A neighborly internet: investigating the possibilities of P2P CDNs

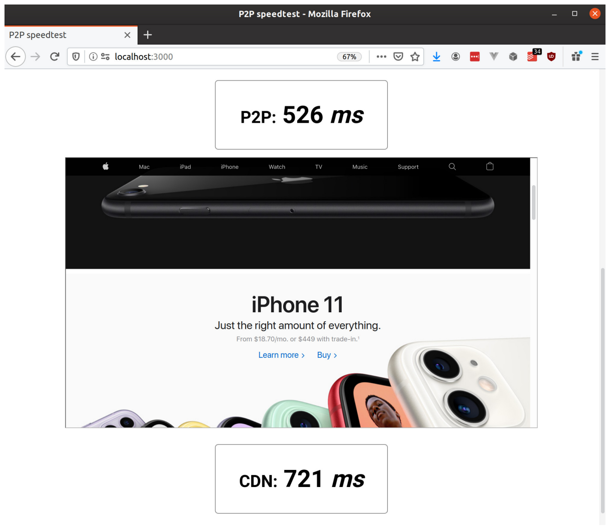
Nate Foss, Michael Silver
MIT
May 4, 2020

Abstract

With a proof of concept peer-to-peer content delivery network built on the Inter-Planetary File System, we demonstrate that P2P may be the foundation for a CDN that is architecturally superior, in terms of speed, to existing, heavily optimized commercial CDNs. We measure raw download speeds in several configurations to determine when P2P is faster, as well as build and measure techniques for organic peer discovery and swarm formation key to the success of the system. Our prototype P2P CDN outperforms Akamai by a factor of 1.28 to 2.06 on small files, depending on network configuration. On large files we find Akamai to be superior.

1 Introduction

Content Delivery Networks (CDNs) are a critical part of today's internet infrastructure. CDNs improve the speed, reliability, and cost of serving modern webpages. While CDNs have been the primary solution for the distribution of webpage assets, peer-to-peer (P2P) has gained wide adoption in the realm of filesharing (e.g. BitTorrent [2]). Recently, the Inter-Planetary File System (IPFS) has gained traction as a library for P2P applications, making a wide range of possibilities much easier to implement [8]. In this paper, we built and evaluated a P2P CDN prototype using IPFS. Our contributions are as follows: we built an in-browser speedtest utility[1] that compared the speed of loading `Apple.com` using our P2P CDN vs. Akamai [1] (Apple's CDN provider), we developed a new session-based peer discovery protocol using IPFS primitives to allow clients to always be connected with nearby hosts, and lastly we ran an experiment[2] across households in Cambridge, MA to measure the speed of our P2P CDN vs. Akamai on simple file retrieval tasks. When retrieving small files, we found our P2P CDN to outperform Akamai by a factor of 2.06 when using MIT-based hosts and 1.28 when using Cambridge residential hosts. On Verizon 4G, we outperform Akamai by a factor of 1.63 using residential hosts when retrieving small files. For large files we found the P2P CDN vastly suffers in performance.



P2P: 526 *ms*

CDN: 721 *ms*

Figure 1: A speedtest showing our P2P CDN outperforming Akamai when fetching the Apple homepage. The test was conducted in Cambridge, MA from an ethernet connected machine on Comcast Cable (gigabit down, megabit up) peering with at least one of two ethernet connected machines on the MIT network (bidirectional gigabit).

CDNs work by distributing assets—images, videos, code, or even entire webpages—via a fleet of "edge" servers located around the world. Because CDN edge servers are globally distributed, positioned close to end users, and equipped with excellent internet connections, they are able to provide users of a website a great experience, no matter where the user is located. Specifically, this

[1]`https://github.com/p2p-cdn/speedtest-site`
[2]`https://github.com/p2p-cdn/experiments`

translates to a fast webpage load for static content (such as webpage frontends) and a fast download/streaming experience for larger assets such as movies and apps downloads. For the company, a fast user experience translates to more revenue. Besides speed, companies use CDNs for reliability and security (such as DDoS protection), as well as to save on networking costs. We believe speed (without compromising on other factors) is the most impactful area of work, as it directly translates to a better user experience and more company revenue.

How can we make CDNs faster? The speed of a content delivery is driven by two main factors: latency and bandwidth. Latency is the time it takes for a packet to travel between two machines. Bandwidth is the rate at which data can be transferred between two machines. Competitive improvements on these two fronts are nontrivial given commercial CDNs. CDN edge servers are typically located at internet exchange points (IXPs) close to end users and equipped with the highest bandwidth connections current hardware allows [9].

To solve this problem we turn to P2P technologies. In a P2P CDN, end users request content directly from other peers in the network. On one hand, it would seem that getting a piece of content directly from a neighbor geographically closer would be very fast, and that there are many more neighbors nearby than CDN edge servers. On the other hand, commercial CDNs are made of extremely powerful machines on hundreds of Gb/s networks. What's more, since CDNs usually peer directly with ISPs, the bottleneck is often between a user's home and their ISP. We test these assumptions in Cambridge, MA and find our P2P CDN to outperform Akamai on simple file retreival tasks.

In the following sections, we describe our P2P CDN prototype (Section 2), a novel peer discovery protocol we designed and built (Section 3), and evaluate the performance of our P2P CDN (Section 4). We wrap up by discussing related work (Section 5) and conclude (Section 6).

2 P2P CDN Prototype

The goal of this prototype was to test the speed (and thus viability) of a functioning CDN composed of regular internet users re-hosting static assets they had previously downloaded. To this end, we made a website where users can visually and quantitatively test the difference in load time for some example webpages, comparing how long it takes images to show up if downloaded from Akamai's CDN versus our P2P network.

We considered several pre-built peer to peer networking stacks, and ultimately settled on the Inter-Planetary File System (IPFS) and their underlying library, libp2p [8, 6]. Section 5 goes into more detail about the trade-offs of different options. IPFS presents a straightforward API for retrieving data from the network, internally handling the complexity of finding and transferring data. IPFS uses gossip among peers, a DHT to locate data, and TCP sockets to transfer data. As a bonus, since content is requested by its hash, all incoming data—which is from potentially untrusted peers—can be quickly validated and confirmed to be what was requested, solving one of the immediate security risks of open systems anyone can join.

Using IPFS as our foundation, we then set up two example webpages:

`apple-cdn.html` which is a snapshot of the Apple homepage containing many large static images, and made sure all images were being retrieved from their default location, the Akamai CDN.

`apple-ipfs.html` which is the exact same page, but with all of the source links to large images replaced with IPFS-interpretable links for the same data.

It is important to note here that we manually added these images to the IPFS network ahead of time and precomputed their hashes. This is akin to convincing Apple to switch its CDN and updating its static asset links, which happens all the time. It is not a way of converting arbitrary websites on the fly, though we do consider this for later work.

When a webpage requests an image via an IPFS-interpretable link—essentially, one which contains the file hash in a certain recognizable format—the IPFS Companion browser extension [4] intercepts the outgoing request and routes it through a locally running IPFS node instead. The IPFS node then requests the data from all of its current peers, or if they don't have it, queries the DHT to find someone who does. Upon finding a peer with the requested data, it downloads and routes it back through the extension, which hands it to the webpage where it's rendered.

In order to test the difference in user experience (namely, speed) between the two, we embed both copies of the Apple site in `iframes`, sequentially load them, and use built-in JavaScript profiling tools to measure the time between starting the request for the initial page to the completion of loading the final image.

With the peer discovery mechanism in Section 3 ensuring one of the content hosts is always a direct peer of the requesting client, we find that residential users in Cambridge see approximately 1.3x faster pageload times for the IPFS-enabled version of the page.

3 Peer Discovery

One key to making a P2P network competitive with a commercial CDN is ensuring that the data of interest can be located quickly, not merely fast to download, since both affect the user experience. It takes much too long—on the order of seconds, in our tests—to start looking for relevant peers once the page has already been requested. We leveraged existing functionality of IPFS to ensure every node on our CDN is frequently discovering and connecting directly to others in the background before the page is even requested, so that it's well positioned in the network when the request happens.

Distributed Hash Tables (DHTs) are a large shared data structure across an entire network which serves simply to map keys to values via a hash table, with the added constraints that the data is much too large for any node, so some node on the the network must store each shard of the overall data and must be findable by any other. IPFS uses a Distributed Sloppy Hash Table which combines ideas from S/Kademlia [7] and Coral [11] (both based on the original Kademlia design [12]) to map content hashes to peer IDs of nodes potentially storing them. This DSHT is one of the more complex components of IPFS, but the properties relevant to us are that it allows any node to query a piece of content and be connected to some other node which has it, without either one knowing of the other beforehand. With a cleverly chosen file to request, two nodes can find each other with the following trick. First, one node generates a file with some known content. The second node then calculates the hash of said content and requests the content from the network. If they aren't already peers, the requesting node will be forced to query the DHT to find this unique file and the only host peer it will find is exactly the one it was trying to connect to.

Concretely, the file we choose has the contents `"Member of P2P CDN Session: N"`, where `N` is the current Unix time rounded to the nearest minute. Crucially, every minute there will be a unique file clients can request, but it's generated predictably so all nodes know what to use without coordinating.

What this means for our P2P CDN is that all of the nodes can alternate hosting and requesting these carefully chosen files and will constantly be discovering and staying connected to other members of the CDN, so that when the user loads a page and they request some image they will already be connected to the right peers. Notably, this doesn't require any modifications to IPFS itself. We are merely taking advantage of their existing network dynamics to create a densly connected subgraph within the larger swarm.

This peer discovery pre-pageload turns out to be key for performance, bringing load time down from multiple seconds to consistently competitive in the hudreds of milliseconds range.

4 IPFS CDN Performance

We evaluated our P2P CDN with timed head-to-head tests against Akamai from the everyday computers of volunteers spread across Cambridge, MA. Our tests covered a number of different possible P2P network topologies, varying home internet connections, and geographic locations, and represent a real world baseline for our P2P CDN.

4.1 Experiment Setup

We facilitated the experiment with a script distributed to participants that was responsible for automatically downloading, configuring, and running `go-ipfs` on their machine. Our test involved two types of participants—hosts and clients–as well as two assets taken from the Apple website—iPhone (22KB jpg [5]) and iPad (134.4MB mp4 [3]). Hosts were always-on IPFS nodes that pinned the iPhone and iPad assets, downloading those assets to the machine, ready to serve to any client. Clients ran a test routine which compared the speed of `curling` the two assets directly from Akamai vs. fetching them from the hosts.

In detail, the client test routine works as follows. First, the client `curls` the iPhone asset repeatedly, using `curl`'s built in ability to return `time_total` (as well as other statistics), recording the timing for each attempt. Then it does the same for the iPad video. Next it peers with the hosts, and repeatedly `ipfs gets` the iPhone image, ensuring that the hosts are connected, and that the file is garbage collected between runs to ensure it isn't cached locally prior to the start of the `get` request. `time` is used as a crude timing measurement in the absence of any internal timers in `go-ipfs`. Then the client does the same for the iPad video. By using `time` for the P2P CDN tests and `curl`'s internal timing feature, we only possibly disadvantage the P2P test's timing, while still measuring as accurately as possible when `curling` from Akamai.

We ran our tests with three different configurations:

Test 1 consisted of two bidirectional gigabit ethernet hosts at MIT (one in Simmons, one in East Campus), and one residential Comcast Cable host on WiFi with a 25 Mbps down, 6 Mbps up connection. All hosts ran `go-ipfs_v0.4.23`.

Test 2 added a host at CSAIL (bidirectional gigabit) and a wired host on Comcast Cable (gigabit down, megabit up). All clients and both MIT dorm hosts were updated to `go-ipfs_v0.5.0`, which promised speed improvements to bitswap.

Test 3 completely removed all MIT and CSAIL hosts, to test if a purely residential P2P CDN in Cambridge, MA is competitive with Akamai.

In all the experiment includes around 680 data points across 18 different tests from 14 different clients. Each iPhone measurement is the mean of 10 trials and each iPad measurement is the mean of 3 trials (to save on bandwidth and respect volunteers' time).

4.2 Results

In this section we cover the results comparing the performance of our P2P CDN to Akamai when retrieving a single iPhone image and a single iPad video, originating from the Apple website. Across all tests, we distinguish two topologies: when all hosts are included (denoted by "MIT" because this includes MIT hosts), and when only residential hosts are included (denoted "Res"). We differentiate between these topologies because having hosts serving content from the MIT network provides an advantage over pure residential P2P connections: MIT's network is a well peered autonomous system with a low AS number and high bandwidth.

Client	Akamai	P2P (MIT)	Ratio (MIT)	P2P (Res)	Ratio (Res)
1	209.50	65.8	3.18	145.5	1.44
2	146.92	67.6	2.17	143.9	1.02
3	125.98	114.2	1.10	183.2	0.69
4 (H)	73.34	85.4	0.86	131.4	0.56
	Mean:		**1.60**		**0.87**
5 (S)	145.36	264.9	0.55	261.6	0.56
6 (M)	75.92	27.4	2.77	121.2	0.63

Table 1: Test 1—P2P CDN performance vs Akamai when retrieving an iPhone image. Times in milliseconds. Ratio shows Akamai time divided by P2P time. Mean is geometric mean. Notable tests: Client 4 is at Harvard, Client 5 is in San Diego, and Client 6 is at MIT. We excluded 5 and 6 from the mean calculation.

Table 1 shows the results for the iPhone part of Test 1. Three different clients (1–3) participated from different homes in Cambridge, MA. Client 4 participated from the Harvard network. Across these clients, the P2P CDN outperformed Akamai by a factor of 1.6 when using MIT hosts and underperformed when just using residential hosts (0.87 times as fast as Akamai) on the iPhone retreival task. The speed of the P2P CDN appeared rather constant across the different clients. On the other hand, the speed of Akamai appeared to vary much greater and appears to have largely influenced the factor of speedup the P2P CDN provides to clients.

A few clients were of particular interest. Client 2 was on a residential connection located across the street from one of the MIT hosts, but didn't appear substantially faster than any of the other clients (instead, the slowness of Akamai made the largest difference). On the other hand, client 4 which was at Harvard was significantly slower using the MIT hosts, but the fastest when using residential hosts. Nonetheless, Harvard's network appears to be well peered, and Akamai outperforms the P2P CDN. Client 5 was based in San Diego, CA and was used as a point for comparison. Naturally, while Akamai performed just as good in California as it does in Massachusetts, our P2P CDN (based in MA) was half as fast as Akamai due to the added latency of cross-country data transfer. Finally, Client 6 was inside of the MIT network, and as would be expected, performed significantly faster than all the other clients with the MIT hosts. Interestingly, this client was slower than Akamai when using residential hosts.

Client	Akamai	P2P (MIT)	Ratio (MIT)	P2P (Res)	Ratio (Res)
7	229.34	96.4	2.38	138.9	1.65
8 (B)	251.20	83.2	3.02	97.9	2.57
	Mean:		**2.68**		**2.06**

Table 2: Test 2—P2P CDN performance vs Akamai when retrieving an iPhone image. Times in milliseconds. Ratio shows Akamai time divided by P2P time. Mean is geometric mean. Notable tests: Client 8 is in Brookline.

Client	Akamai	P2P (Res)	Ratio (Res)
9	131.40	26.9	4.88
10	92.40	82.8	1.12
2	200.05	204.2	0.98
7	126.53	152.1	0.83
	Mean:		**1.45**
6 (M)	79.59	56.6	1.41

Table 3: Test 3—P2P CDN performance vs Akamai when retrieving an iPhone image. Times in milliseconds. Ratio shows Akamai time divided by P2P time. Mean is geometric mean. Notable tests: Client 6 is at MIT. We excluded 6 from the mean calculation.

Table 2 shows the results for the iPhone part of Test 2. In this brief experiment we provisioned more hosts (one at MIT CSAIL and one Cambridge residential), as well as updated to `go-ipfs_v0.5.0` where possible. Across these clients, the P2P CDN outperformed Akamai by a factor of 2.68 when using MIT hosts and by a factor of 2.06 when just using residential hosts. Notably, client 8 was located in Brookline, MA, yet still benefited substantially from the Cambridge-based hosts. We suspect this has to do with peering and topology of ISPs' networks in Brookline and Cambridge.

Table 3 shows the results for the iPhone part of Test 3. In this test notably all MIT-based hosts were completely shut down to ensure the entire P2P CDN was running solely using residential connections. Despite this added challenge, the P2P CDN outperformed Akamai by a factor of 1.45 across four clients tested in Cambridge.

Location	Akamai	P2P (Res)	Ratio (Res)
EC Server room	96.98	189.0	0.51
Baker Ethernet	67.45	58.7	1.15
Baker WiFi	88.84	67.8	1.31
Baker Verizon 4G	208.71	128.2	1.63

Table 4: Test 3—P2P CDN performance vs Akamai when retrieving an iPhone image. Times in milliseconds. Ratio shows Akamai time divided by P2P time. Mean is geometric mean.

Table 4 shows Test 3 done by the same user from different places: a wired gigabit server, MIT Ethernet, MIT WiFi, and Verizon 4G—the latter three all done from Baker House, a dorm at MIT. First, we observe that a wired connection on different parts of campus, all within the same MIT network, results in vastly different results. We also observe that wireless and wired connections in the same place results in little difference. Most surprisingly, we observed that on a smartphone (Verizon 4G data), our P2P CDN vastly outperformed Akamai.

	Client	Akamai	P2P (MIT)	P2P (Res)
2	7	18.87	190.45	41.33
2	8 (B)	76.58	56.20	93.60
3	7	8.97		182.14
3	6 (M)	17.69		35.46
3	9	43.03		3.05
3	10	4.57		50.76
3	EC Server room	2.36		166.18
3	Baker WiFi	15.76		140.00
3	Baker Ethernet	6.70		45.60

Table 5: P2P CDN performance vs Akamai when retrieving an iPad video. Times in seconds. Leftmost column shows test number.

Table 5 shows the results across all tests using the iPad asset, a video taken from the Apple website. We observed across the board that the P2P CDN performed very poorly as compared to Akamai. We initially suspected that IPFS's bitswap protocol was requesting redundant blocks from different hosts simultaneously, possibly saturating bandwidth. Upon further investigation we found almost no redundant reqests for blocks. Research is still ongoing as to why the P2P CDN performs so poorly on large files.

5 Related Work

Peer to peer file sharing has a long and exciting history, and has proven to be a useful technology still widely in use today. The most well known system is BitTorrent [2] with millions of daily active users. BitTorrent is quite similar to IPFS [8] (and heavily inspired its design) in that it focuses on basic block exchange between users of the network and uses DHTs to find those blocks. We considered building this P2P CDN on BitTorrent, but found BitTorrent significantly lacking in developer tooling. We ultimately decided on IPFS because it was designed to be a single massive platform rather than BitTorrent's often app-specific swarms.

Other notable systems include Gnutella [10], which tackled the challenging problem of search and had some interesting optimizations around distinguishing some nodes as particularly useful and powerful and relying more heavily on those. The most similar to this project, however, is Coral [11], which it itself a drop-in P2P CDN using a novel DSHT to direct users' DNS queries to their nearest peers. One reason we chose not to use DNS like Coral is that we assume finding peers on the fly who have the relevant content is inherently too slow to compete with a single request straight to a commercial CDN. However, Coral's design does give it a large advantage where ours fails, which is that it can be used by sites with no action on the part of the user, or by users with no aciton on the part of the site. This is probably one reason it eventually did gain some popularity, because the path to adoption is very easy. While sites can 'upgrade' to our CDN without their users doing anything, our system does not yet include a way for users to 'upgrade' all of their sites, though we have some ideas for future work in this direction.

Ultimately, the world has found use for many different P2P filesharing systems and these few are examples of some that gained real traction, demonstrating that in many cases P2P can be better than centralized alternatives. However, none of the current systems have competed on speed: BitTorrent is about availability, Gnutella search, and Coral reliability. We believe our focus on faster page loads adds to this list another potential advantage P2P systems may have over the status quo: speed.

6 Conclusion

While not a complete product yet, our system and its tests demonstrate that a P2P CDN may be competitive on speed, contrary to conventional wisdom about network optimization and bottlenecks. We have shown in situ that webpage resources loaded from unknown peers can download significantly faster than the same resources coming from a world-class CDN. As a result, we believe our system and its speed advantage is the basis for a commercially viable CDN. Of course, there remain many areas for improvement: Coral demonstrates that user-lead adoption is not only possible, but also popular. The best path to adoption for our current system is to replace existing CDNs and encourage users to participate for speedups, compensation, or both.

Acknowledgements

The authors would like to express our deepest graduate to Prof. Robert Morris for his time providing very thorough feedback on this project for 6.824, to Mark Mayo and Patrick Lu for pushing and supporting us throughout the development of this work as part of the Mozilla Spring Lab[3], and to our classmates and friends who generously participated in our field testing.

References

[1] Akamai. `https://www.akamai.com/`.

[2] Bittorrent. `https://www.bittorrent.com/`.

[3] ipad video. `https://www.apple.com/105/media/us/ipad-pro/2020/7be9ce7b-fa4e-4f54-968b-4a7687066ed8/films/feature/ipad-pro-feature-tpl-cc-us-2020_1280x720h.mp4`.

[4] Ipfs companion. `https://github.com/ipfs-shipyard/ipfs-companion`.

[5] iphone image. `https://www.apple.com/v/home/f/images/heroes/iphone-se/hero__dvsxv8smkkgi_large.jpg`.

[6] libp2p. `https://libp2p.io/`.

[7] BAUMGART, I., AND MIES, S. S/kademlia: A practicable approach towards secure key-based routing. In *2007 International Conference on Parallel and Distributed Systems* (2007), IEEE, pp. 1–8.

[8] BENET, J. Ipfs - content addressed, versioned, p2p file system, 2014.

[9] BÖTTGER, T., CUADRADO, F., TYSON, G., CASTRO, I., AND UHLIG, S. Open connect everywhere: A glimpse at the internet ecosystem through the lens of the netflix cdn. *SIGCOMM Comput. Commun. Rev. 48*, 1 (Apr. 2018), 28–34.

[10] CHAWATHE, Y., RATNASAMY, S., BRESLAU, L., LANHAM, N., AND SHENKER, S. Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2003), SIGCOMM '03, Association for Computing Machinery, p. 407–418.

[11] FREEDMAN, M. J., FREUDENTHAL, E., AND MAZIERES, D. Democratizing content publication with coral. In *NSDI* (2004), vol. 4, pp. 18–18.

[12] MAYMOUNKOV, P., AND MAZIERES, D. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems* (2002), Springer, pp. 53–65.

[3] `https://mozilla.org/builders`