Peer-to-Peer Collaboration in Content Delivery Networks

Hugo Dias
ISCTE – IUL / IT
Lisbon, Portugal
hugompd@gmail.com

Rui Neto Marinheiro
ISCTE – IUL / IT
Lisbon, Portugal
rui.marinheiro@iscte.pt

Abstract— **A low-cost collaboration architecture for web content distribution, that aims to improve all stakeholder's interests, is presented. A peer-to-peer (P2P) contribution among the end users layer is suggested, in order to increase download rates and reduce server traffic and resource usage. In addition, the Internet Service Providers (ISPs) concerns are also considered, with an ISP-aware connection strategy in the P2P protocol. Collaboration among publisher's web server resources is also proposed, in order to improve the CDN architecture performance. All the elements of this architecture have been developed and have been successfully tested in 5 different scenarios, within the PlanetLab large-scale overlay network testbed. Results show that download speed increases after implementing P2P collaboration on a content delivery scenario, with a strong reduction of data transferred via HTTP servers. The ISP-aware approach reduces inter-ISP traffic, with an increase of download speeds. This implementation is fairer as the content popularity grows because end-users extreme download rates tend to approach to the average.**

Keywords-component; content; delivery; networks; low-cost; p2p; colaboration; cdn; peer; isp-aware;

I. Introduction

With the Internet's continuous growth, content publishers have been facing a considerable scalability problem. Traditionally, web content distribution is mainly based on the client-server model, whose performance degrades while the content popularity growth. In this model, the available resources are only those that belong to server. End users don't have any participation in the distribution process and, therefore, there is no scalability.

Content Delivery Networks (CDN) [10] have been created to reduce web content distribution performance problem. CDNs are computer networks that contain multiple copies of data, placed at different locations to maximize bandwidth usage. By using CDNs, dependency on the publishing servers will be alleviated, providing an increase of content availability and scalability. Despite this, CDNs are still faced with limited resources, as these resources depend on the number of servers available in the network, and might not adapt quickly enough to *Slashdot* effect. Thus, they are not ideal implementations and don't take advantage of users available resources

On the other hand, P2P networks are distributed systems composed of participants that make a portion of their resources available to the other network participants. In comparison to the client-server model, these participants simultaneously behave as clients and servers, making P2P networks much more scalable and potentially more robust than traditional client-server based networks. However, the main problem that some of these networks[1], like Gnutella, face is the excessive bandwidth and resource usage for node and content search. For these there is no centralized entity responsible for the indexing of information. Nevertheless there are a few number of P2P protocols that take advantage of a centralized entity to speed up content and node discovery processes. One of those protocols is BitTorrent[2] where the Tracker plays this role. Content accessibility and fast response time are two of the most important features that end users are looking for on web systems, therefore the BitTorrent protocol (with a centralized Tracker) has a stronger potential, when compared to other P2P protocols, to improve CDNs architecture. In the completely pure P2P protocols (with no centralized entity) mechanisms like Distributed Hash Tables (DHTs) are needed to search for available nodes, causing slower content discovery, mainly regarding lower popularity content.

However, P2P protocols are known for higher bandwidth usage patterns than client-server protocols. Internet Service Providers (ISPs). Concerns about this characteristic have been growing since P2P traffic increases inter-ISP bandwidth usage, and therefore ISP expenses [9].

This paper presents a low-cost content delivery network, which should be available to any publisher that has a web server and is prepared to share their resources with other publishers in order to increase content availability. To reach this achievement an alternative CDN collaboration architecture is proposed. Publishers should share their server's resources among them, thereby, avoiding the use of external networks to distribute their content (and the consequent payment for the use of those resources). In addition, end-users will act as an important piece of the content delivery process, sharing the previously downloaded content, by using the BitTorrent protocol. To take into consideration ISP concerns, it is also proposed that BitTorrent connections should be ISP aware,

[1] Pure P2P protocols: There is no central server managing the Network like in Gnutella protocol.

[2] BitTorrent Protocol: http://www.bittorrent.org/

providing lower inter-ISPs bandwidth usage and better download speed to clients.

The remainder of this paper is organized as following: Section II provides a state of art analysis, underlining some problems and positive aspects found on previous work; Section III and IV present the proposed architecture and a set of results in a real large scale environment through the PlanetLab platform; In Section V conclusions are made about the obtained results and to end and Section VI describes future work, in order to improve and analyze the proposed architecture behavior.

II. State of art

In order to create a more reliable, cheap and scalable CDN different studies have proposed amendments that often present solutions based in P2P protocols. [1, 2, 3] compare the different types of P2P protocols and concluded that, despite their lower robustness, the hybrid protocols (with a centralized entity) are more reliable for a CDN implementation. Those studies stated that pure P2P protocols have slower response time when considering node and content discover in the network. They are not regarded as good solutions for a HTTP-like scenario. In addition, pure P2P can't assure the availability of content, because they are fully dependent on users. Thus those studies suggest implementations of hybrid P2P protocols over the edge of CDN, keeping the core advantage of client-server protocols: easy content localization. These studies only suggest changes at the edge of CDN, which could be a good idea for a pre-existent network but not for a new one. None of these proposals have suggested the possibility of taking advantage of resource sharing among the various content publishers, an aim on which the architecture proposed in this paper is focused on. The available research studies haven't made tests in a real large-scale scenario, and all of them have only presented theoretical results based on deterministic calculations.

Considering commercial CDNs, [4] says that *Akamai*[3] has already acquired P2P *RedSwoosh*, which have been gradually implemented. Other companies like *VeriSign*[4], *CacheLogic*[5], *InterNap*[6] and *Joost*[7] have already announced their P2P based CDNs, however theirs detailed architecture is not known in depth. *BitTorrent Inc* has also announced its own free application to provide a BitTorrent based collaboration over the edge of any traditional CDN [5]. Unfortunately, like the studies [1,2,3], their solution is very focused on the CDN edges and should mainly be taken in consideration while implementing a P2P collaboration over a pre-existent CDN.

Studies [6, 7, 8] have analyzed P2P impact over the ISP networks and all concluded that this impact could be reduced if the connection algorithms take into more priority ISP nodes. By applying an ISP-aware connection algorithm it's possible to reduce latency and, therefore, increase download speed. From the ISP point of view, the connections will stay inside their own network, and so inter-ISP bandwidth usage will be reduced, consequently decreasing expenses. [7] suggests context aware BitTorrent Trackers, allowing administrators to select different peer selection strategies and, therefore, different ways of selecting peer nodes. [8] goes further and indicates changes on both the BitTorrent Tracker and the client, in order to guarantee that each peer list is composed by a much larger number of internal ISP peers. Finally, [6] has only focused on the BitTorrent client, giving more priority to peers in same ISP on the peer selection algorithms (unchoke). This implementation has been tested with the PlanetLab platform, like the one presented in this paper.

The main differences of our proposal in relation to the state of the art is that this one proposes a CDN architecture, where collaboration and P2P interaction is on the core and on the end-user layers, and it doesn't consider those aspects as a mere extension. Content publishers share their own servers resources and therefore don't need to use a network created solely for that. This implementation has also been tested on a real large-scale scenario through the PlanetLab unlike the most of research presented in this chapter.

III. Architecture

In order to provide a low-cost scalable CDN, the given architecture, as is clarified in Figure 1, is divided into two layers:

- Content Publishers (Core): This layer represents content publisher's servers that will collaborate in order to increase content availability. Therefore, each server will replicate external content and share his own resources in order to distribute this content. To provide faster P2P node selection, the servers will also act as a BitTorrent Trackers, giving the users the possibility to connect and to obtain a peer list for a specific content;
- End users (Edge): This layer represents anyone who is interested in downloading contents from the top layer. While downloading the contents, users may provide data to others through the BitTorrent Protocol. Thus the data will be downloaded simultaneously by the users from HTTP server and the BitTorrent network. In order to decrease ISP expenses, the BitTorrent connections are ISP-aware, reducing the inter-ISP traffic (detailed explanation on "Tests and Results" chapter).

[3] Akamai: http://www.akamai.com/

[4] Verisign: http://www.verisign.com/

[5] CacheLogic: http://www.cachelogic.com/

[6] InterNap: http://www.internap.com

[7] Joost: http://www.joost.com/

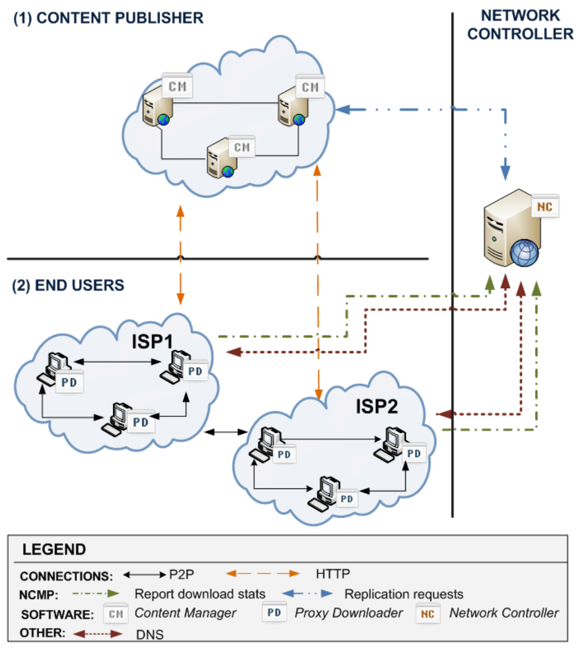
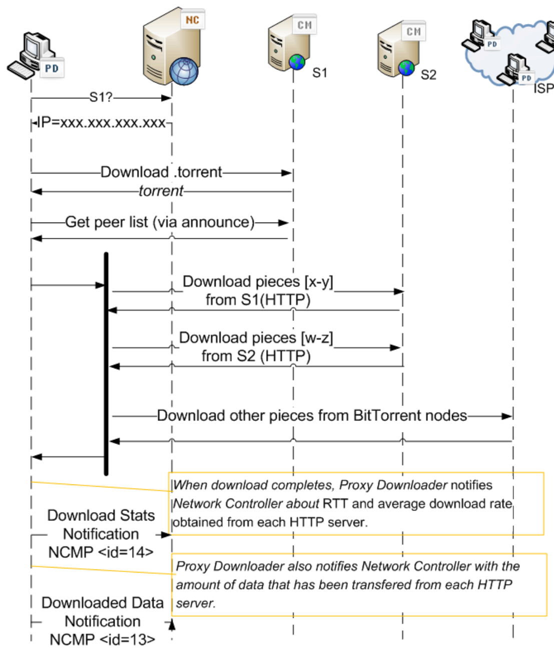


Figure 1. Proposed Architecture



PD
NC
CM
S1
S2
ISP 1
S1?
IP=xxx.xxx.xxx.xxx
Download .torrent
torrent
Get peer list (via announce)
Download pieces [x-y] from S1(HTTP)
Download pieces [w-z] from S2 (HTTP)
Download other pieces from BitTorrent nodes
When download completes, Proxy Downloader notifies Network Controller about RTT and average download rate obtained from each HTTP server.
Download Stats Notification NCMP <id=14>
Proxy Downloader also notifies Network Controller with the amount of data that has been transfered from each HTTP server.
Downloaded Data Notification NCMP <id=13>

Figure 2. Download using Proxy Downloader

Due to the distributed nature of this architecture, the load balancing mechanism will be implemented using a dynamic DNS protocol. The DNS protocol doesn't allow a content-aware load balancing, so the content must be divided in blocks that will be mapped into different domains. To orchestrate this feature a centralized entity was created, which is called Network Controller (NC). This entity is also responsible for replica placement management, based on statistical data received from users. Each user periodically sends information about download stats, like: download speed; round trip time; download bytes and server availability.

Content download using the Proxy Downloader (PD) is resumed in Figure 2. As is possible to see, download will simultaneously proceed from different sources (e.g. HTTP servers and BitTorrent nodes). This is only possible because PD implements BitTorrent HTTP Seeding[8] extension, which upgrades any BitTorrent client to download content from HTTP based sources.

When download completes, the PD should notify the NC, providing information like download speed, round trip time and the amount of data obtained from each server. With this information the NC can make fair decisions in content replication and replica placement algorithms.

[8] BitTorrent WebSeeding Extension: http://www.bittorrent.org/beps/bep_0017.html

IV. TESTS AND RESULTS

For each network architecture entity (servers, users and NC) an application was developed. Transparency was central in this development, and the use of standard browser and server applications was crucial in a way that network completely transparent. For the end users a Proxy Downloader (PD) was used as a proxy to content access and communication with other network element. On the other hand, for the server application a Content Manager (CM) was developed that will automatically answer to NC requests, including the replication of any target content. Communication between PD, CM, and NC applications was implemented using a proposed protocol: Network Controller Message Protocol

A set of tests has been made over the PlanetLab platform, in order to analyze the impact of P2P protocol in this content delivery context. All the tests have been repeated 5 times with 75 PlanetLab different nodes. These tests have used servers with a 10Mb connection, regarding the top layer of the proposed architecture. The default content size is 86 Mbytes, except in the 3rd scenario where the content size impact on this CDN has been tested.

A. 1st Scenario: BitTorrent Impact

In the first scenario, all content requests have been made simultaneously, first by using a direct HTTP request and then by using the PD, which uses an hybrid HTTP and BitTorrent approach to requests. The PD prototype was programmed to download from HTTP only when few parts of data aren't available through BitTorrent and when the download is starting, in order to reduce the BitTorrent slow-start problem.

Has it's possible to see in Figure 3, the average download rate in our proposed architecture is superior to the one obtained while downloading directly from HTTP. With the BitTorrent inclusion, the download rate was increased nearly by 666%.

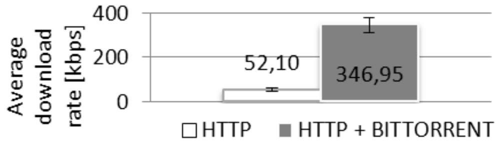


Figure 2. Download rate with and without BitTorrent

Also the generated traffic by the Web Server has been reduced by 82.63%, proving that this type of implementation is highly beneficial for the users and the content publishers.

B. 2nd Scenario: Content Popularity

In this scenario the main objective is to obtain the architecture response for different types of content popularity. To achieve this, a ZIPF equation was used:

$$f(k; s, N) = \frac{\frac{1}{k^s}}{\sum_{n=1}^{N}(\frac{1}{n^s})} \quad (1)$$

This function returns a number between 0 and 1 which represents the percentage of elements 'k' in a set of 'N' elements for a specified 's' curve. In this scenario, the 's' represents content popularity distribution: as higher the value, the higher is the content popularity. For this purpose, 'k' can be seen has the time element that represents the minutes elapsed since the simulation has started.

This equation was not prepared for this kind of use, so it was modified to obtain an integer number and therefore return the correct number of requests to make in each instant of time. As we have a percentage result, we can get the total number of requests by multiplying ZIPF value with the total number of requests (M=75):

$$\mathrm{m}\ (\mathrm{k}; \mathrm{s}, \mathrm{N}, \mathrm{M}) = \mathrm{f}(k; s, N) \times \mathrm{M} \quad (2)$$

where M is the number of requests to execute.

Now we have to transform ZIPF theoretical behavior into a practical one:

$$m_s(k; s, N, M) = INT(m_{s_{aux}}(k; s, N, M)) \quad (3)$$

$$m_{s_{aux}}(k; s, N, M) = \begin{cases} m(k; s, N, M) + DEC\left(m_{s_{aux}}(k-1; s, N, M)\right); k > 0 \\ m(k; s, N, M) \qquad ; k = 0 \end{cases} \quad (4)$$

Where m_s is the number of requests to execute in a specific instant of time (k), INT and DEC the functions that return, respectively, the integer and the decimal part of a number.

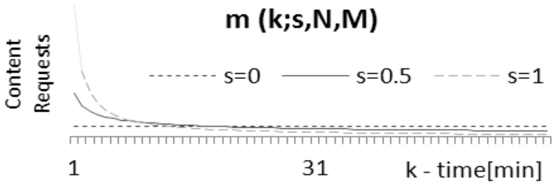


Figure 3. Theoretical curve of Zipf equation

As shown in the Figure 4, we made three different tests: with high popularity (s=1), constant behavior (s=0) and with an intermediate popularity (s=0.5).

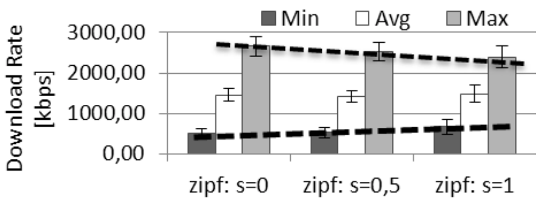


Figure 4. Download rate

Results show that content popularity has no big impact on the download rate provided by this network as the average download speed is similar for all of tested scenarios (Figure 5). We can also see that this implementation tends to be fairer as the content popularity grows, because minimum and maximum values approach themselves to be closer to the average one. This demonstrates that this architecture is robust and scalable regarding content popularity.

Analyzing the amount of data transferred directly from server (Figure 6), we can see that it tends to reduce as the content popularity grows. This proves that we have a scalable solution, saving more network resources (bandwidth) for contents with higher popularity.

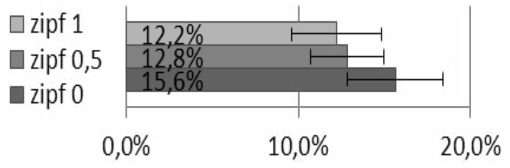


Figure 6 - Data obtained through HTTP [%]

C. 3rd Scenario: Content Size

The 3rd scenario aims to check this implementation behavior using files with diverse sizes. As it is possible to see in Figure 7 and Figure 8, the results show that this implementation is more effective for large files. This could be a limitation, or not, depending on the proposed CDN purpose.

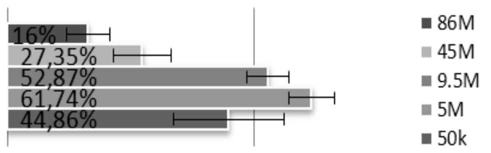


Figure 7. Data Downloaded from HTTP [%]

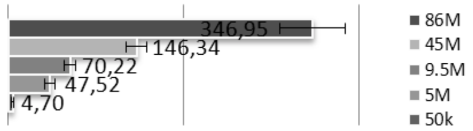


Figure 8. Avg. Download Rate [kBps]

This implementation shows different behaviors while using different files, as observed in Figure 7 and Figure 8. The BitTorrent protocol has a slow start limitation when compared to HTTP protocol, which is more perceptible for as the file size decreases (Figure 8). All tested files, which have a file size below 9,5Mbytes have a lower download rate when compared with the generic HTTP request (52,10Kbps). Depending on the usage scenario, this might be seen as a limitation. P2P collaboration should only be used for larger files otherwise the CDN performance may be deteriorated.

It is also possible to see that the amount of data obtained through the HTTP web server tends to decrease as file size increase, except in the first case where the file is smaller than the default BitTorrent length (256kbyts). For this one the content has only one piece, and due to the Proxy Downloader prototype configuration, it has an equal probability of being downloaded either from the BitTorrent or the HTTP server.

D. 4th Scenario: ISP-aware connections

The final scenario proves that users and ISPs can both benefit from implementing an ISP-aware connection model in the BitTorrent protocol. On this section the PlanetLab nodes have been divided in domain groups, so each domain represents a different ISP (similar to [6]). This grouping is shown it the Table1:

TABLE I. PLANETLAB ISP GROUPING

Domain	edu	jp	net	tw	ch	at	kr	org	other
Num. servers	40	5	3	3	2	2	2	2	16

To implement this model, the tracker has been modified in order to provide a list composed with 80% of peers from the same ISP as the requestor and 20% from others (limited to 30 peers). If this procedure can't return at least 24 peers from the same ISP then the missing ones will be filled with peers from other ISPs. The BitTorrent client has also been modified, so peers from the same ISP have priority in the unchoking algorithm.

Has shown in Figure 9, the data obtained through the same ISP has been increased by 48%, promoting an expense reduction to ISPs (inter-ISP traffic).

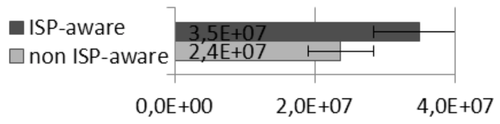


Figure 9. Data obtained through the same ISP [bytes]

Also the average download rate has been increased by 3,8% (1- 346.95/360.75) improving the performance for end users (Figure 11).

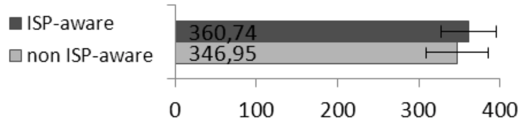


Figure 10. Avg. download rate [kBps]

From the content publishers point of view there is a slight degradation of the performance since there is an increase on the percentage of data obtained through the server (1.2% as shown in Figure 11). This value has incremented because users are restricted to a specific group of connection and, therefore, BitTorrent pieces availability will be reduced. However, this deterioration doesn't have a big impact if we take into consideration that serves traffic has been reduced by 82.63% after implementing BitTorrent protocol.

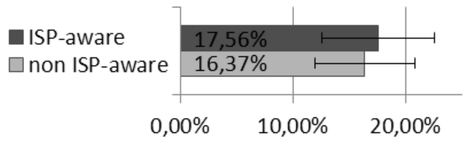


Figure 11. Data obtained from the web server [%]

E. 5th Scenario: Replica placement and server selection

This proposal explores specific algorithms to select servers and blocks for replication. Unlike traditional CDNs, this proposal has statistical information provided by end-users, which can be used as input of those algorithms in order to improve performance. In addition, due to the different nature of this network core, where content publishers share their own servers, different algorithms have to be used. We have made preliminary tests with three algorithms aiming the optimization of: upload speed; free-space and share-ratio. For this propose 4 servers (CM) were used, with 4 different content sizes: 5, 9.5, 30 and 86 Mbytes. In order to test those algorithms random requests were made to each content, following a zipf s=0 distribution (see paragraph 2nd scenario).

The share ratio algorithm is measured as the ratio between the amount of external contents replicated in one server and the amount of data from that server that is replicated along the network. Preliminary results shown in Figure 12 lead us to conclude that the share ratio based algorithm is more reliable in terms of collaboration fairness because is the one nearest to the ideal value, which is when the server replicates the same amount of external contents as the amount of this server contents that is replicated along the remainder CDN servers. This value indicates that the server gets from the network the same that it shares. In the other algorithms the average values are higher, and also the confidence interval is larger. This indicates that those algorithms cause a lower fairness in the network, because servers share ratios are more dissimilar.

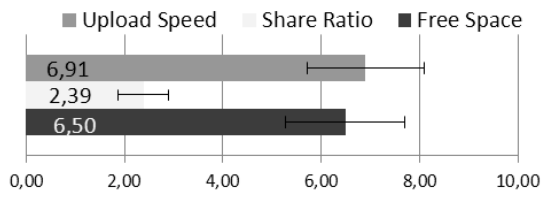


Figure 12. Share Ratio among different algorithms

The overall analysis to these preliminary results allow us to understand that those three algorithms should work together as one, because each one is better in optimizing one particular aspect. This is true in the above figure, where share ration optimization is the goal. Nevertheless the combination of these algorithms depends very much on the optimization that is aimed, and on the meaning of fairness in the share of publisher resources.

V. CONCLUSIONS

The proposed architecture reliability has been successfully proved either from the user and the content publishers point of view. After implementing BitTorrent collaboration in the user layer, it was possible to reduce server traffic and increase download speed significantly. Also this implementation has behaved accurately while testing different content popularities, presenting lower server usage, as content popularity tends to decrease. Results show that this CDN is fairer as the content popularity increases because minimum and maximum download rates approaches themselves to the average value as popularity grows.

This kind of implementations is not suitable for the distribution of small files because of the slow start limitation of BitTottent. Files with a size lower then 10MB tend to have a worst performance compared to a direct download from the web server (without BitTorrent collaboration).

Finally, an ISP-aware connection model improves user download rate as well as reduces inter-ISP and consequently reducing ISP expenses. The drawback is that the HTTP traffic is increased by 1.2% which is not a big impact has this traffic has been reduced 82% with the BitTorrent implementation.

VI. FUTURE WORK

Taking advantage of the prototypes created during this work we intend to make tests with different replication and server selection algorithms, trying to understand their pros and constrains. With this study it will be possible to find out how and when specific algorithms should be combined in order to improve CDN performance.

REFERENCES

[1] Pakkala, Daniel and Latvakoski, Juhani. Towards a Peer-to-Peer Extended Content Delivery Network. 14th IST Mobile & Wireless Communications Summit. Jun.2005

[2] Huang, Cheng, et al. Understanding Hybrid CDN-P2P: Why Limelight Needs its Own Red Swoosh. International Workshop on Network and Operating System Support for Digital Audio and Video. 2008.

[3] Jiang, Hain, et al. Efficient Hierarchical Content Distribution using P2P Technology. 16th IEEE International Conference on Networks. 2008 Dec.

[4] Jiang, Hai, et al. Efficient Large-scale Content Distribution with Combination of CDN and P2P Networks. Internation Journal of Hybrid Information Rechnology. 2, 2009 Vol. 2.

[5] BitTorrent DNA.BitTorrent's Delivery Network Accelerator (DNA) Service Improves the Online Experience for Streaming Video, Downloadable Software and Video Games. [Online] http://www.bittorrent.com/pressreleases/2007/10/09/bittorrent%E2%80%99s-delivery-network-accelerator-dna-service-improves-the-online-e.

[6] Lin, Minghong, Lui, John C.S. and Chiu, Dah-Ming. Design and Analysis of ISP-friendly File Distribution Protocols. 46TH Annual Allerton Conference on Communication, Control and Computing. 2008, Vol. III.

[7] Sousa, Pedro. Context Aware Programmable Trackers for the Next Generation Internet. The Internet of the Future. s.l. : Springer Berlin / Heidelberg, 2009, Vol. Volume 5733/2009, 78-87.

[8] Bindal, Ruchir, et al. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. 26th IEEE International Conference on Distributed Computing Systems. 2006.

[9] Karagiannis, Thomas, Rodriguez, Pablo and Papagiannaki, Konstantina. Should Internet Service Provider Fear Peer-Assited Content Distribution. 5th ACM SIGCOMM conference on Internet Measurement. 2005.

[10] Hofmann, Markus e Baumont, Leland R.Content Networking. s.l. : Morgan Kaufmann, 2005.