LUND UNIVERSITY

Concurrency and locality of content demand

Nordell, Viktor; Aurelius, Andreas; Gavler, Anders; Arvidsson, Åke; Kihl, Maria

2013

Link to publication

Citation for published version (APA):
Nordell, V., Aurelius, A., Gavler, A., Arvidsson, Å., & Kihl, M. (2013). *Concurrency and locality of content demand*. Paper presented at First International Workshop on Quality Monitoring (IWQM 2013) Conference, Paris, France.

Total number of authors:
5

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Concurrency and locality of content demand

Viktor Nordell[1], Andreas Aurelius[1,3], Anders Gavler[1], Åke Arvidsson[2], Maria Kihl[3]
[1]Acreo AB, Networking and transmission lab, Electrum 236, SE-164 40 Kista, Sweden
[2]Ericsson AB, Ericsson Research; Packet Technologies, SE-164 80 Kista, Sweden
[3]Dept. of Electrical and Information Technology, Lund University, Sweden

***Abstract*—Distribution of media data over the Internet is increasing in popularity and volume. This poses challenges not only for network operators but also for service providers when it comes to serving the demand in a cost-efficient way. In this paper, we approach this problem by investigating the potential of co-operative approaches where locality in space (users in the same network) and locality in time (concurrent downloads) are exploited such that as many requests as possible may be handled inside the access and metro networks. This approach may contribute not only to reducing transport costs (less traffic in core networks and at peering points) by but also improve the end user experience (by reduced round trip times and exclusion of some possible bottlenecks). To this end we develop a method to measure the possible gains from, firstly, optimal handling of concurrent downloads and, secondly, optimal utilization local availability. We apply the method to BitTorrent data from two metropolitan access networks and find that the bandwidth savings amount to between 10% and 20% when optimizing concurrent downloads and between 56% and 66% when exploiting local availability with a simulated network cache.**

***Index Terms*—Locality, content demands, concurrency, access networks, P2P, ICN, CDN, BitTorrent, distributed caches**

I. Introduction

Internet media applications, such as TV and music services, have a heavy impact on today's networks, and there is a common understanding that the capacity demands will continue to increase at least over the next few years. In order to design service-optimized architectures for networks and protocols in future content delivery solutions, it is important to understand the interplay between network topology, service characteristics and user interests.

One solution is to keep the traffic more local, by using various locality-aware mechanisms, for example local caches and peer-to-peer offloading. Caches could be deployed at strategic points in the networks where devices may store data and where other devices may fetch the content from, or via transparent caching proxies. With peer-to-peer offloading, the content is fully or partly distributed between clients using software which also runs in a distributed fashion, most commonly on the client devices themselves. These solutions could be made locality-aware so that clients prefer peers that are geographically close to themselves.

These solutions have benefits for all parties in the content distribution chain, from content providers, through network operators, and finally to end customers. Content providers will benefit from reduced bandwidth costs as less traffic flows from their own servers. Network operators will see lower costs for peering as less external traffic is transited via other providers and, with local bridging enabled, for their core networks as less internal traffic is tromboned (none optimal routing/switching design) internally. Finally end customers may experience better QoS in terms of lower delay and higher throughput as content is fetched from nearby locations in their local network.

Inter ISP traffic analysis has been presented in, *e.g.*, [1], where the focus is on traffic flows between ISPs but not the actual content. Various measurement studies of P2P applications have shown that there are large bandwidth gains for ISPs if caches can be implemented in the networks, see for example [2], [3], [4]. In [5], a university network was monitored for three days with focus on BitTorrent files. The analysis showed that 10.4%–18.2% of the content files could be downloaded locally. In [6], it was shown that up to 88% of the traffic for a P2P live video distribution could be kept locally within an ISP if the P2P application used a locality–aware peer selection scheme. Further, different locality–aware content fetching solutions have been proposed, with or without assistance from ISPs [7], [8], [9], [10], [11], [12].

The purpose of this paper is to investigate the extent to which traffic related to content such as videos, music *etc*. could be kept local. To this end we examine how the demand is distributed between users in access networks, especially focusing on shared user interests and the timing of content accesses. The minimum (exploiting only direct concurrency) and maximum (exploiting an infinite network cache) potential for bandwidth savings in realistic cases are evaluated.

The contribution of the paper is threefold. First, we analyst data from *ordinary, residential users* (as opposed to campus network users). Second, the data is collected close to the end users, which means that the measurement is rather distributed, and we can quantify the gains from *relatively small user groups* (on the order of thousands). Third, we *propose a scheme for analyzing direct user concurrency and cache assisted concurrency* of content demand. The measurement data is retrieved from Bit Torrent tracker traffic in two fiber based access networks in Sweden with residential users.

The measurement procedure and the networks are described in Section II-B. The BitTorrent protocol is described briefly in Sections III-A and III-B. The methods for post processing of data and for defining sessions are presented in Sections III-C and III-D. In Section III-E, we provide precise definitions of direct user concurrency and network assisted concurrency. Finally, our measurement results are presented in Section IV and the conclusions are given in Section V.

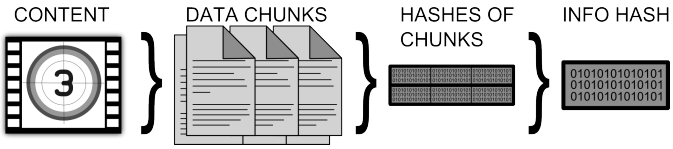

Figure 1. The info-hash is based on hashes of chunks of the real content, this means that the same content will always have the same info-hash.

II. Measurement procedure and networks

In this section, the measurements are described regarding the networks and the equipment used.

A. Networks

The data in this paper is collected in two residential metropolitan access networks in Sweden. The networks are fiber based, and provide connectivity to residential users, with connection speeds ranging from 1 Megabit per second (Mbps) to 100 Mbps. Most subscriptions are symmetrical. The networks are denoted network one and network two. Network one has a customer base of approximately 5000 households, and network two has 1500 households.

B. Measurement procedure

The measurements in this paper are performed by the network operators. Post processed data is made available to the researchers for analysis. The equipment for acquiring data is PacketLogic [13], a traffic management device capable of identifying IP traffic by deep-packet-inspection and deep-flow-inspection. A signature database is used for matching to the traffic characteristics. A large portion of the traffic is identified and matched to a signature, around 95%. The probe location in the networks is on the Internet edge, which means that all traffic between the metro network and the Internet is monitored.

There are several ways to retrieve statistics from the device. The most common way is to use the statistics database, which stores data per 5 minutes, per host and application (a host here is an IP number). In order to enable post processing of data, traffic matching certain conditions may be dumped to a pcap file. This has been the main method for collection of the statistics used in this study.

III. Data processing and session calculation

In this section the data processing procedure is described. Also specific characteristics of the BitTorrent protocol are described, since the understanding of this is essential for the processing procedure.

A. BitTorrent Info-hash

Each object distributed with the BitTorrent protocol is described in a "torrent" file which contains information about the content and information to bootstrap the download of the content. The content information includes a list of files (data chunks) and a list of hashes (of the data chunks), these hashes were in turn calculated from chunks of the original content. By hashing all the file chunk hashes, a unique hash of the entire content called the info-hash is obtained, cf. Figure 1. This means that the info-hash uniquely identifies the content, and it is therefore possible to know that two clients with identical info-hashes are downloading exactly the same content.

B. BitTorrent Tracker

Traditionally each client in the BitTorrent system would contact the BitTorrent Tracker to receive new peers that participate in the distribution of a BitTorrent file. This is still partly the case today, but only as one of several possible sources of peers as, *e.g.*, modern clients also exchange peers directly between each other. For the work in this paper it is important to note that most clients still contact trackers and that this traffic can be intercepted by traffic analyzers using deep packet inspection (DPI).

A BitTorrent client will send information to the tracker via HTTP GET messages, containing a set of parameters encoded in the request URL [14]. The most important of these parameters for this work is the "info_hash" parameter which specifies the info hash, which, as explained above, identifies the content that the client is downloading. Other interesting parameters in our case are "left", "downloaded" and "uploaded", which respectively specify the amount left to download, the downloaded amount and the uploaded amount. The BitTorrent client continues to send these messages after it has fully downloaded the content, as long as it keeps this content active and allows for uploading.

We remark that encrypted HTTP traffic cannot be analyzed as above, but fortunately this seems to be quite uncommon among BitTorrent trackers.[1] The results of this study should therefore not be greatly affected; even if some BitTorrent traffic may be left out of the measurements because of encryption, the locality results will be valid for the captured traffic.

C. Data processing

The measurement device described in Section II-B was configured to dump packets matching BitTorrent tracker traffic into PCAP packet dump files.

These files where downloaded from the collection point, and the addresses were anonymized. Data was analyzed by Wireshark [15] and converted to XML after which the following parameters were extracted:

- The time when the request to the tracker was made.
- The source IP address of the client, hashed and salted to protect the privacy of individual users.
- The info hash that the request refers to.
- The amount of bytes downloaded, uploaded and left to download.
- The host name of the tracker.

The extracted information was inserted in an SQL database, and used in the next step as described below. Using the captured information, an approximation of size of the BitTorrent files was calculated, based on the maximum value of the "left"

[1] A large random sample of trackers was downloaded, and it was found that none of them were HTTPS enabled. Nevertheless, encryption may be supported by some trackers.

parameter reported by the BitTorrent clients. This gives a good estimation of file size, since a starting client will report the full size of content, as it has not yet started the actual download.

D. Session identification

In order to obtain detailed records of user activities, *i.e.*, downloads or uploads of specific pieces of content, it is necessary to identify sessions for each combination of user (IP address) and content (info hash) delimited in time by a start time and an end time. A particular challenge in this context is that we must cater for the fact that users may turn off their computers temporarily (*i.e.*, a single user-content combination may be described by multiple sessions). Our solution to this problem was to form user-content sessions from the extracted data in the following manner:

1) Extract all data points for (observations of) client A and info hash X, sorted by ascending time.
2) Set session starting time from the time of the first data point.
   a) Analyst next data point.
   b) Is the time stamp of the above data point within a time period α from the last data point?
      i) If yes, assume that the session continues, go to a.
      ii) If no, assume that the session has ended, insert the session in to the database and go to 2.

This calculation requires a parameter, α which represents the maximum time period between two data points for which the two data points can be considered to be part of the same session.

To find suitable values for α the "interval" parameter which is reported from the tracker was examined. This parameter specifies the recommended interval between tracker updates. The result of a small sample of data showed that, by far, the dominating choices for update interval were 30 minutes and 100 minutes, and no message specified a number higher than 100 minutes. We propose that α should be slightly longer than this and thus settled for $\alpha = 101$ minutes.

E. Concurrency

Direct user concurrency and cache assisted concurrency are two concepts that may be exploited in a locality-aware mechanism. We define direct user concurrency as two users that download and/or upload the same content at the same time, and we define network assisted concurrency as one user downloading content that has already been downloaded and now exists in a cache located in operator domain of the network.

From the data, direct user concurrency can be determined, as illustrated by the example in Figure 2.

The example contains two clients, A and B, and a network cache. Both A and B are at some point active with torrent X. Please note that there is a difference between being active and still downloading, and being active with fully downloaded torrent X, denoted by dashed and full lines, respectively. When

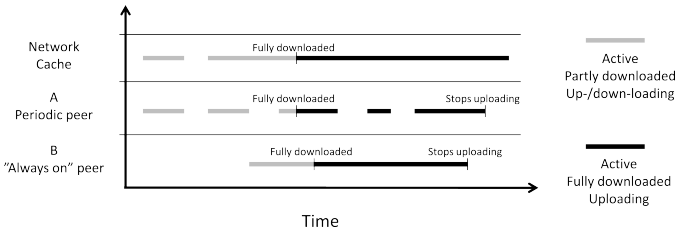

Figure 2. Example of calculation of concurrent active sessions with torrent X, for client A and B. A represents a standard user with a BitTorrent client that is switched off periodically and B represents a user with a network attached storage (NAS) in the home which is always on. Also present is a simulated network cache so that when no client is active with X, it is available from the network cache, if it has previously been downloaded by any peer in the network.

Table I
MEASUREMENT META DATA

Network	Start date	End date
1	2011-07-01	2011-10-10
2	2011-04-14	2011-06-23

a client requests X, a check is performed to see if any other client or the network cache is active with X. If not, the X must be downloaded externally, see *e.g.*, client A in the beginning of the example figure. However, if a peer is active with X, this peer is preferred, and we define that there is direct user concurrency between the two respective sessions. Another case is when a peer has been active with (and fully downloaded) X, but has been deactivated. This time, X is found in the network cache, and we thus have network assisted concurrency. Content may be deleted or paused from the "active torrent list" by the user, which means that the client stops uploading that torrent, at least for some period. It is common for torrent clients to automatically stop uploading or being active with a torrent after it has uploaded 150% of it.

This procedure was used to estimate the number of downloads which could could have been kept local by exploiting concurrency. We also translated the results from requests to bandwidth by means of approximate sizes as described in III-C.

IV. RESULTS

A. Traffic measurements

The measurements were performed during 2011, for slightly over two months in network two, and slightly over three months in network one. The exact dates of the measurements are displayed in Table I.

B. General traffic characteristics

To see the role of BitTorrent we analyzed both networks and found very similar results. For reasons of brevity we restrict ourselves to network two in what follows.

During the measurement period 2036 MAC addresses from 1399 households in network 2 generated 200 (320) TB of data in the inbound (outbound) direction. The significance of BitTorrent is clearly demonstrated by the fact that 46 (84) per cent of this was generated by BitTorrent in the inbound

Table II
TRAFFIC MIX IN % OF TRANSMITTED BYTES FOR NETWORK TWO DURING THE MEASUREMENT PERIOD.

Category	% of downlink	% of uplink
Entertainment	0.8	0.1
File Sharing	55.6	93.3
File Transfer	1.7	0.6
Messaging and Collaboration	1.1	0.9
Network Infrastructure	4.7	1.2
Remote Access	1.1	1.3
Streaming Media	24.0	1.9
Web Browsing	10.9	0.8

Table III
AVERAGE DAILY TRAFFIC PER USER (TOTAL MACS FOR THE WHOLE PERIOD), IN GIGABYTES (GB) IN NETWORK TWO.

Traffic type	Inbound	Outbound
Total traffic	1.4	2.3
BitTorrent	0.65	1.9
Encrypted BitTorrent	0.068	0.26

(outbound) directions respectively. File sharing thus made up 56 % of the downlink traffic, and 93 % of the uplink, as can be seen in Table II.

A similar picture is given in Table III which displays the average daily traffic for the measurement period, measured in Gigabytes (GB) per MAC seen during the entire measurement period, in total, for BitTorrent and for encrypted BitTorrent and separated by inbound (to the user) and outbound (to Internet). It is seen that the outbound traffic is dominating (2.3 GB per day compared with 1.4 GB inbound) and that this is entirely because of BitTorrent. We also note that the percentage of encrypted BitTorrent downloads is around 10% of the total BitTorrent downloads and about 14% of the uploads.

The traffic volume produced by an application does not reveal its popularity in number of users. To analyze this further, the number of users per day that used BitTorrent was calculated and compared to the overall number of users and we found that, averaged on a daily basis, 51% of the MAC addresses in network two used BitTorrent. This means that the application is not only dominant in terms of traffic volume, it also has a high penetration among users.

C. Demand patterns and popularity

This section describes the popularity distributions and the concurrency characteristics of the most popular objects. User request characteristics are commonly described by Zipf-like distributions. This is the case in our data as well. The frequency of downloads, averaged over each day of the measurement period, are shown in Figure 3 (left) versus the popularity rank of the objects in a log-log scale. Clearly, the most popular objects are totally dominant in the request characteristics, and the curve resembles a Zipf distribution (which manifests as a straight line in log-log).

The popularity over time is an important factor to characterize, in order to exploit concurrency. The objects in the trace show very different characteristics regarding the number of daily requests (sessions). The number of daily sessions in

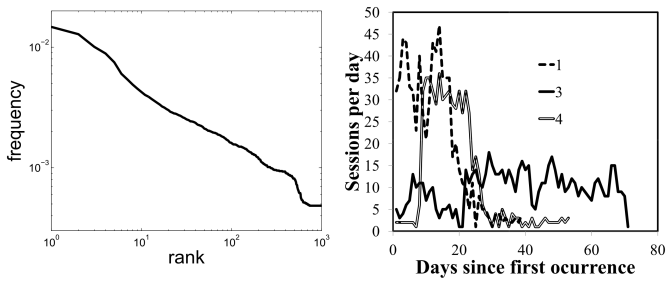

Figure 3. Frequency vs popularity rank for all objects in network 2 (left), averaged over each day of the measurement period. To the right, number sessions per day for three of the four most popular objects in network 2.

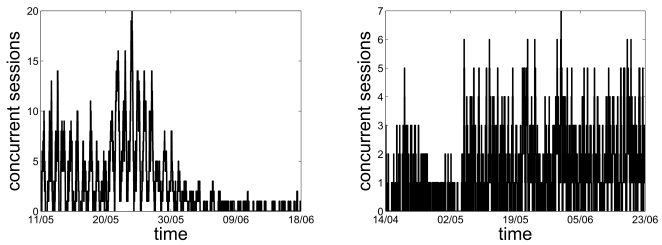

Figure 4. Concurrent sessions over time for objects 1 and 3 in network two.

network 2 for the objects ranked 1, 3 and 4 by popularity respectively are shown in Figure 3 (right). Object 1 exhibits a rapid increase in popularity and then remains popular with a varying demand until it decays after around 25 days. Object 4 has a very low demand at first, then rapidly ramps up, and also decays after roughly 25 days. Completely different characteristics are shown by object 3, which is the only one of these three that is present at the start of the measurement and which is requested at quite a low frequency throughout the whole measurement period. The concurrency patterns of objects 1 and 3 are also quite different, which is illustrated in Figure 4, where the number of concurrent sessions on a per second timescale is shown. Object 1 exhibits a high popularity in the beginning of its lifetime, and fades slowly to a state with only occasional requests. Object 3, however, hardly reaches up to a fourth of the amount of concurrent sessions as compared to object 1, but it maintains a certain level of popularity throughout the measurement period.

D. Concurrency and network assisted concurrency analysis

The post processing of the BitTorrent data revealed that 4821 (1848) distinct IP addresses requested 71612 (27018) distinct BitTorrent hashes in network one (two) respectively. This accounted for 678194 (170564) unique sessions according to the definition of sessions in Section III-D.

We find that for network one, direct user concurrency could handle on average 18% of the requests (20% of the bandwidth, where the bandwidth savings are calculated as described in Section III-C.) Adding a network cache, *i.e.* exploiting network assisted concurrency, 56% of the requests (62% of the bandwidth) could be saved.

Table IV
CONCURRENCY RESULTS IN % OF TOTAL

Network	Direct user		Cache assisted	
	Requests	Bandwidth	Requests	Bandwidth
1	18	20	56	62
2	10	10	66	68

For network two, 10% of the requests (10% of the bandwidth) could be handled by fully exploited direct user concurrency and 66% of the requests (68% in terms of bandwidth) could be handled by adding network assisted concurrency.

The numbers are summarized in Table. IV.

V. CONCLUSIONS

In this paper, content demand patterns have been investigated in order to analyze the potential to keep traffic local within the metro access network. For this purpose, two metrics of concurrency were defined, namely direct user concurrency and network assisted concurrency, together with a method to estimate these metrics in live networks. The results were derived from detailed traffic measurements during more than two months in two metro access networks in Sweden. The application chosen for the analysis was non-encrypted BitTorrent, which has a high penetration (51% of MACs) among end users and generates a large amount (46% of downlink, 83% of uplink) of data. Our results show that a considerable amount of the content demands could be kept local, thereby lowering the load on the aggregation network, the core network and the peering links.

The larger network, network 1, showed a lower percentage (56%) of potential local content downloads (network assisted concurrency) compared to the smaller one, network 2, for we noted 66%. This may sound counter intuitive, but the BitTorrent activity is higher in network 2, and this naturally yields higher possibilities for exploiting locally available content. The number of concurrent requests were 18% and 10%, for the two networks, respectively. This is the fraction of traffic that could have been kept local within the metro access network without adding any network cache, simply by utilizing the active local peers.

Looking closer at the concurrency pattern of individual objects, we showed that there were quite different characteristics among the most popular objects. Some exhibited a pattern of quick rise and steady fall in popularity, but there was also examples of steadily maintained popularity with no evident peak. This indicates that caching decisions are non-trivial, and that this data is useful for future work in on caching algorithms.

ACKNOWLEDGEMENT

This research has received funding from the European Community's Seventh Framework Programme under project 249 025 OASE and EUREKA/CELTIC under project CP07-009 IPNQSIS. Maria Kihl and Andreas Aurelius belong to the Lund Center for Control of Complex Engineering Systems (LCCC). Also, Maria Kihl is a member of the Excellence Center Linköping-Lund in Information Technology (ELLIIT). Andreas Aurelius is partly financed by the Swedish National Strategic Research Area (SRA) within the program TNG (The Next Generation) and the project eWIN.

REFERENCES

[1] J. S. Otto, M. A. Sánchez, D. R. Choffnes, F. E. Bustamante, and G. Siganos, "On blind mice and the elephant: understanding the network impact of a large distributed system," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 110–121, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2043164.2018450

[2] F. Lehrieder, G. Dan, T. Hossfeld, S. Oechsner, and V. Singeorzan, "The impact of caching on bittorrent-like peer-to-peer systems," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, aug. 2010, pp. 1–10.

[3] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak, "Cache replacement policies revisited: the case of P2P traffic," in *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*, April 2004, pp. 182–189.

[4] R. B.-S. N. Leibowitz, A. Bergman and A. Shavit, "Are file swapping networks cacheable? characterizing P2P traffic," 2002.

[5] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?" in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, ser. IMC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251086.1251092

[6] Y. Liu, L. Guo, F. Li, and S. Chen, "A case study of traffic locality in internet P2P live streaming systems," in *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, June 2009, pp. 423–432.

[7] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in bittorrent via biased neighbor selection," *2012 IEEE 32nd International Conference on Distributed Computing Systems*, p. 66, 2006.

[8] L. Sheng and H. Wen, "Reducing cross-network traffic in P2P systems via localized neighbor selection," in *Communications and Networking in China, 2009. ChinaCOM 2009. Fourth International Conference on*, aug. 2009, pp. 1–5.

[9] B. Liu, Y. Cui, Y. Lu, and Y. Xue, "Locality-awareness in bittorrent-like P2P applications," *Multimedia, IEEE Transactions on*, vol. 11, no. 3, pp. 361–371, april 2009.

[10] V. Aggarwal, O. Akonjang, and A. Feldmann, "Improving user and ISP experience through ISP-aided P2P locality," in *INFOCOM Workshops 2008, IEEE*, April 2008, pp. 1–6.

[11] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 363–374. [Online]. Available: http://doi.acm.org/10.1145/1402958.1403000

[12] H. Xie, A. Krishnamurthy, A. Silberschatz, and R. Y. Yang, "P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers."

[13] "Procera networks," http://www.proceranetworks.com.

[14] B. Cohen, "The bittorrent protocol specification," 2008. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html

[15] "Wireshark." [Online]. Available: http://www.wireshark.org/