Website file download acceleration using WebRTC

João Domingos
Instituto Superior Técnico, Universidade de Lisboa
Computer Engineering
Email: joao.domingos@tecnico.ulisboa.pt

***Abstract*—The Internet, at its birth, was always thought of as a platform for communication, sharing and collaboration. Everything has evolved around those concepts.**

As size and quantity of the data continues to increase, owners of website with popular content have to respond to the high demand of their users. For this they expand their infrastructure, reserve cloud computing resources and resort to Content delivery network (CDN) services. This is costly and sometimes it is too much, during low traffic times, and it is not enough when in high demand and popularity.

In response to this challenge, this document presents a distributed, collaborative content distribution solution using Web Real Time Communication (WebRTC), allowing the creation of scalable websites, capable of hosting popular content without the need to invest in extensive hosting resources.

This allows for self scalability of a system and better file transfer performance with the selection of the best and closest peers to share the content. Infrastructural costs for the owner of the website are lower as fewer resources are required, as the clients share the work, easing the load on the server.

A prototype was built that recruits the users of a website and makes them share the content they already gathered from the server and distribute it through the other simultaneous visitors through Peer-to-peer (P2P) connections. Using geographically distributed browser simulators, test cases were executed to assess the capacity of the prototype and the potential of the solution bringing positive results were some clients experienced a reduction of almost 80% in the download times.

Keywords: web, WebRTC, Peer to Peer, Content Distribution Network, performance, resources, distributed

I. Introduction

The Internet, at its birth, was always thought of as a platform for communication, sharing and collaboration. Everything has evolved around those concepts. When Tim Berners-Lee presented [1] the proposal for creating a "small" Web at CERN, it was the beginning of something that has changed human life forever. The World Wide Web (WWW) has evolved so much since the start, and is still evolving, that nowadays it is not possible to imagine a world without the Web. The size and quantity of the data in the Internet will increase [2], keeping up with human knowledge growth, improved bandwidth and network capabilities.

There are several methods to serve content online but the simplest and most used one is through a web server. Logically structured text documents are made available that include other documents, like images, videos and scripts, to enrich the text that they provide. The creator and distributor of the content has to adapt its response capacity according to the popularity of what it is serving. This demands some heavy or elastic infrastructure to respond to user requests and it is even more so when serving large content. Also, the infrastructure needs some capacity margin to "survive" unpredictable events like network anomalies [3] or an eventual slashdot effect. The client-server architecture is the architecture used in the WWW. It is simple, cheap and convenient solution for small projects, companies and individuals that have a limited number of users, like a local pet clinic or a local restaurant. The problems arise when it has to scale to serve a larger audience [12]. It becomes more expensive, harder to maintain, to make it reliable and constantly available, even with a large affluence, as it is required for almost every business.

P2P networks are very resilient to, for instance, *flash crowds* [28]. When a large number of users, all of a sudden, want some file, they request it to the seeders of the file. Initially some will get parts of the file and become themselves servers of the content. So the number of peers that request the content is the potential number of servers of that same content. This is a self scalable solution [26], [29] and easy to maintain as each peer is responsible of their own processing, storage and bandwidth. It is a good way to share files but, at least for solutions like Gnutella [18], Napster [19], eMule [20] or BitTorrent [16], it is not appropriate for the WWW as it doesn't have the logic that browsers provide and neither was it designed for that.

Delivering data to thousands of globally distributed users, millions in the case of popular websites, is a tricky task and surely no job for a single web server. To solve that problem, a lot of websites adopted the use of CDNs. A CDN is an efficient and less costly way of distributing content to users. CDNs have replica servers that cache the served content from the origin and serve it themselves [4]. There replicas are scattered over the Internet and respond to requests that originate in locations closest to each of them. One of the goals is to achieve smaller round-trip time (RTT), that will reduce the latency when, for instance, loading a web page. CDNs, such as CoDeeN [37] or Akamai [34]–[36] also help ease the load on the origin server, making static files that are requested frequently available on those endpoints. Because the files are replicated through a collection of servers geographically dispersed [5], that answer to the nearby user requests with some load balancing mechanisms.

There is also some testing on hybrid solutions [6], [7], using client-server and P2P architecture in order to provide the best service to the end-user. But CDNs do not fully solve the problem, since costs still increase with user traffic and their scalability is limited to its capacity. One of the main

advantages of CDNs is that the services that they provide are specialized in delivering content and for that they offer a high capacity infrastructure that would be very difficult for a ordinary company to have. This results in higher availability and lower latency than what a regular host can provide. The downside of the CDNs is the cost. In order to increase the Quality of Service (QoS) for the users, the website owner has to pay. Even though a content provider can save some money by reducing traffic on the original host server, it will still have to pay the distribution network even if it is a lower value than a self maintained solution with similar performance.

This is where a relatively new technology comes into play. WebRTC [8] is being adopted and supported by all the major web browsers that support HTML5, and allows for real time device-to-device, mobile or not, communication without the need of plug-ins, just by using the browser itself.

What if a user, when accessing a website, securely shared the contents retrieved with a nearby user also needing those same images, videos and files?

The purpose of this project is to merge some advantages of the P2P networks with the WWW, transforming users that were previously only consumers into content providers, helping to spread the contents of the website that they've already accessed, while they are accessing it. A proposed solution (Figure 1) for this is to create a coordinator of peers that will be placed side by side or integrated with the traditional Hypertext Transfer Protocol (HTTP) server. This coordinator has the information of what are the clients connected to the website and who has which file. This technology is revolutionizing the Internet. First of all, it is free. Free for the users and free for the content providers if they decide to use it on their websites. WebRTC does not depend on a operating system or on a type of device. It needs a browser which almost every machine has nowadays. Another advantage is that there is no intermediary when sharing content. The content is exchanged only between the ones that have requested it and that eliminates unnecessary calls or requests and maximizes transfer rates. There are solutions out there like Maygh [50], PeerCDN [51] and Peer5 [55] but they are either proprietary or not in active development.

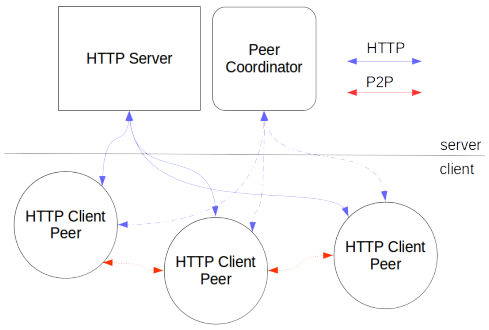

Fig. 1. Proposed solution

The clients of that website will download an additional module file that allows them to connect to the coordinator of peers and eventually to the peers that are visiting that same page. When a peer gets a file of that page, e.g. an image, it will announce to the coordinator that it is now a server of that same file, so other peers can find it and request that same file. Then, based on some logical decisions, the peers will decide if they will download the files included in the HyperText Markup Language (HTML) document from the server or if they will get them from another visitor of that page. The contributions of this work are:

- Proposal of an architecture and solution of a P2P CDN that allows peers to server static contents of a website to other peers, while browsing the web pages. This is intended to adapt to the traditional web servers and work under the hood to help reduce server costs and improve download times.
- Implementation of a prototype to demonstrate the validity of the proposed solution.
- Experimental evaluation of the prototype implemented.

II. Architecture

A. Non-functional requirements

- Self scalable
- Reasonable transfer times
- Choose the peers wisely to minimize download times
- Reduce server load
- Secure and trustworthy file transfers
- Up to date files

B. Functional requirements

- Allow for P2P resource transfer
- Only peers on the page are available for collaboration
- Client transparency
- Fall-back mechanism
- Failed connection or transfer tolerance

The architecture of the P2P CDN, displayed in figure 2, is composed of client nodes, that can be either providing or obtaining content, of a peer coordinator and a web server. These two nodes on the server side, the **Coordinator** and the HTTP server, can be together or separated, depending on the type of relationship it is intended for them to have. Detaching the coordinator from the HTTP server can allow it to provide its service to a variety of websites.

On the client side, the most important parts are the local modules Content Loader and Peer Connection application programming interface (API). The Peer Connection API module will control the connections between peers and the Content Loader module will handle everything that has to do with the Document Object Model (DOM) and decide if it needs to use the WebRTC capabilities or just request things from the origin server. This first design of the architecture was achieved taken into account the usage of the WebRTC technology. This included the interaction, Figure 3, between the various participants in the solution.

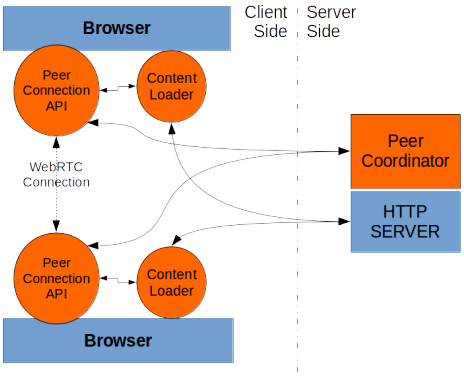

Fig. 2. Proposal for the architecture of a solution

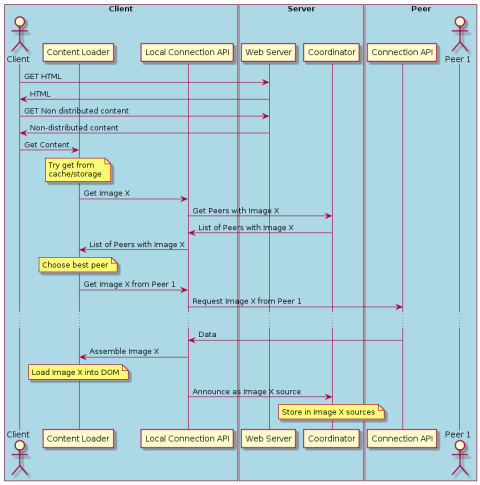

Fig. 3. Interaction between peers and server

1) Peer Coordinator: The peer coordinator must record which peers are available in the network, identify a peer when another peer asks and answer with the details needed to establish a connection between them. Without the coordinator, clients cannot know what peers are available for connection. A central entity for handshaking of connections is necessary and to avoid distributing sensitive information, like Internet Protocols (IPs), between the peers. If the list of peers would be distributed it would bring some reliability issues as a part of that information could disappear with the disconnection of peers. This was an extension of an already existent Coordinator server for WebRTC, PeerJS Server [54].

2) Connection API module: It is essential to have a solid module that communicates with the coordinator and also that wraps the creation of a WebRTC connection between two peers (example in Figure 4) in a easy to use function. Being a recent API and still in development there are different actions that need to be made, depending on the browser. This was an extension of an already existent API for WebRTC, PeerJS [54].

3) Content Loader: The content loader is the module that will implement all the logic, that determines whether a resource is already loaded, were is it going to be loaded from, opens and manages the peer connections.

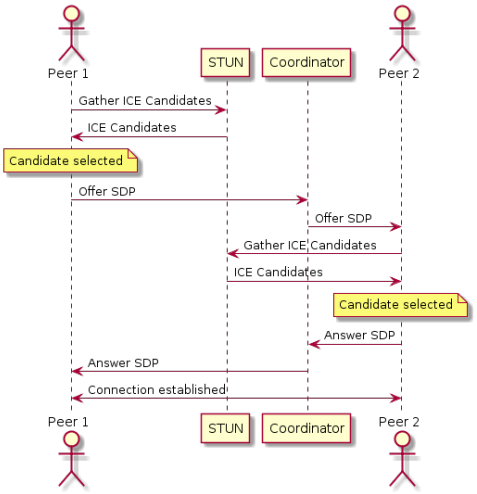

Fig. 4. WebRTC connection establishment

Assuming that the content loader and the whole P2P CDN solution would only load images from peers, the action flow when a user loads the web page will be as follows.

First the module needs to prevent the browser from fetching any of the images on the document. Only images that are already cached on the browser can load since it is a local request. For each image that did not load from cache, the peer needs to ask the coordinator what are the seeders in the network for that file. If there are no seeders in the network, besides the original server, then a regular HTTP request is made to the static file server and that image loaded into the HTML document.

III. Evaluation

For the server, an Amazon EC2 t.micro instance was used, located in Dublin. For the clients, servers from Digital Ocean were used, from different locations: New York, San Francisco, Toronto, Singapore, London and Amsterdam. For every peer that accessed the prototype and integrated the peer network, a connection limit of 10 megabits per second (Mbps) of download and 1 Mbps of upload was established. This was done to simulate a connection of a user, using their home networks.

To simulate a full-featured browser using a headless machine, software from Selenium, called WebDriver, was used. Simulation an environment with a Mozilla Firefox version 41.0.1, using a virtual display server, Xvf, and Python scripting to send commands to the virtual browser.

A. Tests Objectives

The tests preformed on the prototype have several objectives:

1) Assess if a WebRTC solution can lower the load on a the websites main server.

2) Determine the impact on download times caused by the usage of the P2P transfers.
3) Evaluate the overhead of having extra network requests for the users.
4) Inspect the effects of having transfers between close peers and distant peers.
5) Appraise the behavior of the network of peers when the number of clients is higher.
6) Examine the performance of the network with resources with different sizes to share.

B. Close peers scenario

The first scenario is an interaction between peers that are geographically close to each other. Peer1 was a computer in New York and Peer2 was a computer in Toronto. The first peer is to access the page first, stay on the page and then, one minute later, Peer2 request the HTML page. This scenario intends to demonstrate that of a wise P2P connection where, in a future version, the choice of peers will matter and if a nearby peer already has the data there is no need of fetching it across the globe.

Even though this client still gathers the files from the origin server, it still has to ask the coordinator if are there any peers and this is accounted on the times recorded. This adds a little overhead, as can be seen in Figure 5, even for the clients that do not resort to the network of peers to get their files.

But looking at the setup and download times of the second peer (Figure 6), compared to the first test round where P2P was off and Peer 2 got the images from the origin server, you see some improvement. The biggest improvement can be seen in the File 1 and this can be explained as this is the first data that it is sent, because of being placed before that the others in the HTML document.

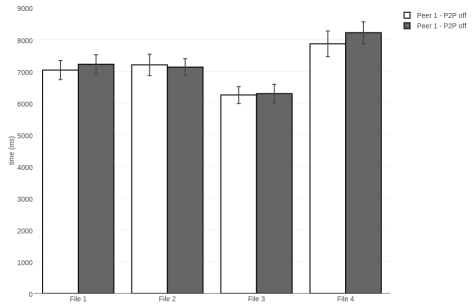

Fig. 5. Image download times of Peer 1 for Scenario 1, Page 1

Since File 1 starts to fetch before and the other files transfers are launched some milliseconds after, File 1 gets almost all bandwidth and finishes in a faster time, damaging the download times of the other resources.

As a conclusion of this test, it brings an overall improvement on the download times on the second peer, when the P2P network is activated.

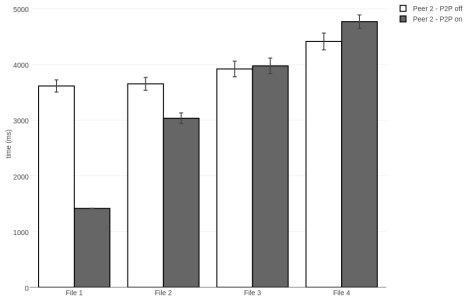

Fig. 6. Image download times of Peer 2 for Scenario 1, Page 1

	Peer 1		
	no P2P	P2P	Variation
File 1	6618	6785	+2.52%
File 2	6461	6846	+5.96%
File 3	5653	6181	+9.34%
File 4	7496	7348	-1.97%
Average	6557	6790	+3.96%

TABLE I
SCENARIO 2, PAGE 1, PEER 1 RESULTS

C. Far away peers scenario

The results for Peer 1 are very similar to those from the first scenario, since this peer is in the same location, New York. It can be verified that the times with P2P activated are a bit more than when it is deactivated, as explained before, and this overhead amounts to about 200-300 milliseconds, in this case. As for Peer 2, located in Singapore, that is significantly distant from the server (Dublin), increased server request times can be observed. What a client in New York takes 6 seconds to download, it takes for that client in Singapore more than 3 times more. When the peer to peer solution was activated, large improvements can be expected.

This test scenario was used mainly to validate, as a comparison with Scenario 1, objective 4 in III-A. Transfers are actually slower (see Tables I and II) between two distant peers but on this scenario was beneficial as the server time were even higher.

D. Multiple peers scenario

The intention of this test scenario was to validate 5 in III-A. It shows that the system behaves very well with more users and that the set up of the peer network is so fast that even

	Peer 2		
	no P2P	P2P	variation
File 1	22959	2023	-91.19%
File 2	21272	2041	-90.41%
File 3	20442	3032	-85.17%
File 4	22772	3807	-83.28%
Average	21861.25	2725.75	-87.51%

TABLE II
SCENARIO 2, PAGE 1, PEER 2 RESULTS

peers that receive the page 1 to 2 seconds later than the first client already have peers to download content from.

It also showed that a correct selection of the peers to download from would bring benefits. Since all selection is random, some peers had increases in the transfer times which could be avoided by connecting to closer peers or go directly to the server.

IV. Conclusion

The goal of this work is to show a solution for website owners and developers to transform their systems into self scalable ones using the power of their own clients. On the websites nowadays, replication of servers and subscription of CDNs are the mechanisms used for supporting large numbers of users. The more users a website has the more infrastructure support it has to have. This is costly and this works for websites that have some sort of prediction about what the near future will be like in terms of popularity.

For smaller websites or companies that are trying to take their web page to the next level, they now have to scale accordingly so that they can respond to the demands of heavy usage or flash crowds. Startups usually tend not to spend so much resources on infrastructure because they are small.

If a user next door is browsing a website does it make sense that another user, right beside him or even in the same city or country, that wants to see it as well makes the same request to the server of that website? That is where the WebRTC comes in. Browser-to-browser communication, while still in development, is already a reality. This work intended to make available a open-source solution, accessible for developers and bringing benefits for both the content providers and the users. This would allow load reduction on the server and create a community of clients of a particular website, that are looking at it at the same time and sharing its resources.

In theory and for small files, if clients are closer to each other than they are from the server, a P2P connection would also increase performance on the data transfers, thus benefiting the users as well. For larger files, where upload capacity is more important, it is possible to download them using several peers to obtain parts of the file, much like the BitTorrent protocol but in the browser.

To prove that it is in fact possible, a prototype of a system that can support this was build. Composed of a regular HTTP server, extended with a coordinator that acts as a connection broker and a reference to where the resources are located within the network, this prototype uses the very modern WebRTC. When some user goes on his browser and inserts a Uniform Resource Locator (URL), the browser contacts the server to fetch the initial files. Then, available for all the modern browsers, the HTML5 WebRTC kicks in and with the help of the coordinator determines if there are peers that can help the server serving its resources. The coordinator is a simple security and lookup element that can be deployed using little resources and that helps peers find each other and establish a connection. Once the peers are connected, any data exchanged between them does not cross any server. It is an direct connection and that means fewer network hops, if they are closer than the server, and better performance.

In the scope of this work only a prototype was implemented that did not apply all what was defined in the architecture like the smart selection of peers and parallel downloads of file parts, which will be future work.

The conclusion for this work is that a solution like the one it was presented is a very good solution to implement on all websites.

As the tests have shown, performance improvements were obtained and even more so when only a simple peer selection mechanism is in place. This would happen in many cases, specially in global websites that do not need to extend their servers whenever or wherever there are aggregations of users. Also, being a self scalable solution, the servers would be much more resistant to sudden popularity bursts while having the same structure costs.

It was demonstrated that with a solution like this, a website would only serve part of the content it serves now and with that could remove/downgrade infrastructural nodes and reduce costs.

References

[1] W3C: World wide web history. http://www.w3.org/History.html

[2] Cisco: Cisco visual networking index: Forecast and methodology. Technical report, Cisco (2013–2018)

[3] Barford, P., Plonka, D.: Characteristics of network traffic flow anomalies. In: Proc. Internet Measurement Workshop (IMW). (2001)

[4] Peng, G.: Cdn: Content distribution network. arXiv preprint cs/0411069 (2004)

[5] Scellato, S., Mascolo, C., Musolesi, M., Crowcroft, J.: Track globally, deliver locally: Improving content delivery networks by tracking geographic social cascades (March 2011)

[6] Shakkottai, S., Johari, R.: Demand-aware content distribution on the internet. Networking, IEEE/ACM Transactions on (2009)

[7] Bronzino, F., Gaeta, R., Grangetto, M.: An adaptive hybrid cdn/p2p solution for content delivery networks. In: Visual Communications and Image Processing (VCIP), 2012 IEEE. (2012)

[8] Bergkvist, A., Burnett, D.C., Jennings, C., Narayanan, A.: Real-time communication between browsers. In: WebRTC 1.0. W3C WebRTC Working Group (2011-2015) http://www.w3.org/TR/2015/WD-webrtc-20150210/.

[9] Offutt, J.: Quality attributes of web software applications. IEEE Software **19**(2) (2002)

[10] Steinmetz, R., Wehrle, K.: Peer-to-peer systems and applications. Volume 3485. Springer Science & Business Media (2005)

[11] Stumm, C.: Client-server system for delivery of online information (June 16 1998) US Patent 5,768,528.

[12] Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Transactions on Internet Technology (TOIT) **2**(2) (2002)

[13] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol–http/1.1. Technical report (1999)

[14] Postel, J.: Transmission control protocol. (1981)

[15] Oram, A.: Peer-to-peer: harnessing the benefits of a disruptive technology. " O'Reilly Media, Inc." (2001)

[16] Bit Torrent, Inc.: Bit torrent. http://www.bittorrent.com

[17] Steinmetz, R., Wehrle, K.: 2. What Is This "Peer-to-Peer" About? Springer (2005)

[18] Stokes, M.: Gnutella. http://rfc-gnutella.sourceforge.net/

[19] John Fanning, S.F., Parker, S.: Napster. http://mp3.about.com/od/history/a/The-History-Of-Napster.htm

[20] eMule-Team: emule. www.emule-project.net/

[21] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communication Review **31**(4) (2001)

[22] Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware 2001, Springer (2001)

[23] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. Volume 31. ACM (2001)

[24] Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Peer-to-Peer Systems. Springer (2002)

[25] Qiu, D., Srikant, R.: Modeling and performance analysis of bittorrent-like peer-to-peer networks. In: ACM SIGCOMM Computer Communication Review. Volume 34., ACM (2004)

[26] Pouwelse, J., Garbacki, P., Epema, D., Sips, H.: The bittorrent p2p file-sharing system: Measurements and analysis. In: Peer-to-Peer Systems IV. Springer (2005)

[27] Neglia, G., Reina, G., Zhang, H., Towsley, D., Venkataramani, A., Danaher, J.: Availability in bittorrent systems. In: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, IEEE (2007)

[28] Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites. In: Proceedings of the 11th international conference on World Wide Web, ACM (2002)

[29] Stading, T., Maniatis, P., Baker, M.: Peer-to-peer caching schemes to address flash crowds. In: Peer-to-Peer Systems. Springer (2002)

[30] Gadde, S., Chase, J., Rabinovich, M.: Web caching and content distribution: A view from the interior. Computer Communications **24**(2) (2001)

[31] Hofmann, M., Beaumont, L.: Content Networking: Architecture, Protocols, and Practice. The Morgan Kaufmann series in networking. Morgan Kaufmann (2005)

[32] Mulerikkal, J.P., Khalil, I.: An architecture for distributed content delivery network. In: Networks, 2007. ICON 2007. 15th IEEE International Conference on, IEEE (2007) 359–364

[33] Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Surveys (CSUR) **36**(4) (2004)

[34] Nygren, E., Sitaraman, R.K., Sun, J.: The akamai network: a platform for high-performance internet applications. ACM SIGOPS Operating Systems Review **44**(3) (2010)

[35] Akamai Technologies: Akamai netsession. http://www.akamai.com/client

[36] Zhao, M., Aditya, P., Chen, A., Lin, Y., Haeberlen, A., Druschel, P., Maggs, B., Wishon, B., Ponec, M.: Peer-assisted content distribution in akamai netsession. In: Proceedings of the 2013 conference on Internet measurement conference, ACM (2013)

[37] Princeton University: Codeen. http://codeen.cs.princeton.edu/

[38] Wang, L., Park, K., Pang, R., Pai, V.S., Peterson, L.L.: Reliability and security in the codeen content distribution network. In: USENIX Annual Technical Conference, General Track. (2004)

[39] W3C: Webrtc. http://www.webrtc.org/

[40] Johnston, A.B., Burnett, D.C.: WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web. Digital Codex LLC (2012)

[41] Vogt, C., Werner, M.J., Schmidt, T.C.: Leveraging webrtc for p2p content distribution in web browsers. In: Network Protocols (ICNP), 2013 21st IEEE International Conference on, IEEE (2013)

[42] Ng, K.F., Ching, M.Y., Liu, Y., Cai, T., Li, L., Chou, W.: A p2p-mcu approach to multi-party video conference with webrtc. International Journal of Future Computer and Communication **3**(5) (2014)

[43] Mozilla Foundation: Firefox hello. https://www.mozilla.org/en-US/firefox/hello/

[44] Chiang, C.Y., Chen, Y.L., Tsai, P.S., Yuan, S.M.: A video conferencing system based on webrtc for seniors. In: Trustworthy Systems and their Applications (TSA), 2014 International Conference on, IEEE (2014)

[45] Werner, M.J., Vogt, C., Schmidt, T.C.: Let our browsers socialize: Building user-centric content communities on webrtc. In: Distributed Computing Systems Workshops (ICDCSW), 2014 IEEE 34th International Conference on, IEEE (2014)

[46] Grigorik, I.: High Performance Browser Networking: What every web developer should know about networking and web performance. " O'Reilly Media, Inc." (2013)

[47] Cisco: How nat works. http://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/26704-nat-faq-00.html

[48] Tuexen, M., Loreto, S., Jesup, R.: Webrtc data channel establishment protocol. (2015)

[49] F. Aboukhadijeh: Webtorrent. https://webtorrent.io/

[50] Zhang, L., Zhou, F., Mislove, A., Sundaram, R.: Maygh: Building a cdn from client web browsers. In: Eurosys'13. (2013)

[51] J. Hiesey, F. Aboukhadijeh, A. Raja: Peercdn. https://peercdn.com/ (2013)

[52] Swarm Labs, LLC: Swarmify. http://swarmify.com/ (2014)

[53] Viblast Ltd.: Viblast. http://viblast.com/ (2015)

[54] M. Bu, E. Zhang: Peerjs. http://peerjs.com/ (2014)

[55] Peer5, Inc.: Peer5. https://peer5.com/ (2015)

[56] Peer5, Inc.: Sharefest. https://www.sharefest.me/ (2015)

[57] Ihm, S., Pai, V.S.: Towards understanding modern web traffic. In: Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, ACM (2011) 295–312

[58] Pries, R., Magyari, Z., Tran-Gia, P.: An http web traffic model based on the top one million visited web pages. In: Next Generation Internet (NGI), 2012 8th EURO-NGI Conference on, IEEE (2012) 133–139

[59] Fielding, R.T.: Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine (2000)