WHITE PAPER

How Peering POPs Make Negative Latency Possible

Web downloads with minimal delays

TABLE OF CONTENTS

4 The Classical Approach
4 Peering POPs as an Innovative Solution
5 How Does a Virtual POP work
6 Reasons for Low Latency
6 Optimized Routing
7 Support of HTTP/2
9 Bigger Pipes
10 Test Results
11 Summary

How Peering POPs Make Negative Latency Possible

Web downloads with minimal delays

With each download from the internet, the user waits a certain amount of time between the application sending the initial request, the arrival of the first part of the response, and the completion of the download. This is latency. If the downloaded data is a web page, then there are many objects of different sizes involved—typically HTML code, JavaScript files, several images, and style sheets—and the complete download experience consists out of many individual HTTP transactions (request and response pairs of downloading each object).

When adding a proxy—like a filtering proxy that runs a number of security checks and filters on the web objects—into the flow between the application and the internet server(s), the user can expect an additional delay. This is because there is no longer a direct connection to the server and additional work has to be conducted on that data. The additional delay caused by the filtering proxy is the latency caused by that proxy solution. Most vendors are trying to keep this latency low, so users do not experience a huge penalty by having their data sent through the proxy.

How far can this latency be reduced? Can it be brought down to (almost) zero? Can it even be reduced to below zero? If that were possible, it would be negative latency. Is the negative latency an incremental improvement on the connection or, as implied here, are you able to get an answer before you requested it? A better definition of negative latency is required.

Of course, there is no science fiction proxy solution available that plays by those rules and aims to provide the user with web objects even before sending the request.[1] When we at Skyhigh Security designed our Web Gateway Cloud Service, we explored and invented a number of technologies and researched the effect on the overall web download experience. We asked ourselves how we could create a reliable, always-available solution that also keeps the additional proxy latency as low as possible?

Even we were surprised when our results showed that, in some cases, the download through the proxy was faster than the direct connection. The latency values we measured were showing negative values. These values were not breaking any physical law, but, in comparison to the normal download experience of a direct internet connection, it appeared that the proxy connection was faster. There was no additional latency on top of the normal delay that the user experienced, but the standard latency was lower. It still sounded too good to be true.

How can something be faster even when there is obviously a kind of detour through the proxy? After all, the proxy is even doing more than just routing the data. It has to apply logic on the data such as URL filtering, anti-malware scanning, and more.

The answer has to do with a mix of technologies and the location of our Peering Points of Presence (POP). Latency in a packet-switched network is typically defined as the time elapsed for the round trip of a packet from source to destination and back. This is the sum of the transmission delays for each link, forwarding delays for each router, and the processing and queueing delays for each router or gateway along the route. The following sections will provide a quick introduction to Peering POPs and will discuss those technologies and how they affect latency minimization. The paper will also present some real-life examples of negative latency.

The Classical Approach

When deploying a web Software-as-a-Service (SaaS) solution, most vendors start by placing POPs in as many locations as possible—at least in all major markets. That comes with a great deal of effort, especially because local failover and elasticity in each location leads to significant overprovisioning. Skyhigh Security experimented with that approach some years ago using low-cost hosting providers in all regions, forming the micro-POP approach. The results were not compelling for the reasons mentioned and others.

Peering POPs as an Innovative Solution

A decent but limited number of Peering POPs strategically deployed at major Internet Exchange Points (IPXs) and serving as "Virtual POPs" for several countries around the physical location of the Peering POP can solve requirements for country-specific ingress and egress IP addresses in the same way as a local micro-POP. Speed, latency, and failover build-out are superior in a Peering POP compared to a micro-POP. The exception is a few remote regions with a low user population. Skyhigh Security is actually deploying a mix of Peering POPs and micro-POPs in production today.

Our Peering POPs are deployed at major IPXs. Our analysis shows that almost all traffic between any client and server will cross at least one of those major IPXs in normal, direct routing. This makes them the ideal place to deploy a central proxy server, as we will see below.

This image shows sample locations of 13 physical POPs worldwide:

Figure 1. A sample of Skyhigh Security Peering POPs worldwide.

How Does a Virtual POP Work?

Skyhigh Security is hosting several whole autonomous system number (ASN) networks with a set of IP addresses in one server farm in any physical Peering POP. For example, one set of IP addresses in a block of 17 consecutive addresses is 185.125.224.3 to 185.125.224.19.

By policy, we choose which of the transactions on that server farm will use which of those 17 addresses as an egress address to connect to the internet. Policy could use, for example, incoming proxy address, source address, customer name, user group/name, or any other rule to determine the egress IP address.

Thanks to our peering contracts with major internet players, we can register the IP addresses of our ASN networks in their IP geolocation databases, and that information will be replicated to the IP-to-location services of all other vendors.

Although servers that are physically located close to Frankfurt/Germany are using the above 17 addresses, each of these addresses is officially registered for a server location that is placed in a different city in Europe[2]. Following our example above, the list is as follows:

185.125.224.3 Gdansk

185.125.224.4 Paris

185.125.224.5 Amsterdam

185.125.224.6 Vienna

185.125.224.7 Rome

185.125.224.8 Zurich

185.125.224.9 Prague

185.125.224.10 Copenhagen

185.125.224.11 Madrid

185.125.224.12 Lisbon

185.125.224.13 Budapest

185.125.224.14 Bratislava

185.125.224.15 Brussels

185.125.224.16 Cork

185.125.224.17 Oslo

185.125.224.18 Stockholm

185.125.224.19 Helsinki

As a result, the 13 sample physical locations shown above are showing up as 52 sample locations (physical and virtual) that our customers can leverage.

Figure 2. A sample of virtual and physical POP locations that Skyhigh Security customers can leverage.

Reasons for Low Latency

We have identified three core reasons to maintain a low latency experience and can show how our Peering POP locations offer certain advantages: optimized routing, support of HTTP/2, and bigger pipes.

Optimized routing

Most people assume that physical proximity of clients and servers guarantee low latency, but that is only true if clients and servers are joined by a direct connection on a fast link. If the server is not located in the same subnet as the client, the traffic will be routed through multiple routers on the way. The total roundtrip time between client and server corresponds to the sum of the roundtrip times between all routers on their way and the speed of the routers.

The following example illustrates a typical scenario: client A wants to exchange data with server S.

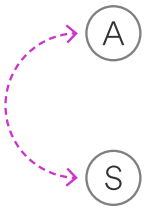

Figure 3. Client exchanges data with a server.

The direct route typically includes many routers at the ISPs of the client and server, plus routers at exchange points between those internet service providers (ISPs). In our example, it looks like this:

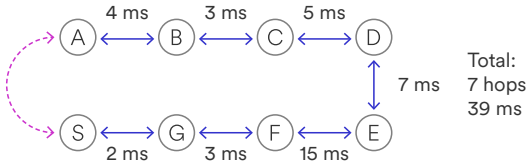

Figure 4. Direct Route

The intermediate hosts (routers) B to G are used to route traffic between A and S, such that the total time for traffic between A and S is 39 milliseconds.

When adding a classic proxy, the chain above usually becomes longer. More hops and a longer roundtrip time is the result. By adding the time that the filtering proxy servers require for security filtering, latency can increase and result in recognizable delays for the user.

When handling the traffic via a Peering POP, routes between servers and routers can be optimized by establishing good peering partnerships with major providers of internet architecture and internet services and then constantly optimizing the routes to those from and to our proxies. Continuing with our example, the result can look like this:

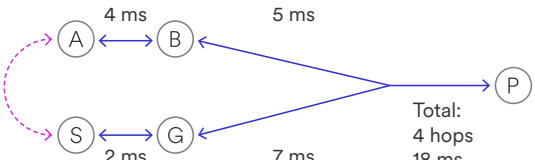

Figure 5. Handling network traffic via a Peering POP.

The Peering POP proxy P is introduced, and the physical location of P might be geographically farther away from the routers of the direct connection. However, there are two advantages. First, due to peering arrangements with the service provider (S) and with the ISP of the customer (B), the number of hops can be drastically reduced. (In the example, they are down from seven hops to four hops.) Also, the speed of the routing decision in P and the switching power of the links between the remaining hops (fast switches, modern fiber cable) can reduce the time needed for the individual hops. Even though the distance is greater, traffic is moving faster. In our example, the total time is now only 18 milliseconds. The above example has been validated in multiple real deployments.

Support of HTTP/2

This is less of an advantage of a proxy connection versus a direct connection (unless the proxy connection leverages HTTP/2 and the direct connection does not) and more of a mitigation strategy to avoid a delay penalty that the proxy is paying in HTTP/1.1 scenarios. This is hard to beat even with most optimized routing advantages.

Many websites consist of many, mostly small objects, and the data associated with those objects can be transferred with one or a few TCP/IP packets. Therefore, the time it takes to send the request to the server and waiting for and receiving the response headers are a large portion of the overall transaction time—especially when a set of small files is being downloaded. Even though browsers are opening a few parallel HTTP/1.1 connections, modern websites have so many objects to download that more object requests are queued on the client side, waiting for the last object to complete before the next request is sent. Figure 6 shows three objects (blue, orange, and purple) that are queued for a single connection. The request is sent out and responded to by the server. Both client and server are waiting for the next packets to arrive, hence the whole download takes 370 milliseconds.

HTTP/1.1 introduced the concept of pipelining to mitigate the waiting effect as much as possible. However, the pipelining concept had some challenges, leading to poor results, so it is not actively used in modern browsers or servers.

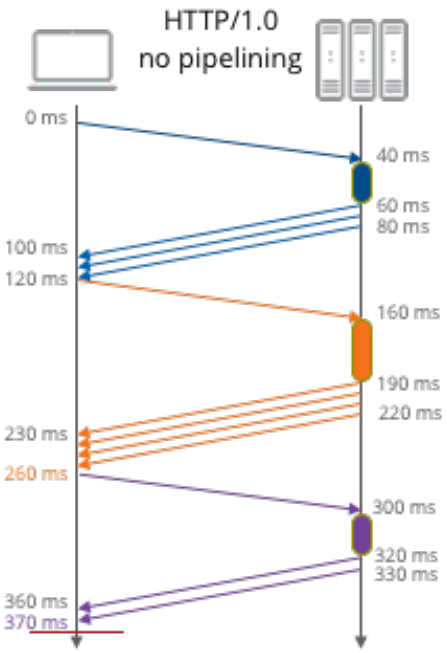

Figure 6. HTTP/1.1 pipeline.

Instead, the industry is rapidly shifting to HTTP/2 and potentially even HTTP/3. These protocols have asynchronous, multiplexed handling of data built into the protocols from the start. Figure 7 illustrates the same example from above in the HTTP/2 world. The three requests no longer have to be queued but rather can be immediately sent to the server. The server can process these requests in parallel and send any packets from any of these objects down to the client in any order on the single connection. Because there is now only a single waiting period on the client side for the first request, the overall time to deliver the three objects comes down significantly—from the 370 milliseconds down to only 180 milliseconds—so basically about half the time.

This is a huge latency benefit for any connection but particularly for proxy connections. Consider the waiting periods in the HTTP/1.1 connections again. Not only will the client and server need to wait between requests and responses and between the last and the next transaction, but the proxy will now need to do the same. These waiting periods basically double, resulting in significant additional latency for these types of web pages.

If the browser, server, and web proxy support HTTP/2, delays disappear. Due to the early support of HTTP/2, Skyhigh Security's Web Gateway and Web Gateway Cloud Service have a significant latency advantage over the solutions offered by vendors that do not support HTTP/2.

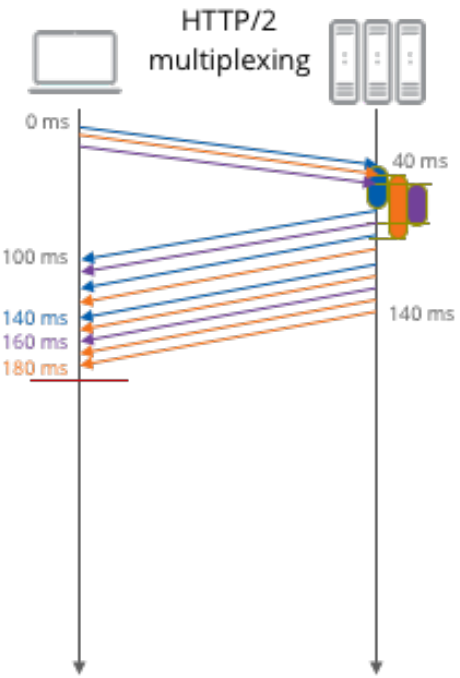

Figure 7. HTTP/2.

Bigger Pipes

Another benefit of the peering arrangements at major IXPs is that the servers are directly connected via fiber cable with the main switch, where the peering partners are also connected. The technology provided by those switches at the IXP can provide extremely high bandwidth. We are not just talking about hundreds of megabits per second or gigabits per second of connectivity, but several tens or hundreds of gigabits per second. The two largest pipes that Skyhigh Security Web Gateway Cloud Service is maintaining with a peering partner offer a bandwidth of 500 gigabits per second each. Currently, Skyhigh Security maintains more than 2,000 peering connections with more than 1,000 peering partners globally and handles more than half of its global traffic via those peering connections.

This allows for extremely high volumes of traffic to those services without saturating a classic internet upstream connection.

These big pipes make a real difference, especially when it comes to downloading larger files.

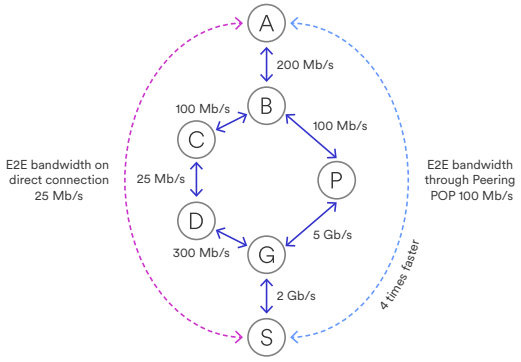

Figure 6. The benefits of a Skyhigh Security Peering POP.

The direct connection to the ISP has a known bandwidth, but the bandwidth of the hops to the final server vary, and, depending on connections to popular providers, the transaction competes with its digital neighborhood on the shared bandwidth. The end-to-end bandwidth is only as good as the slowest link between two intermediates on the route. When the routing occurs across a Skyhigh Security Peering POP, the download benefits from the larger pipes. Throughput is much higher, so the total download time, especially for larger files, can be significantly faster than the direct connections illustrated in Figure 8.

Web proxy cloud services from vendors that do not have special peering arrangements and use shared network connections are likely to suffer from competing bandwidth limitations, making it impossible to benefit from the reduction in latency for larger downloads.

Test Results

As part of its research, Skyhigh Security reconfigured a small but representative subset of its sensors to compare direct download speed and download speed via its proxy service from locations globally into different, popular networks and of different data sizes. The following download tests were performed:

The following download tests were performed:

1. Larger movie from a page at a public cloud hoster
2. Test web page (HTML+CSS+images) hosted in the

public cloud

3. Wikipedia page (with all sub-objects) via HTTP/1.1
4. Wikipedia page (with all sub-objects) via HTTP/2
5. Antivirus engine file from Akamai's CDN
6. Page at Live.com via HTTP/2
7. Microsoft Office update file
8. Google.com home page
9. Shared document at docs.google.com

The tests measured all phases of the download: establishing the connection, the time it takes for the first byte of the download, and the time it takes to complete the download. The following tables, which compiled the results of 2,614 individual test runs, only show the results for the completed downloads. Many of the tests were completed in a fraction of a second, so it was challenging to achieve negative latency. In fact, we can see that across all tests, the majority of the test runs show positive additional latency introduced by the proxy. The average download time across all tests is 13% higher than doing a direct download. (When the average download time is 600 milliseconds to complete a test, the average time through the proxy across all tests is 678 milliseconds).

However, the results also show that in 1,258 or 48% of all cases, the download through the proxy was indeed faster than the direct download, so the download through the proxy results in negative additional latency.

Faster:Slower	All Tests	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
All sensors	**1258:1356**	**220:75**	**145:151**	**36:261**	**57:187**	**200:129**	**196:89**	**151:165**	**79:179**	**174:120**
Sensor Amsterdam	**104:184**	39:11	3:30	0:30	3:21	18:14	16:15	4:23	0:25	21:15
Sensor Los Angeles	**147:149**	13:16	17:11	0:40	4:28	29:11	22:7	14:24	23:6	25:6
Sensor Miami	**136:149**	28:10	16:21	0:37	0:22	33:5	21:11	20:7	0:19	18:17
Sensor Montreal	**119:166**	28:0	41:0	0:29	0:27	0:37	28:8	1:42	5:14	16:9
Sensor Paderborn	**142:152**	25:0	8:20	2:30	2:22	21:16	26:16	24:14	3:25	31:9
Sensor Paris	**106:187**	0:26	1:32	3:29	14:13	27:14	17:11	22:22	3:30	19:10
Sensor Singapore	**232:66**	34:0	36:1	22:9	19:10	29:5	24:6	23:8	19:13	26:14
Sensor Sydney	**164:126**	32:8	8:22	5:23	11:12	26:6	29:5	20:17	18:18	15:15
Sensor Tokyo	**108:177**	21:4	15:14	4:34	4:32	17:21	13:10	23:8	8:29	3:25

Table 1. The ratio of tests with negative additional latency versus positive additional latency.

Avg.TimeFactor	Avg.Tests	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
Average sensors	**+13%**	**-20%**	**-1%**	**+91%**	**+29%**	**-25%**	**-14%**	**+22%**	**+32%**	**-0%**
Sensor Amsterdam	**+55%**	-36%	+41%	+195%	+48%	+19%	-5%	+119%	+115%	-4%
Sensor Los Angeles	**+12%**	+1%	-1%	+127%	+28%	-45%	-21%	+46%	-19%	-11%
Sensor Miami	**+12%**	-39%	+2%	+108%	+58%	-43%	-12%	-33%	+58%	+9%
Sensor Montreal	**+44%**	-17%	-42%	+162%	+72%	+77%	-20%	+135%	+20%	+6%
Sensor Paderborn	**-5%**	-65%	-10%	+41%	+32%	-30%	-9%	-12%	+18%	-13%
Sensor Paris	**+10%**	+40%	+27%	+34%	-12%	-18%	-14%	+6%	+34%	-9%
Sensor Singapore	**-32%**	-39%	-47%	-46%	-21%	-57%	-26%	-41%	-4%	-9%
Sensor Sydney	**-7%**	-20%	+15%	+15%	+3%	-79%	-16%	+18%	+1%	-1%
Sensor Tokyo	**+25%**	-4%	+3%	+182%	+49%	-51%	-5%	-38%	+63%	+28%

Table 2. The average time delta of a proxy download compared to a direct download.

The two tables show the results in two ways. The first table simply counts all test runs and shows in how many runs the proxy download finished earlier than the direct download. The green cells indicate when a significant number of tests show negative latency. The table also shows the sum for all sensors and for all test scenarios. The second table builds the average of the download time for each test and again builds the average across all sensors and all tests.

It is interesting to dig one level deeper and to compare the measured results with the theoretical findings above. When comparing Test 3 and Test 4, it becomes obvious that HTTP/2 has a positive effect and helps a proxy solution not to lose speed and incur additional latency while downloading a larger number of small embedded objects. While the average time delta comes down significantly, it is not enough to create negative latency across a majority of test sensors in this particular test scenario. In Test 6 however, when HTTP/2 is used to download a page from another popular provider, most tests show a negative additional latency. From each sensor, the average download time via the proxy is shorter than the direct download time.

In test scenarios where the downloaded object(s) are larger (Tests 1, 5, and 9), the negative latency effect is measurable. From some sensors, each and every test run is faster via proxy than via a direct connection. The average download time can be up to 65% faster than a direct download. This is primarily a result of the bigger pipe effect explained above.

Summary

Additional latency caused by a web proxy cloud service can cause a great deal of user dissatisfaction. Bringing down that latency is not trivial at all, but a combination of technologies and innovations such as HTTP/2 multiplexing, optimized routing, and bigger pipes in peering deployments can bring down the delays so much that the user is actually experiencing negative additional latency. And that occurs even when additional security filtering is applied to the download. At this time, there is no external test lab that compares those data points and can compare different web proxy cloud service vendors. Skyhigh Security Peering POP deployments provide advantages that vendors with classic deployment forms cannot offer. This can be achieved by adding a performance-boosting service to its web security filtering.

1. This excludes solutions that analyze user behavior and download internet resources proactively, keeping them cached locally before the user explicitly asks for them. The goal of this paper is to discuss how to minimize the time between when the browser sends a request and when the response data is received.
2. You can enter those addresses into your favorite IP-location service: https://whatismyipaddress.com/ip/185.125.224.6

About Skyhigh Security

When your sensitive data spans the web, cloud applications, and infrastructure, it's time to rethink your approach to security. Imagine an integrated Security Service Edge solution that controls how data is used, shared, and created, no matter the source. Skyhigh Security empowers organizations to share data in the cloud with anyone, anywhere, from any device without worry. Discover Skyhigh Security, the industry-leading, data-aware cloud security platform.

Learn More

For more information visit us at skyhighsecurity.com

Skyhigh Security

6220 America Center Drive
San Jose, CA 95002
888.847.8766
skyhighsecurity.com