Steam Pipes - An Analysis of Steam's Content Delivery Strategy

Julian Baldwin, Sengdao Inthavong and Daniel Zhao

ABSTRACT

Our research delves into Steam, the leading digital distribution platform for PC games, examining its content delivery strategy amidst the exponential growth of gaming. We analyze Steam's approach to handling massive file sizes and global demand, investigating implications for content delivery optimization. By employing simulated user interactions and network analysis tools, we dissect Steam's CDN architecture to unravel its dependency mechanisms. We find that Steam utilizes a network of cache servers to speed up data downloads, with the Steam client preferring downloading from multiple of these cache servers at once, over third party CDNs. Our findings contribute insights into network dynamics and offer implications for content delivery systems beyond gaming platforms.

1 Introduction

It is no secret that gaming has become a significant part of many people's lives. Of the many ways people download and play video games, Steam stands out as a leading platform in the industry. From its launch in 2007, it has rapidly risen to becoming the most popular digital distribution platform for PC games, having an estimated 75% market share in 2013, to a concurrent user count of over 30,000,000 by 2023.[1] Steam's primary service is allowing users to purchase and download games, which presents unique challenges due to the massive file sizes of modern games—often eclipsing 100 GB [2]. Steam handles petabytes of data daily and has a global reach, including regions with poor internet infrastructure [3].

How Steam handles such a load on demand and deliver to its users as fast as possible, amongst the potential implications of this strategy to other applications, becomes our interest for this analysis. Specifically, we wish to analyze the content delivery strategy of Steam. Additionally, we wish to investigate any potential biases in play when it comes to delivering content for paid or free packages, as well as any biases for 1st-party games made by Valve, the creator and distributor of Steam, as opposed to 3rd-party games. As such, the focus of the study is to uncover the underlying user experience during Steam content delivery by download.

2 Prior Work

There has been plentiful analysis of similar entertainment platforms in the past. For example, multiple studies have come out performing similar analysis of the content delivery strategy of Hulu[4] and Netflix[5]. While these are similar in that they perform analyses of entertainment platforms, we believe that Steam would have a notably different delivery strategy, as the previously analyzed platforms are focused on streaming platforms. For video streaming, content is delivered on demand in set order, and is not downloaded permanently onto the user's device. Steam, which primarily which primarily distributes video games and supporting software, needs to deliver larger file sizes in whole before the user can begin playing, making absolute end-to-end speed of delivery much more important for their business.

As of the writing of this report, there has been no significant formal analysis of Steam. However, there is a paper to be presented at a virtual Internet measurement conference hosted by Northwestern (Passive and Active Measurement Conference 2024) on Steam's load balancing strategy and existing infrastructure soon after this report is completed. The paper is not yet public, but its findings may add additional context towards the analysis of Steam's content delivery strategy.

3 Approach

3.1 Theory of Approach

Our approach to analyzing Steam's CDNs are follow similar paradigms to those of the previous analyses of Netflix, Hulu and other such streaming platforms. Our analysis is centered on the user's experience, and determining which servers a user client would connect to in order to download a game. To do so, we would initiate a download as a simulated user for a game or other software through the Steam client, and capture packets to see what servers and other content providers a client would connect to in order to download the requested game.

3.2 Tools Used

Amazon EC2: A popular reliable compute platform that can scale. We use the AWS cloud computing platform to setup EC2s that will host instances of our Steam client across var-

ious regions that essentially behave as vantage points for our study. Each instance was running Ubuntu Linux. The Amazon Machine Image (AMI) Copy feature is utilized to easily copy the initial Amazon Machine Image created between AWS Regions. This means that our tools and configurations can be replicated and scaled for multi-region deployment, giving consistency and replicability across instances.

SteamCMD: SteamCMD is a Valve developed command-line tool that acts as a stripped down version of the regular Steam client. It's primary purpose is to act as a way for servers to easily download and update dedicated servers on Steam. Additionally, it allows users to log in anonymously, allowing them to download server files or other free applications without the need for them to authenticate themselves.

TShark: Wireshark is a widely used too for packet capture, and is the primary data collection method of our work. In our project, we used a command line interface variant of Wireshark called TShark.

iptables: iptables is a commonly used command-line utility on Linux systems that allows a user to configure the IP packet filter of a machine's firewall. In this project's testing, it was used to create adversarial conditions to see how Steam handles poor connections to servers. To do so, we use it to drop incoming packets from IP addresses that the Steam client was to connect to.

3.3 Methodology

We will be applying similar principles to the work done in analysis of video streaming, though for Steam as opposed to media streaming services. To begin our analysis of Steam, we created multiple AWS EC2 instances in multiple regions across the world to act as vantage points for our data collection. To do so, we created a single AWS EC2 instance, and pre-loaded it with the tools mentioned in the previous section, being SteamCMD and TShark. Additionally, we added a short bash script that would be used to run a single download test, as well as the file system setup required for SteamCMD to function. This script is located in the link provided in the Appendix. From there, we created Amazon Machine Instances (AMI) to allow us to duplicate our EC2 instances around the world. These AMIs were deployed in the United States, Canada, Brazil, The United Kingdom, Germany, South Africa, Australia and Japan. (Fig. 1)

Our data collection was automatically performed by running a simple bash script. (See Appendix) This bash script which would run a download test. This script would begin packet capture through TShark, then use SteamCMD to obtain either a video-game or a dedicated server file. If needed, the script can be configured to automatically log in a user as well. Once the game is finished downloading, SteamCMD closes, and the packet capture ends as well.

Most of our data collection was for the package *Half-Life Dedicated Server*, which is an application for running dedicated servers for a number of Valve developed GoldSrc games, such as *Half-Life*, *Counter-Strike* or *Team-Fortress Classic*.

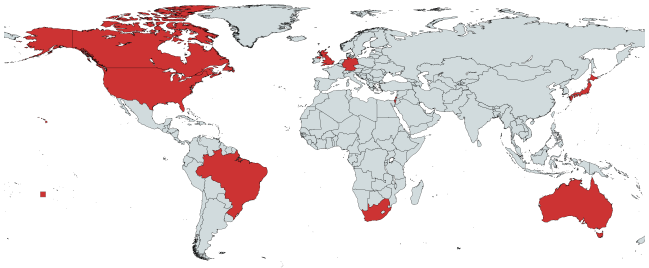

Figure 1: Highlighted countries show our deployed EC2 vantage points

This package was valuable as a test package, as it is anonymously available without the need to authenticate first, as it is a free tool designed for hosts to easily obtain and run to set up servers. With that said, this package is no longer actively developed by Valve, only receiving the occasional engine fix or security update. Therefore, we additionally downloaded several other packages through SteamCMD (see Table 1).

Package	Cost	Developer	Size
Half-Life Dedicated Server	Free	1st Party	48 MB
Unturned Dedicated Server	Free	3rd Party	2.5 GB
Ricochet	$4.99	1st Party	450 MB
Slay the Spire	$24.99	3rd Party	1 GB
Jump King	$12.99	3rd Party	1.7 GB

Table 1: Downloaded packages

We believe that these packages represent a relatively diverse spread of both 1st party and 3rd party free, as well as paid games, with *Half-Life* and *Ricochet* representing 1st party Valve developed games, and *Unturned*, *Slay the Spire*, and *Jump King* representing 3rd party games. As both *Half-Life Dedicated Server* and *Unturned Dedicated Server* are tools for setting up dedicated servers, neither package required user authentication and could be obtained anonymously. The other three packages are paid games, which require proof of purchase in the form of user authentication. In this case, we used one of the author's Steam accounts to log in and authenticate through SteamCMD in order to download these games.

When running tests, we found no significant difference between Steam's CDN strategy across regions, other than local CDNs that would be best suited for the vantage point. Therefore, for our more in-depth testing, we focused on two regions, the United States and Japan.

Once the download is completed, and Steam has installed the requested package, our bash script automatically closes SteamCMD, as well as the packet capture, saving it as a .pcap file. This allows us to download the resultant .pcap files for analysis. To analyze the files, we used the Python package Scapy to perform automatic analysis of the packet

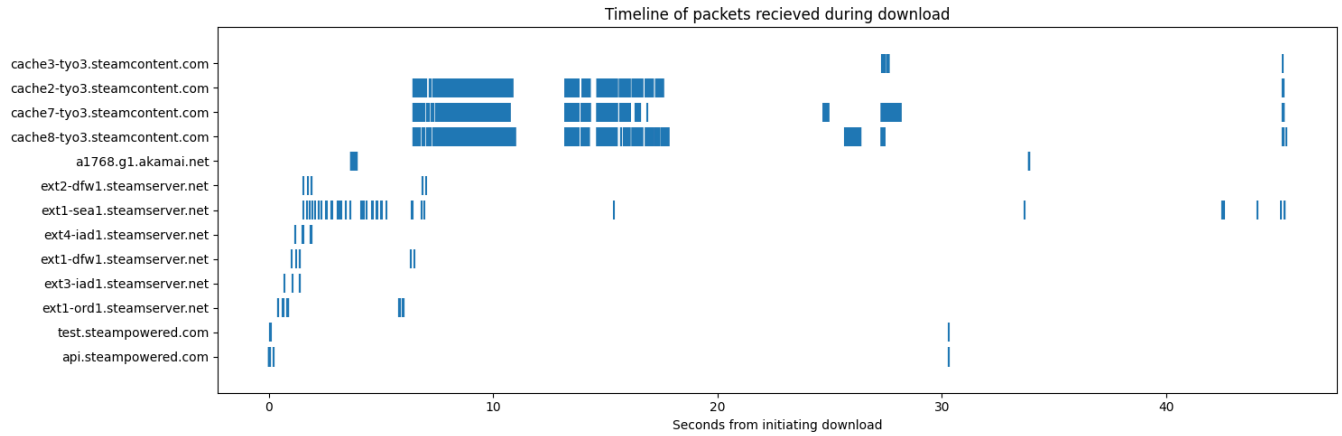

Figure 2: Timeline of each packet received after initiating download, sorted by source host. Download is from our Tokyo vantage point.

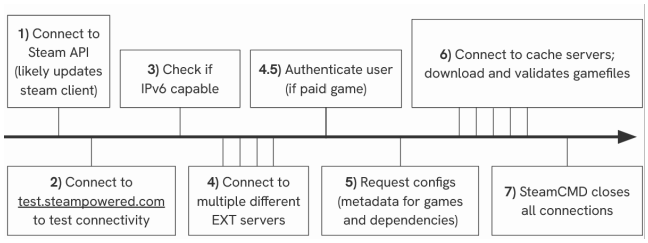

Figure 3: Typical download timeline

captures, as well performing manual analysis through a GUI version of Wireshark. The scripts written were to review the resultant .pcap file and see which DNS requests were made during the download time, and output the name of the domain, as well as the resultant IP address. Our manual analysis consisted of observing which servers Steam would connect to at which times, as well as the size of the packets being transferred. Depending on the results of the test, we would rerun the test again with varied conditions. The analysis scripts are located in the repository linked in the appendix.

To emulate restricted download conditions, we used the Linux utility iptables in order to drop incoming packets from IP addresses which the Steam client attempts to connect to. Doing so forces Steam to reroute the servers it connects to, allowing us to perform a more in depth analysis of Steam's content delivery strategy. We would repeat this firewall reconfiguration, repeatedly blocking addresses that Steam would attempt to connect to through using iptables until SteamCMD would either fail to download the requested package, or would re-route to 3rd party CDNs.

The results of each tests would be downloaded for additional analysis, as well as to observe any changes to the strategy used by Steam. These observations would inform our next tests, allowing us to determine which changes we applied to iptables for the next test. In our case, we identified two distinct types of servers which Steam seems to as-

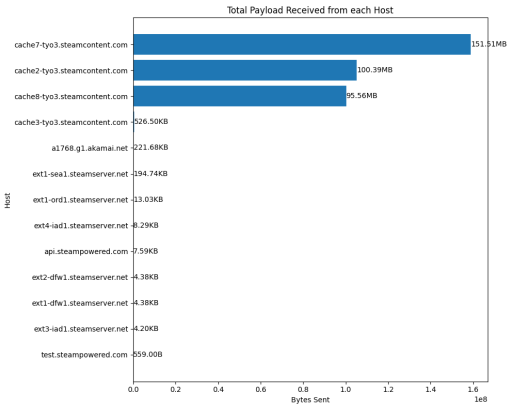

Figure 4: Total payload downloaded per host, from Tokyo vantage point. Note that the total volme from caches is significantly larger than the download size of the game (48MB).

sociate with, which we refer to as EXT servers, as well as Cache servers. In our testing, we would repeatedly download *Half-Life Dedicated Server*, though with either more cache servers being blocked, or with more EXT servers being blocked. These tests would repeat until Steam would fail to download and install the server, and we would switch either to blocking the other server type, or to a different region to repeat the tests. During this testing, we would perform similar analysis as before to review any changes to how Steam would download the server.

4 Results

4.1 Download Structure

From our analysis, we found that the Steam client makes a relatively consistent set of connections. A general timeline of connections created by Steam during a download is shown in Fig. 3, and a per-packet timeline is shown in Fig. 2 Steam

typically begins a connection with a request to an API, which we speculate to be a check to see if the Steam client initiating the download is up to date, or if it needs an update. Steam then connects to a test domain, likely to check if the connection is suitable, before following with a connection to check if the client is IPv6 capable. Steam then creates multiple connections to different ext servers within the users region. Then, if the requested package is one that requires a purchase or license, Steam will connect to the API again, and authenticate the user according to their input credentials. Once the user is authenticated, or if the user is anonymous, Steam will request multiple metadata files for the game which is being downloaded, as well as the metadata for any related dependencies. Steam will then connect to multiple cache servers, and begin downloading the game in chunks from all of them at once. Once the game is downloaded, validated and installed, Steam will close all connections at once.

These results seemed to be relatively consistent across vantage points, as well as for different packages. The only notable difference we found was between dedicated server files, and other games. Dedicated server files could be obtained anonymously, which did not require any authentication, while games proper often needed licenses, which would then require the user to log in and authenticate. Aside from this difference, we found that there was no discernible difference between 1st or 3rd party games when it comes to the distribution strategy. Any difference between the two would likely come through throttling, which we did not measure. Location also had no significant difference in strategy, although the ext and cache servers Steam connected to changed to better match the user's location.

We found that Steam typically downloads significantly more data from the cache servers than would be expected considering the file sizes of the games. For instance, when downloading *Half-Life Dedicated Server*, Steam would consistently download 100-200 MB of data from each of the cache servers - significantly more than the relatively small 48MB of size the installed game takes up on the user's disk. (Fig 4) We speculate that this is due to Steam distributing the load of the download across multiple servers, with each server sending as much bandwidth as possible in order to speed up the download. This would result in repeat packets which would be rejected by the client, though it would increase the overall bandwidth required by the download.

Additionally, we discovered what we would refer to as ext servers, who we named after their domain containing "ext" in the name. Unfortunately, the purpose of these servers is unclear, though Steam would consistently make connections to these ext servers. However, we believe it is very clear that these servers do not contain any game data due to the small size of the packets captured, neither downloading or uploading a significant amount of data back to these servers. (Fig 4, 5) Despite this, these servers seem important to the download overall, and when blocked, the download would be unable to continue. Therefore, we believe that these servers would either be collecting user statistics, or be servers who manage and organize the download.

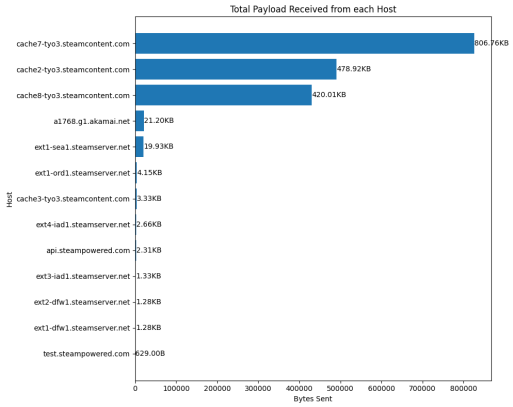

Figure 5: Total payload uploaded per host, from Tokyo vantage point. Most communication from the client is to the caches, presumably to validate the core game-files.

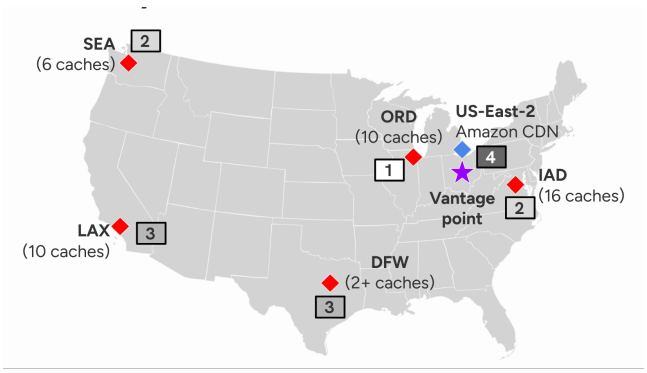

Figure 6: Sequence of cache dependency for Columbus, Ohio if previous set is blocked; for example, if [1] ORD is blocked, [2] SEA and IAD are utilized for download

4.2 Adversarial Conditions (IP blocking)

Through our IP blocking procedure, we can explore the larger network of Steam's cache servers beyond just the closest to our vantage point, and gain insight into their fault tolerance strategy. Fig 6 shows the progression of caches used as we repeatedly downloaded the same package from our vantage point in Columbus, Ohio. We find that Steam tends to prefer closest location within what we refer to as a "super-region", which is generally the country the user is located in, and will fall back to other cache servers within the super-region after the best cache times out. For our Ohio vantage point, this means Steam first falls back from Chicago to Seattle and Washington D.C., before then falling back to LA and Dallas caches when all others have been blocked. Throughout, Steam continues to connect and concurrently stream game data from 2-4 cache servers. If all super-region caches are block, Steam will then connect to non-cache CDNs hosted by mainstream, 3rd party CDN providers such as Akamai,

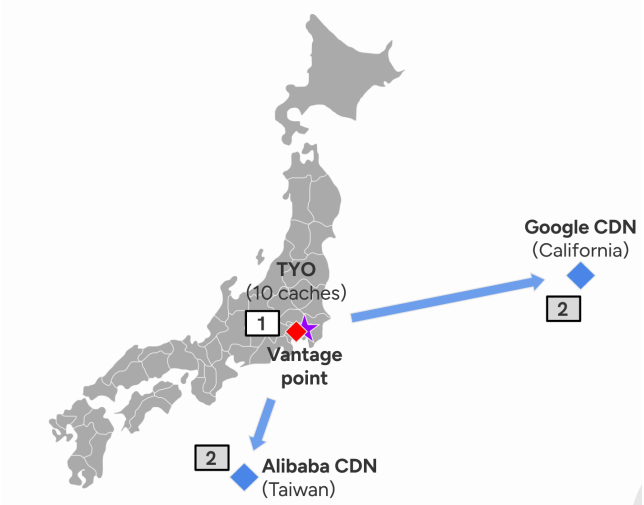

Figure 7: Sequence of cache dependency for Tokyo, Japan; if [1] TYO is blocked, [2] outer-CDNs utilized for download

Google, Alibaba and others.

Fig 7 shows this strategy is largely consistent in Japan, where our Tokyo vantage point will initially connect to various Steam caches in Tokyo. Then, once these are blocked, there are no more Steam caches within the super-region of Japan, so Steam falls back to Google CDNs in California and Alibaba CDNs in Taiwan.

Beyond Steam's caches, a crucial step of the download process is the usage of EXT servers, so we used IP blocking to investigate these as well. Fig 8 shows that they demonstrate a different fallback strategy than caches; once all EXT servers in Tokyo are blocked, the Tokyo vantage point connected to Hong Kong, then Singapore, then went as far as LA to continue to connect to Steam's 1st party EXT servers. The role of EXT servers is still somewhat of a mystery to us: they do not contain any game files, and packets do not contain significant identifying information. Yet, we can infer that it is crucial for Steam's download to function since the connection with these EXT servers opens early in the download and remains open until completion, and Steam is willing to search much further to connect to EXT servers rather than rely on 3rd party CDNs like with the caches.

5 Limitations and Future Work

Due to limitations in time and resources, our study only conducted in-depth investigations on two vantage points: Columbus, Ohio and Tokyo, Japan. Our results seem to corroborate well, but this is too limited a sample size to draw broad conclusions about Steam's global content delivery strategy. We conducted brief experiments with 7 other vantage points (Fig. 1) which did not show significant differences in CDN strategy, but future work could confirm these observations with in-depth data.

For our experiments, we selected games and packages that

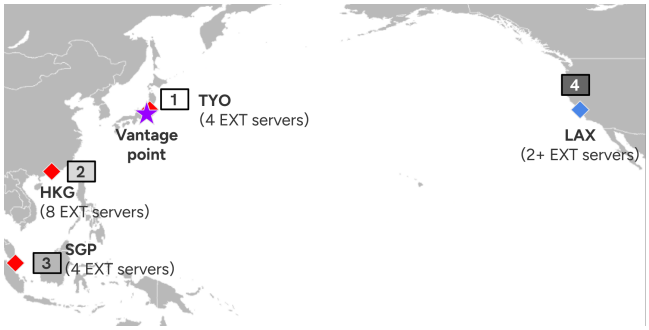

Figure 8: Sequence of EXT server dependency for Tokyo, Japan; e.g. if [1] TYO is blocked, [2] HKG is utilized

are globally popular and supported in many languages to be broadly representative. However, Steam may show different CDN behavior when downloading regionally popular or unpopular games from different vantage points, so a more complete, wide-ranging list of test packages to download could help fill this limitation. Additionally, we were restricted in potential games to download by the maximum storage of free tier EC2 instances, which is around 8 GB, not accounting for storage taken up by the tools required, as well as Linux itself (downloading small games also saved us time and resources). Steam may use a different strategy to serve very large games, so this is worth specifically investigating.

Amazon's EC2 instances are only given an IPv4 public address, so our study is limited to Steam's behavior in IPv4. Steam clearly attempts to check if the connection is IPv6 capable, and so accounts for the potential of an IPv6 connection. Therefore, future studies could use IPv6 capable vantage points to analyze if Steam's content delivery strategy changes when using IPv6 over IPv4.

Additionally, while we believe that there is no significant difference between the server strategy between free and paid/1st and 3rd party games, we cannot make any firm judgement that there is no bias present. We do not perform any analysis of the characteristics of download speed when comparing free vs. paid games and 1st- vs. 3rd-party games, and so there may still be a difference in prioritization of the downloads by Steam's servers.

6 Conclusion

In this report, we study Steam's content delivery strategy through downloading games and dedicated server files across a variety of vantage points. We found that Steam distributes games in a somewhat unorthodox manner, using multiple cache servers to both load distribute, as well as to download game files to a client rapidly. Under adversarial conditions, or when cache servers are unavailable, Steam will attempt to reroute the cache servers, connecting to alternative caches within the client's greater region. If no cache servers are available, Steam will reroute to 3rd party CDNs which are not hosted by Steam itself.

Additionally to this report, we discovered additional con-

nections created by the steam client, which we refer to as EXT servers. While the exact purpose of these EXT servers are unclear, downloading a game on Steam depends on these servers, Steam will always attempt to connect to one, even if they are outside of the user's greater region. These EXT servers are all hosted by Steam, and if none are available, the download will fail, with no connections being made to 3rd party EXT servers.

Ethical concerns

Our research uses the Valve provided SteamCMD client, and only makes connections through official means. While we do make a large number of download requests, we believe that our download amount is very minor when compared to the typical amount of traffic experienced by Valve servers. Therefore, we believe that our work raises few, if any, ethical concerns.

7 Appendix

Our code is available at: `https://github.com/icemoon97/steam-pipes`.

8 References

[1] Steamworks. Steam year in review 2022. 2023.

[2] Nick Evanson. Bigger than godzilla: Why are games using so many gigabytes? `https://www.techspot.com/article/2680-game-install-sizes/`, 2023. Accessed: 3/14/24.

[3] Valve Corporation. Steam download statistics. `https://store.steampowered.com/stats/content`, 2024. Accessed: 3/14/24.

[4] Vijay K. Adhikari, Yang Guo, Fang Hao, Volker Hilt, Zhi-Li Zhang, Matteo Varvello, and Moritz Steiner. Measurement study of netflix, hulu, and a tale of three cdns. *IEEE/ACM Transactions on Networking*, 23(6):1984–1997, 2015.

[5] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *2012 Proceedings IEEE INFOCOM*, pages 1620–1628, 2012.