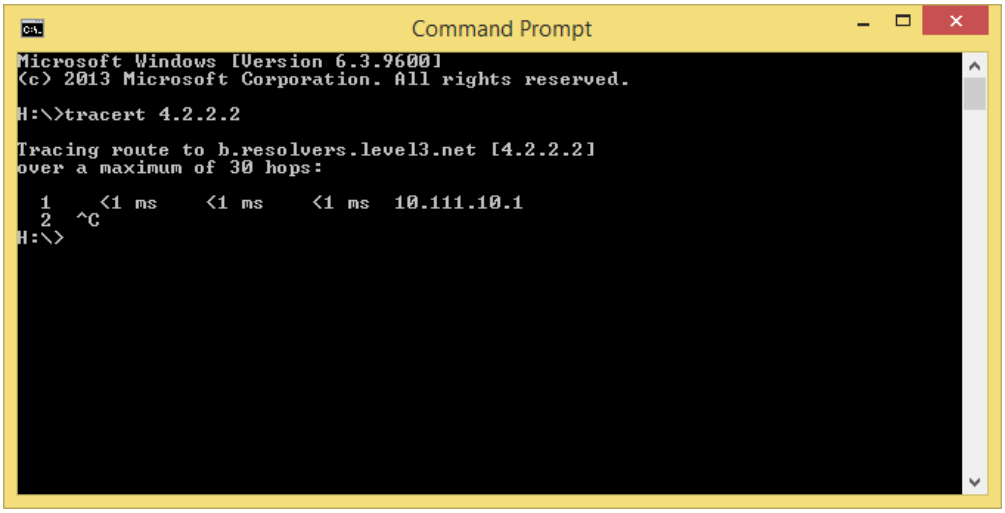
There are many different mechanisms to prevent routing loops. One of them was built directly into the IP packet format. IP packets have a field aptly called "time to live," or TTL (see RFC: https://tools.ietf.org/html/rfc791). Every time a router receives a packet and routes it to another interface, it reduces the TTL by one. This is a way to store the number of times a packet has been routed inside the packet itself. Packets always start their lives from endpoints and have a specific number assigned to the TTL field. The number depends on the operating system, and can be customized, but the number is often 64 or 128. Because the TTL field is 8 bits, the number can't be larger than 255.

So a packet starts with a TTL of 64, and each time it is routed, that TTL goes down by one. If that number ever reaches zero, the router throws away the packet. The router then creates a new packet from itself to inform the source of the packet that the TTL has expired. This is an ICMP Expiry message. As a result, packets can loop, but not an infinite number of times.

Great! What does that have to do with traceroute? Traceroute uses this mandatory function of every router to learn about the network.

Let's walk through how traceroute works, step-by-step in the diagram below.

When you type "tracert" (or traceroute in *nix), traceroute constructs a packet with a destination IP of the destination and a TTL of 1. R1 receives the packet, decrements the TTL by 1, sees that the TTL has reached zero, and throws it away. R1 then creates an ICMP TTL expiry packet sourced from R1 to the Source. When the Source receives this packet, it inspects the source IP and discovers R1's IP. Success! We've discovered hop 1. By measuring the time that has passed between when we sent the packet and when we heard the response, we have also discovered the latency to that node. By default, Windows® sends two more packets just like the first, and the result is a picture like this:



One little known fact is that this is a response from the ingress interface—the interface that received the original packet, and has the ingress interface's IP. Traceroute is therefore a path of ingress interface IPs.

Next, your computer increases the TTL by 1, and repeats. R1 receives the packet. When it decrements the TTL, the result is 1. That is valid, so R1 passes the packet to R2. When R2 receives the packet and decrements the TTL from 1 to 0, it throws the packet away and generates an ICMP reachable back to the source. Thus, the source discovers hop 2. Rinse and repeat for hop 3. This continues until the packet makes it to the final destination. When the packet is able to reach the final destination, the TTL never expires, and the destination itself responds. Your computer receives a response packet where the source IP address matches the destination it initially sent to. This is how your computer determines it has finally reached the end.

That's traceroute. Simple, right? Not quite. Let's take a look at some of the complex problems this method results in.

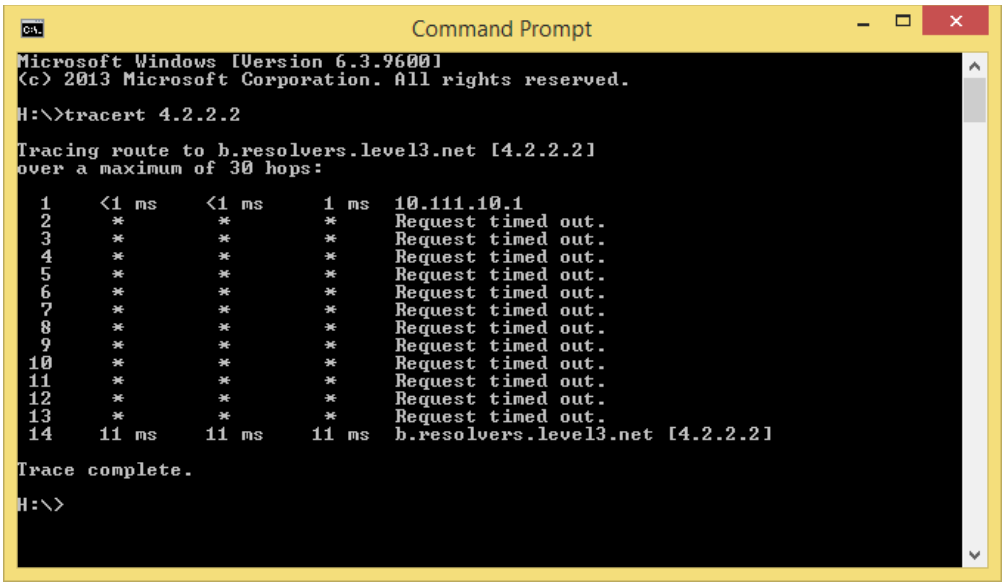
FIREWALL BLOCKS

When traceroute was invented, the internet was much more open. As the internet grew more popular and started to be used for ecommerce, it became more of a target, and security mechanisms had to be put in place to protect it. Today, smart security policy is to allow what you need, and block the rest. Traceroute is not required for users to access their applications, so it is often not allowed through firewalls.

If you do try to allow traceroute through firewalls, it can be a challenge. Most people use the traceroute command included with their operating system. Different operating systems have different implementations of traceroute that vary significantly, most importantly, the type of traffic sent. Some implementations use UDP. Some use ICMP. In rare cases, TCP is used. Some operating systems uses many different ports. Return traffic from transit nodes comes in as a couple of different ICMP types (TTL expiry and unreachable). When the traffic does make it to the endpoint, the endpoint will respond using the source protocol or ICMP, depending on the status of the port you're trying to reach. As you can imagine, getting traceroute to work reliably can be a mess.

The result is that traceroute often does not make it through firewalls. The essence of the problem is that traceroute does not look like end-user traffic and is not needed for users to stay connected with applications.

The result is that using traceroute often looks like this:



Needless to say, this is not helpful.

TIP: When traceroute is not working in your environment, make sure you're allowing the ports your operating system needs outbound, as well as ICMP expiry and unreachables inbound.

CONTROL-PLANE VS DATA-PLANE SLOWDOWNS

Routers are designed to forward traffic. Traceroute is designed to force the router to NOT forward traffic, and respond itself instead. In many cases, routers handle forwarded traffic (data-plane) differently than traffic they have to respond to themselves (control-plane). Traditionally, routers handle forwarding traffic "in hardware," meaning they have dedicated hardware that is designed to perform that exact function. Meanwhile, since the router has to respond directly to very few packets, these are treated as the exception. They are "punted" to a more generic processor that can handle a wide variety of tasks: the CPU. It's possible the performance provided to them is different because these two types of traffic are handled differently. In the event of congestion or other situations strain the router, it is likely that the traffic is handled differently.

When you traceroute, the results you get are what control-plane traffic gets. It may differ from what your applications experience, which is what you care about. A common example of this is traceroute that shows high latency at a node, because its CPU is pegged, but regular traffic does not see high latency.

Fortunately, this is pretty easy to spot.

```
C:\Users\Administrator>tracert 10.0.67.7
Tracing route to 10.0.67.7 over a maximum of 30 hops
  1      5 ms      9 ms      8 ms  10.0.0.1
  2    132 ms    179 ms    145 ms  10.0.12.2
  3     30 ms     56 ms     32 ms  10.0.23.3
  4     54 ms     53 ms     42 ms  10.0.35.5
  5     54 ms     66 ms     53 ms  10.0.56.6
  6     75 ms     80 ms     68 ms  10.0.67.7
Trace complete.
```

We see hop two seems to be very slow. When we're interacting with it directly, we see 132 ms, 179 ms, and 145 ms. BUT, when we send traffic through it to hop 3, we see much lower values. This means that transit traffic is performing fine. In general, it means you can ignore the high values in 2.

The trap here is, upon first look, many engineers who see the traceroute above will conclude that hop 2 is the problem. In fact, that slow down only applies to control-plane traffic and not user traffic.

TIP: When you see a latency or packet loss value for a later hop that is below the value of an earlier hop, you can generally consider that the earlier hop is okay too.

MULTI-PATH

Transit over the internet is often not a simple single path. It's often multi-path.

Figure 1: Single Path Network

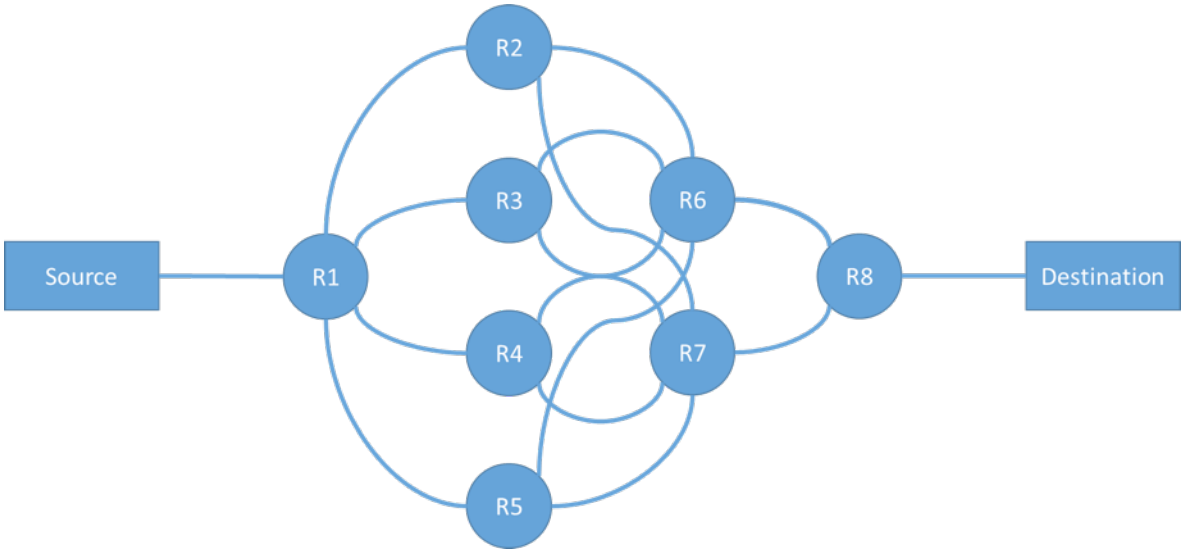


Figure 2: Multi-Path Network

In fact, academic research tells us that today more than 80% of paths over the internet are multi-path and that number is climbing. This presents a distinct problem for traceroute.

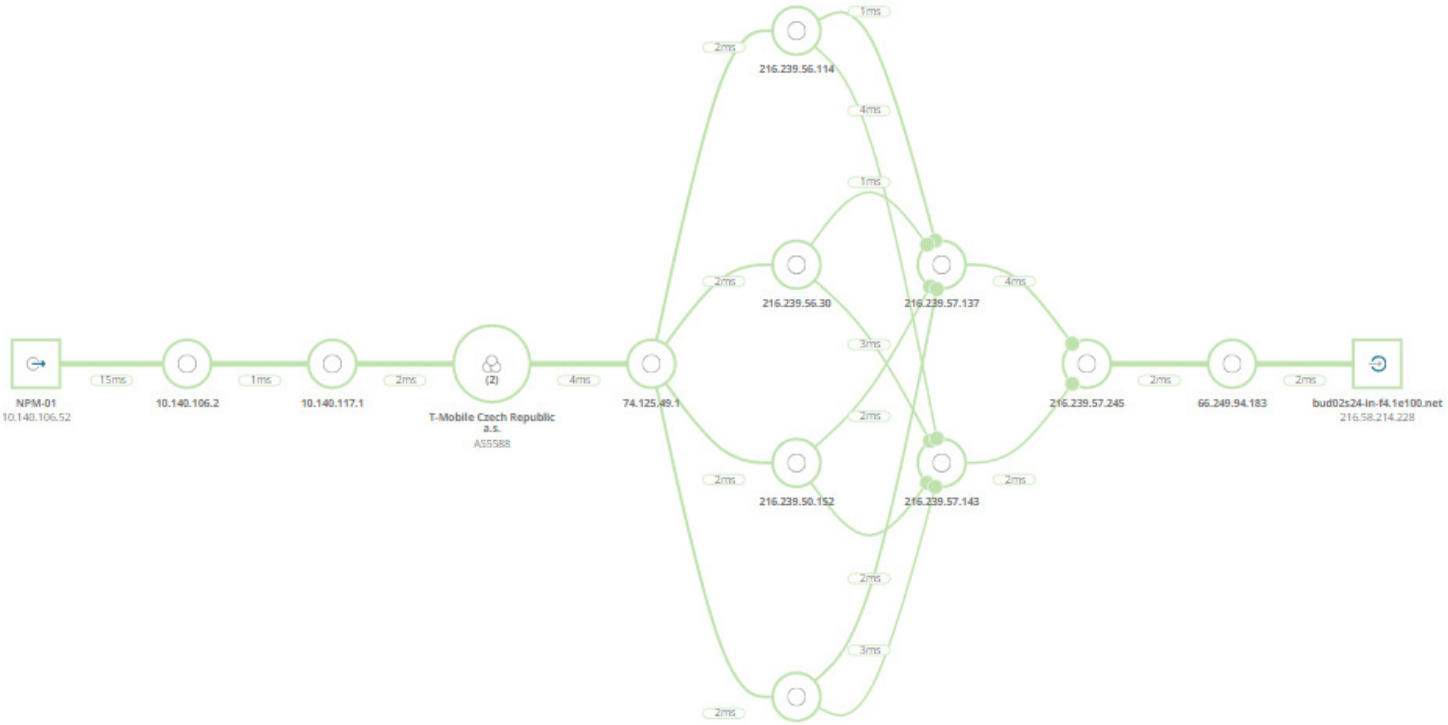
We know that traceroute by default sends a batch of three individual packets for each TTL. That single packet transits some path, and is returned when the TTL reaches zero. Unfortunately, there is very little relationship between one packet and the next. Each packet can go down a different path, therefore traceroute doesn't do a good job with multi-path.

TIP: When you see multiple IPs at a certain hop count, multiple paths are in use. You should be very wary of results.

CONCLUSION

Acknowledging and understanding the limitations of traceroute can help you analyze traceroute results and come up with the best conclusions you can in today's networks. Traceroute remains one of the most ubiquitous network troubleshooting tools. If you're responsible for a network, it's crucial that you understand it well.

The best thing about traceroute is how well it fits our mental model of a network. At SolarWinds, we've spent a lot of time understanding and solving for the many problems with traceroute. We wanted to create something that would provide a clear, intuitive picture of the health of network connectivity from a source to a destination, but would also work in modern networks. We've done just that with NetPath™.



The NetPath feature uses a proprietary discovery method to identify network paths in relatively simple or very complex multi-path networks and then visually displays performance details regardless of the location: on-premises, in hybrid networks, and in the cloud.

© 2018 SolarWinds MSP Canada ULC and SolarWinds MSP UK Ltd. All rights reserved.

This document is provided for informational purposes only. SolarWinds MSP makes no warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information contained herein.

The SolarWinds and SolarWinds MSP trademarks are the exclusive property of SolarWinds MSP Ltd. or its affiliates and may be registered or pending registration with the U.S. Patent and Trademark Office and in other countries. All other SolarWinds MSP and SolarWinds trademarks, service marks, and logos may be common law marks or are registered or pending registration. All other trademarks mentioned herein are used for identification purposes only and are trademarks (and may be registered trademarks) of their respective companies.