Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Research Report

Max Jonas Werner

Peer-to-Peer Networking using Open Web Technologies

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Max Jonas Werner

Peer-to-Peer Networking using Open Web Technologies

Research Report submitted in the context of Seminar MINF-AW1

in the course Master of Science
at the Department of Computer Science
at the Faculty of Engineering and Computer Science
of Hamburg University of Applied Sciences

Submitted on: 15 February 2013

Max Jonas Werner

Title of the paper

Peer-to-Peer Networking using Open Web Technologies

Keywords

Peer-to-peer, Web, Web Browser, Chord, DHT, JavaScript, DOM, HTML5, WebRTC

Abstract

This report presents the results of my research in the area of peer-to-peer systems on the one hand and web technologies on the other hand. The aim was to found a solid scientific basis for following the vision of implementing a peer-to-peer system using native browser technologies only, namely WebRTC.

Contents

1 Introduction

The web is since its incarnation in 1996 [4] formed by classic client/server architectures using the HTTP protocol. There are several use cases, though, where a peer-to-peer (P2P) approach is preferable, e.g. for video communication or secure file transfer between two parties. Additionally the reliance on servers that users have no control over poses a great security and privacy risk for sensitive data.

A set of new web technologies is currently being developed to enable a real browser-to-browser communication channel. The WebRTC standard defining these technologies consists of an API [3] defined by the W3C and a set of underlying protocols defined by the IETF Rtcweb Working Group[1]. Additionally, with the WebSocket protocol specified in RFC 6455 [7] – which may be used for signalling – the IETF has defined a means for two-way communication between two nodes, with an accompanying DOM API specified by the W3C [10].

In the past peer-to-peer overlay techniques were based on installing a piece of specialized software (the peer-to-peer software) on every node; examples are Gnutella[2] and BitTorrent[3]. This is likely to be a reason for the rather low acceptance rate of peer-to-peer technologies in the mainstream user base. Nowadays, though, every computing device comes with a web browser pre-installed, so that a peer-to-peer network based solely on web technologies has the potential to reach every user immediately without the need to install additional software. The web would then form a universal platform.

This research report serves as a foundation to investigate the possibilities offered by WebRTC to build a P2P web infrastructure. In Chapter 2 I'll describe the basic technologies enabling such an infrastructure and current research topics. Chapter 3 describes in more detail the vision that I'll follow in the next semesters to build a P2P web application.

[1] http://tools.ietf.org/wg/rtcweb/
[2] http://rfc-gnutella.sourceforge.net/developer/stable/index.html
[3] http://www.bittorrent.org/beps/bep_0003.html

2 Fundamentals

In this chapter I'll describe the technologies that my work on a WebRTC based P2P application will be based on. Principles of P2P networks are describe in Section 2.1, the web technology basics are explained in Section 2.2. Each section contains information about current implementations as well as research aspects of each discipline.

2.1 Peer-to-peer Networking

Steinmetz and Wehrle define P2P systems in [19] (extending the definition by Oram [16]) as

> a self-organizing system of equal, autonomous entities (peers) [which] aims for the shared usage of distributed resources in a networked environment avoiding central services. In short, it is a system with completely decentralized self-organization and resource usage.

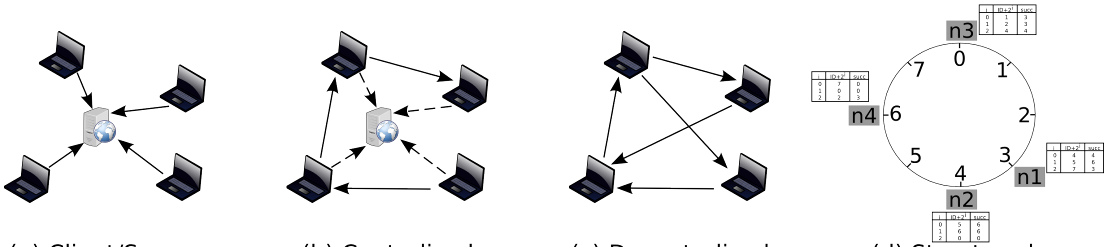
They further state that future Internet-based applications are ought to provide the following properties, which P2P systems are able to supply:

- Scalability with regards to bandwidth, storage and computing capacity
- Stability with regards to security (e.g. DoS attacks) and reliability (hardware or software failures)
- Flexibility/Quality of Service with regards to integration of new services or features.

The differences between the various P2P and the client/server systems are depicted in Figure 2.1. Here you can clearly see that every node in a P2P network acts as a client and as a server at the same time (best described by the term “servent” as a portmanteau of “server” and “client”). As such there's no single point of failure in the network anymore; if one node fails the functionality of this node may be taken over by another node.

Internet P2P systems run atop the IP infrastructure in the form of overlay networks. They are usually divided into structured and unstructured systems where unstructured systems are

Figure 2.1: Comparison between Client/Server and the different P2P solutions



(a) Client/Server (b) Centralized (c) Decentralized (d) Structured

either centralized (e.g. by using a central lookup server) or decentralized (employing a flooding mechanism to discover nodes and content). An example of a centralized P2P system is the original form of the file-sharing service Napster [19] where all search queries were conducted against a central server whilst the nodes only delivered content. Decentralized systems include the Gnutella 0.4 protocol[1] and FreeNet[2].

Structured P2P systems behave in a very different way and the most widespread implementations incorporate distributed hash tables (DHT). In DHT-based overlays a unique hash is computed for each object (e.g. a file) and is assigned to a certain node. This way the search for an object can be conducted in logarithmic time as Table 2.1 depicts.

Table 2.1: Overview of the P2P scaling properties

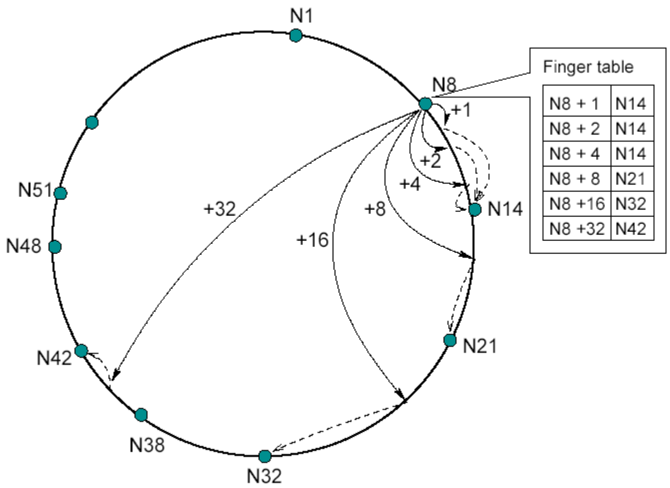
	Centralized	Decentralized	Structured
Communication overhead	$O(1)$	$O(n)$	$O(log(n))$
Node states	$O(n)$	$O(1)$	$O(log(n))$

2.1.1 State of the Art

Structured P2P systems represent a very good compromise between centralized systems (single point of failure, huge node state complexity) and decentralized systems (huge communication overhead). This is why they'll be focused in this paper.

[1] http://rfc-gnutella.sourceforge.net/developer/stable/index.html
[2] https://freenetproject.org/whatis.html

Figure 2.2: A Chord ring topology



Chord

One prominent DHT approach – and one of the first around – is the Chord protocol and algorithm [20], which defines a simple and effective DHT technique. To understand the general concepts of a DHT Chord is a good starting point.

The keys from the available key space 2^m (with m usually being 160 since SHA-1 is used as consistent hashing function) are placed on a circle as pictured in Figure 2.2.

Every node stores m other nodes together with their IP address and port number in a so called finger table. The ID of each node is assigned from the available key space so that the nodes are placed on the Chord ring, too, having a successor s and a predecessor p. Every node is responsible for the keys from $p-1$ to its own ID so that the whole key space is always covered by the available nodes. When a node wants to retrieve data with key k it looks up the node that is the closest successor of k and sends the request to that node. This way a logarithmic search complexity of $O(log(n))$ is assured.

Chord is a quite famous protocol since it's a theoretically proven model and relatively easy to implement. This is also the reason why it's a very good candidate to be used in the prototypical implementation mentioned in Section 3.3. One disadvantage is that node proximity information in the underlying Internet is not taken into account in the communication. Other DHT approaches such as Pastry, Tapestry and CAN incorporate methods for proximity neighbor selection or geographic layout but are more complex to implement. An overview of the different proximity methods and their applicability to certain DHT algorithms is given in [5].

2.1.2 Research Aspects

The current research topics in P2P systems gather around different areas such as fundamentals like content distribution and caching but also on advanced topics such as quality of service, trust and security as well as location based routing.

In [22] Vijendran and Thavamani present different current techniques of content caching and replication to overcome problems in the underlying network structure such as high latency and traffic. Libin Yang and Wei Lou also deal with QoS in [24]. They present a means for assisting service providers in streaming content to mobile devices using a P2P approach. What's rather innovative in their approach is the consideration of economic incentives to run such a peer-assisted service.

P2P systems have by their very nature no central authority that could authenticate nodes in the network. Thus, trust and content integrity have to be guaranteed differently. The potential problem of untrusted nodes taking part in a P2P system is covered by Huang et al. in [11]. Another example of research on the applicability in mobile environments is presented by Bakht et al. [1]. They developed a solution for mobile devices to perform neighbor discovery without infrastructure support.

Conferences discussing P2P system design include the International Workshop on Hot Topics in Peer-to-peer computing and Online Social Networking (HotPOST)[3] as well as the IEEE International Conference on Peer-to-Peer Computing[4].

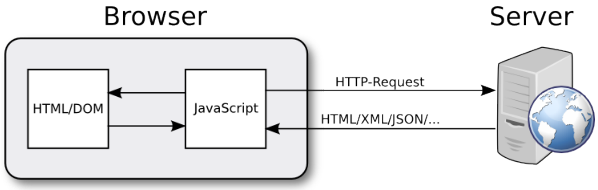
2.2 Web Technologies

In order to understand the benefits of using web technologies for certain tasks one has to have knowledge of what comprises the World Wide Web as we know it today. The general approach to a definition can be broken down into three fundamental concepts:

- Identification of resources: URIs[5]
- Transfer of data: HTTP[6]
- Presentation of information: HTML[7]

[3] http://cseweb.ucsd.edu/~tixu/HotPost/
[4] http://www.p2p12.org/
[5] http://tools.ietf.org/html/rfc3986
[6] http://tools.ietf.org/html/rfc2616
[7] http://www.w3.org/TR/html5/

Figure 2.3: Schematic view of an Ajax request



Besides the named three standards there exist further technologies like CSS (for layouting/designing), JavaScript (for programmatic interaction with documents), HTTPS (for secure transfer using HTTP over SSL/TLS) and the Document Object Model (DOM, used to interact with the presentation programmatically, e.g. via JavaScript).

2.2.1 State of the Art

Until the beginning of the 21st century the web was comprised mostly of static web pages that barely let users interact with the content or the server side. New technologies like the XMLHttpRequest object and improved JavaScript performance in the browsers served the transformation of web pages into web applications that could be used interactively; applications such as Google Maps gained popularity. Certain workarounds for enabling push events from server to client like HTTP long-polling[8] are used for realtime use cases such as web chats.

Currently there exist three major standardized DOM APIs that let programmers build interactive and realtime applications: XMLHttpRequest, WebSocket and Server-sent Events. These are described in more detail in the following sections.

XMLHttpRequest

The XMLHttpRequest object makes it possible to asynchronously open a connection to a remote server using JavaScript as outlined in Figure 2.3. This technique is known as Ajax which initially stood for Asynchronous JavaScript and XML because in the beginnings of Ajax it was used to transfer mainly XML documents; nowadays most applications transfer JSON strings or otherwise lightweight serialized data. The benefit of using Ajax is that certain actions conducted by the user (e.g. clicking a button) don't need to result in a full page refresh anymore. This way the overhead of retrieving new data from the server is kept to a minimum and the web application becomes more responsive.

[8] http://en.wikipedia.org/wiki/Push_technology#Long_polling

Figure 2.4: Schematic view of a WebSocket connection

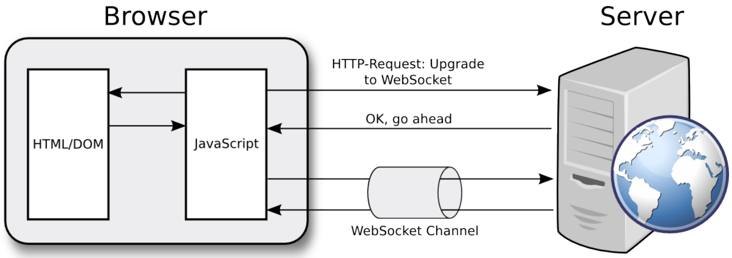
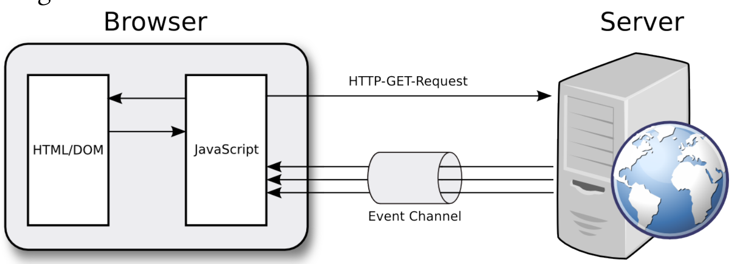


Figure 2.5: Schematic view of a Server-sent Event connection



WebSocket

The concept of WebSockets extends the use cases enabled by XMLHttpRequest by the possibility to acquire a bibirectional channel between client and server, as depicted in Figure 2.4. The WebSocket standard introduces two new URI schemes, `ws:` and `wss:`, for unencrypted and encrypted connections, respectively. The WebSocket protocol uses the HTTP Upgrade mechanism in the initial handshake (a simple HTTP GET-like request) to switch from HTTP to WebSocket. After a successful establishment browser and server are capable of communicating in a bidirectional way.

Server-sent Events

This W3C specification adds an additional DOM interface to browsers: EventSource. A programmer may instantiate an EventSource object providing a URL and the browser opens a connection to the URL (by adhering to the same-origin policy) that is held open. This way the server may push data to the client when it becomes available. Server-sent Events qualify as a lightweight alternative to WebSockets while enabling similar use cases such as auto-updating news feeds. Figure 2.5 outlines the mechanism of Server-sent Events.

2.2.2 Research Aspects

The current research topics in the web sector cover a wide area ranging from mobility, browser performance and security to social networking, deployment aspects and cloud computing:

Wang et al. [23] investigated the possibilities of improving mobile browser performance using techniques such as caching, pre-fetching and speculative loading. Covering performance aspects on the server/deployment side Traverso et al. [21] researched on the possibilities of implementing a content distribution network (CDN) where the content is distributed by examining the social network of authors. Also on the social networking side, the authors Meij et al. [12] have worked on possibilities to inject semantic information into microblog postings. The authors propose a way for identifying concepts in postings by linking them to Wikipedia articles.

The better usage of hardware resources on the client by executing code concurrently in the browser is targeted by Erbad et al. [6]; the authors leverage the Web Workers API[9]. A security-related paper has been published by Singh et al. [18] which describes possibilities of eliminating the integrity constraints that HTTPS poses on content by developing an alternative protocol named HTTPi. The hot topic of cloud computing shall be represented by a paper from three Spanish universities [14]: The authors focus on the problems current cloud users have when they'd like to switch between different cloud providers since there's no unified API and a vendor lock-in happens regularly.

Important web-related conferences covering the mentioned topics include the International World Wide Web Conferences series[10], the International Conference on Web Services[11], the International Workshop on Web APIs and Service Mashups[12] and the International Conference on Web Search and Data Mining[13].

[9] http://dev.w3.org/html5/workers/
[10] http://wwwconference.org/
[11] http://www.icws.org/
[12] http://mashups2012.aifb.kit.edu/
[13] http://wsdm2012.org/

3 Peer-to-peer using Web Technologies

The vision I'll push forward next is to integrate the P2P system approach with the world wide web. In the next section I describe the technology enabling this move and afterwards point out current research aspects on the topic. In the last section I'll give a brief outlook on the next steps.

3.1 What WebRTC is about

The basic idea behind WebRTC is depicted in Figure 3.1. Like with the techniques described in Subsection 2.2.1 WebRTC extends the DOM by methods to request a direct connection to another host. This connection can either be comprised of an audio stream, a video stream, a pure data channel or a mix of them. As Figure 3.1 shows the only function that an HTTP server is necessary for is to initially request the web application. From then on the JavaScript code is able to directly communicate with other hosts without an intermediary server.

Currently the specification of WebRTC is in active development and the API as well as the underlying protocols are still in heavy flux. Also, it is not clear, yet, which functionalities (besides audio, video and data channels) will be included in the final specification. There's active discussion of including a protocol for realtime text, for example.

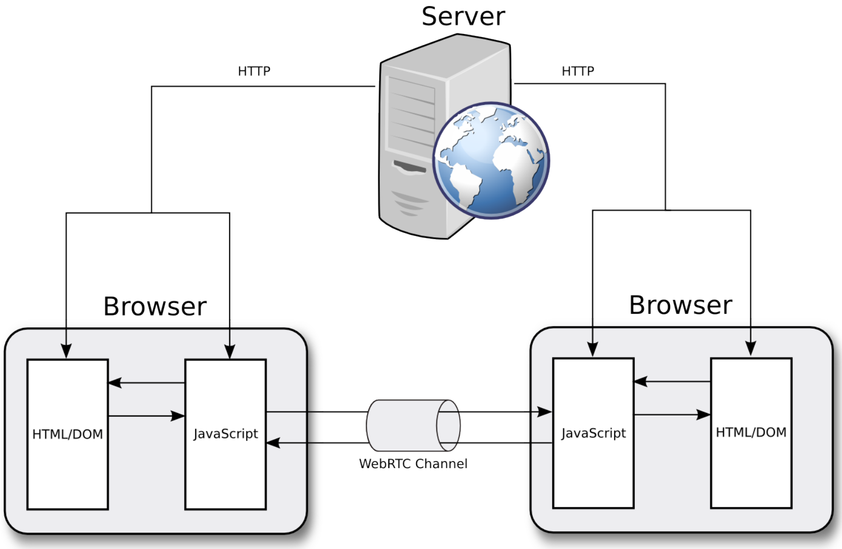
3.2 Current Research

Most approaches towards a peer-to-peer web infrastructure are currently targeting the server-side (e.g. Bari et al. [2] and Mittal et al. [15]) or depend on third-party software to be installed (e.g. the Akamai NetSession Interface[1]).

Research on leveraging native browser technologies – each achieving a different set of goals – is already being conducted: Meyn [13] examines a way to distribute the load and stream video content between browsers using WebRTC, thus reducing the bandwidth cost of content providers. The author uses a BitTorrent-like architecture involving a tracking server for discovering content.

[1] http://www.akamai.com/html/solutions/client_overview.html

Figure 3.1: Schematic view of a WebRTC connection



However, most current implementations and demos leveraging WebRTC are currently focussing on audio/video communication using SIP, like sipML5[2].

Ownership of personal data in web applications is a matter of ongoing passionate discussion. The main problem is that data resides on the providers' servers. A peer-to-peer architecture has the potential to mitigate the impacts of storing data on foreign servers since it can be distributed and encrypted. Hagemeister [9] investigates the possibilities of a censorship-resistant peer-to-peer collaboration architecture, but without focussing on web technologies. Fifield et al. [8] show a way to evade censorship by making every browser a proxy using WebSockets.

3.3 Outlook

Despite the ongoing research a real peer-to-peer overlay network using WebRTC/WebSockets has yet to be investigated. Such a network would work completely without a central authority. Additionally – using the web as application platform – the opportunities of a peer-to-peer web aren't exhausted at this point. HTML5 technologies like the Geolocation API [17] provide application developers with a powerful tool set for creating location-aware peer-to-peer groups for example. WebRTC is the enabling standard to push forward a new mindset in web application development. As a next step I'll go further by doing a prototypical implementation of a DHT (most probably Chord) in the browser using WebRTC.

[2] http://sipml5.org/

Bibliography

[1] Mehedi Bakht, John Carlson, Alexander Loeb, and Robin Kravets. United we find: enabling mobile devices to cooperate for efficient neighbor discovery. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, pages 11:1–11:6, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1207-3. doi: 10.1145/2162081.2162097. URL http://doi.acm.org/10.1145/2162081.2162097.

[2] M.F. Bari, M.R. Haque, R. Ahmed, R. Boutaba, and B. Mathieu. Persistent naming for p2p web hosting. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 270 –279, 31 2011-sept. 2 2011. doi: 10.1109/P2P.2011.6038745.

[3] Adam Bergkvist, Daniel C. Burnett, Cullen Jennings, and Anant Narayanan. WebRTC 1.0: Real-time Communication Between Browsers. W3C working draft, W3C, October 2012. URL http://www.w3.org/TR/2012/WD-webrtc-20120209/.

[4] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext transfer protocol – HTTP/1.0. RFC 1945, Internet Engineering Task Force, May 1996. URL http://www.rfc-editor.org/rfc/rfc1945.txt.

[5] Miguel Castro, Peter Druschel, Y. Charlie Hu, and Antony Rowstron. Exploiting network proximity in distributed hash tables. In *in International Workshop on Future Directions in Distributed Computing (FuDiCo)*, pages 52–55, 2002.

[6] Aiman Erbad, Norman C. Hutchinson, and Charles Krasic. Doha: scalable real-time web applications through adaptive concurrent execution. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 161–170, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187859. URL http://doi.acm.org/10.1145/2187836.2187859.

[7] I. Fette and A. Melnikov. The WebSocket Protocol. Rfc, Internet Engineering Task Force, December 2011. URL http://www.rfc-editor.org/rfc/rfc6455.txt.

[8] David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Dan Boneh, Roger Dingledine, and Phil Porras. Evading censorship with browser-based proxies. In *Proceedings of the 12th international conference on Privacy Enhancing Technologies*, PETS'12, pages 239–258, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-31679-1. doi: 10.1007/978-3-642-31680-7_13. URL http://dx.doi.org/10.1007/978-3-642-31680-7_13.

[9] Philipp Hagemeister. Censorship-resistant Collaboration with a Hybrid DTN/P2P Network. Master's thesis, HHU Düsseldorf, March 2012.

[10] Ian Hickson. The WebSocket API. W3C candidate recommendation, W3C, September 2012. URL http://www.w3.org/TR/2012/CR-websockets-20120920/.

[11] Zhihua Huang, Songnian Lu, Aixin Zhang, and Jie Gu. Impact of feedback on trust in p2p networks. *Journal of Networks*, 7(8), 2012. URL http://ojs.academypublisher.com/index.php/jnw/article/view/jnw070811821188.

[12] Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 563–572, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5. doi: 10.1145/2124295.2124364. URL http://doi.acm.org/10.1145/2124295.2124364.

[13] A. J. Meyn. Browser to Browser Media Streaming with HTML5. Master's thesis, Technical University of Denmark, DTU Informatics, E-mail: reception@imm.dtu.dk, Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2012. URL http://www.imm.dtu.dk/English.aspx. Supervised by Associate Professor Christian Probst, probst@imm.dtu.dk, DTU Informatics.

[14] Javier Miranda, Juan Manuel Murillo, Joaquín Guillén, and Carlos Canal. Identifying adaptation needs to avoid the vendor lock-in effect in the deployment of cloud sbas. In *Proceedings of the 2nd International Workshop on Adaptive Services for the Future Internet and 6th International Workshop on Web APIs and Service Mashups*, WAS4FI-Mashups '12, pages 12–19, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1566-1. doi: 10.1145/2377836.2377841. URL http://doi.acm.org/10.1145/2377836.2377841.

[15] P. Mittal, A. Dixit, and A. K. Sharma. A scalable, extensible web crawler based on p2p overlay networks. In *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, ICWET '10, pages 159–162, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-812-4. doi: 10.1145/1741906.1741937. URL http://doi.acm.org/10.1145/1741906.1741937.

[16] Andy Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001. ISBN 059600110X.

[17] Andrei Popescu. Geolocation API Specification. W3C proposed recommendation, W3C, May 2012. URL http://www.w3.org/TR/2012/PR-geolocation-API-20120510/.

[18] Kapil Singh, Helen J. Wang, Alexander Moshchuk, Collin Jackson, and Wenke Lee. Practical end-to-end web content integrity. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 659–668, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187926. URL http://doi.acm.org/10.1145/2187836.2187926.

[19] Ralf Steinmetz and Klaus Wehrle, editors. *Peer-to-Peer Systems and Applications*, volume LNCS 3485. Springer-Verlag, 2005.

[20] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4):149–160, August 2001. ISSN 0146-4833. doi: 10.1145/964723.383071. URL http://doi.acm.org/10.1145/964723.383071.

[21] Stefano Traverso, K'evin Huguenin, Ionut Trestian, Vijay Erramilli, Nikolaos Laoutaris, and Kostantina Papagiannaki. Tailgate: handling long-tail content with a little help from friends. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 151–160, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187858. URL http://doi.acm.org/10.1145/2187836.2187858.

[22] Anna Saro Vijendran and S. Thavamani. Survey of caching and replica placement algorithm for content distribution in peer to peer overlay networks. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, CCSEIT '12, pages 248–252, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1310-0. doi: 10.1145/2393216.2393259. URL http://doi.acm.org/10.1145/2393216.2393259.

[23] Zhen Wang, Felix Xiaozhu Lin, Lin Zhong, and Mansoor Chishtie. How far can client-only solutions go for mobile browser speed? In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 31–40, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187842. URL http://doi.acm.org/10.1145/2187836.2187842.

[24] Libin Yang and Wei Lou. A contract-ruled economic model for qos guarantee in mobile peer-to-peer streaming services. In *Proceedings of the 2012 IEEE 20th International Workshop on*

Quality of Service, IWQoS '12, pages 34:1–34:4, Piscataway, NJ, USA, 2012. IEEE Press. ISBN 978-1-4673-1298-1. URL http://dl.acm.org/citation.cfm?id=2330748.2330782.