Design and Implementation of a BitTorrent Tracker Overlay for Swarm Unification

Călin-Andrei Burloiu, Răzvan Deaconescu, Nicolae Țăpuș
Automatic Control and Computers Faculty
Politehnica University of Bucharest
Emails: calin.burloiu@cti.pub.ro,{razvan.deaconescu,nicolae.tapus}@cs.pub.ro

***Abstract*—The deployment of the BitTorrent protocol in the early 2000s has meant a significant shift in Peer-to-Peer technologies. Currently the most heavily used protocol in the Internet core, the BitTorrent protocol has sparked numerous implementations, commercial entities and research interest. In this paper, we present a mechanism that allows integration of disparate swarms that share the same content. We've designed and implemented a novel inter-tracker protocol, dubbed TSUP, that allows trackers to share peer information, distribute it to clients and enable swarm unification. The protocol forms the basis for putting together small swarms into large, healthy ones with reduced performance overhead. Our work achieves its goals to increase the number of peers in a swarm and proves that the TSUP incurred overhead is minimal.**

***Index Terms*—Peer-to-Peer, BitTorrent, tracker, unification, TSUP, swarm**

I. Introduction

The continuous development of the Internet and the increase of bandwidth capacity to end-users have ensured the context for domination of content-distribution protocols in the Internet. Currently, most Internet traffic is content Peer-to-Peer traffic, mostly BitTorrent. Peer-to-Peer protocols have positioned themselves as the main class of protocols with respect to bandwidth usage.

The arrival of the BitTorrent protocol in the early 2000s has marked a burst of interest and usage in P2P protocols, with the BitTorrent protocol currently being accounted for the biggest chunk in Internet traffic [3]. Modern implementations, various features, focused research and commercial entities have been added to the protocol's environment.

In this paper, we address the issue of unifying separate swarms that take part in a session sharing the same file. We propose a tracker unification protocol that enables disparate swarms, using different .torrent files, to converge. We define swarm unification as enabling clients from different swarms to communicate with each other. The basis for the unification is a "tracker-centric convergence protocol" through which trackers form an overlay network send peer information to each other.

A. BitTorrent Keywords

A *peer* is the basic unit of action in any P2P system, and, as such, a BitTorrent system. It is typically a computer system or program that is actively participating in sharing a given file in a P2P network. A peer is generally characterized by its implementation, download/upload bandwidth capacity (or limitation), download/upload speed, number of download/upload slots, geolocation and general behavior (aggressiveness, entry–exit time interval, churn rate).

The context in which a BitTorrent content distribution session takes place is defined by a BitTorrent **swarm** which is the peer ensemble that participate in sharing a given file. It is characterized by the number of peers, number of seeders, file size, peers' upload/download speed. swarm, one that allows rapid content distribution to peers, is generally defined by a good proportion of seeders and stable peers. We define **stable peers** as peers that are part of the swarm for prolonged periods of time.

Communication of peers in a swarm is typically mediated by a BitTorrent **tracker** or several trackers which are defined in the .torrent file. It is periodically contacted by the peers to provide information regarding piece distribution within the swarm. A peer would receive a list of peers from the tracker and then connect to these peers in a decentralized manner. As it uses a centralized tracker, the swarm may suffer if the tracker is inaccessible or crashes. Several solutions have been devised, such as PEX (Peer EXchange) [1] or DHT (Distributed Hash Table) [11]. The tracker swarm unification protocol presented in this article enables redundancy by integrating multiple trackers in a single swarm.

B. Tracker Swarm Unification Protocol

The goal of the tracker swarm unification protocol is the integration of peers taking part in different swarms that share the same file. These swarms, named **common swarms**, use the same content but different trackers.

We have designed and implemented a tracker network overlay that enables trackers to share information and integrate peers in their respective swarms. The overlay is based on the Tracker Swarm Unification Protocol (TSUP) that allows update notification and delivery to trackers from the overlay. The protocol design is inspired by routing protocols in computer networks.

At this point, as proof of concept, the tracker overlay is defined statically. All trackers know beforehand the host/IP addresses and port of the neighboring trackers and contact them to receive required information. The integration of dynamic tracker discovery is set as future work. Each tracker may act as a "router", sending updates to neighboring trackers that may themselves send them to other trackers.

 ISBN: 978-1-61208-133-5

II. Tracker Unification

A. Motivation

It is also possible that different users create different .torrent files, but with the same files for sharing. If the .torrent files don't refer the same tracker, each one will represent another swarm. The peers from different swarms do not know each other and integration is not accomplished.

By unifying swarms the communication between peers is possible. Every client will have the opportunity of increased connections to other peers, increasing the download speed and decreasing the download time. By having more peers in the swarm, the number of stable seeders also increases and the client can approach its maximum potential.

B. Solution

The protocol proposed in this article, named **Tracker Swarm Unification Protocol (TSUP)**, renders possible the unification of swarms which share the same files, by employing a tracker network overlay. A tracker which implements this protocol will be referred here as an **unified tracker**.

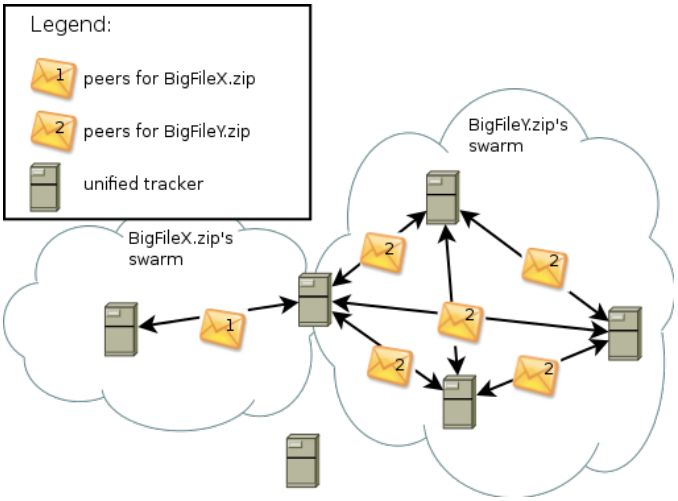

Figure 1. Unified trackers

Torrent files created for the same content have the same *info_hash*. So in swarms that share the same file(s) (common swarms), peers will announce to the tracker the same *info_hash*. Therefore, TSUP capable trackers can "unify" them by communicating with each other in order to change information about peers which contribute to shared files with the same *info_hash*. In order to accomplish this, unified trackers send periodic updates to each other, containing information about the peers from the network.

III. TSUP

As mentioned above, *TSUP* is the acronym for *Tracker Swarm Unification Protocol*. TSUP is responsible with the communication between trackers for peer exchange information in common swarms.

A. Protocol Overview

For transport layer communication, the protocol uses UDP (User Datagram Protocol) to reduce resource consumption. A tracker already possesses a lot of TCP (Transport Control Protocol) connections with each other peers. Adding more TCP connections to each neighboring tracker would increase TSUP overhead too much for a big tracker overlay. The messages passed from one tracker to another do not need TCP's flow control and need a lower level of reliability than TCP as it will be explained below. The simplicity of the UDP protocol gives the advantage of a smaller communication overhead.

In TSUP's operation the following processes may be identified:

- **Virtual connections establishment process:** A three-way-handshake responsible with establishing a UDP "connection" between two linked trackers at the application layer. The process is started by a SYN packet (synchronization packet).
- **Unification process:** The trackers exchange unification packets (named *SUMMARY* packets) during a three-way-handshake in order to find out which are the common swarms.
- **Updating process:** The trackers exchange peers from common swarms, encapsulated in *UPDATE* packets.
- **Election process:** The establishment of a *swarm leader* which is responsible with receiving all updates from the neighboring linked trackers, aggregating them and sending the results back.

The *unification process* includes an updating process in its three-way-handshake, such that the two operations, unification and update, are run in pipeline. Whenever a new torrent file is registered to the tracker, a new swarm is created. The unification process is triggered and a SUMMARY packet is immediately sent to each neighboring tracker, informing the others of the new swarm.

The *updating process* is started periodically, such that UPDATE packets are sent at a configurable interval of time to each tracker in a common swarm. A typical update interval is 30 seconds.

In order to maintain the virtual connections between trackers, HELLO packets are sent periodically, acting as a keep-alive mechanism. A typical HELLO interval is 10 seconds, but its value may be changed from protocol configuration. If no HELLO packet is received during a configurable interval, called disconnect interval, the virtual connection is dropped and the virtual connection establishment process is restarted for that link by sending a SYN packet.

Some packets, such as UPDATE packets, must be acknowledged. If no answer or acknowledgement is received, the packet is retransmitted. For example, UPDATE packets are resent at each hello interval until an acknowledgement is received.

It is not a problem if some UPDATE packets are lost and arrive later to destination. However they need to be

 ISBN: 978-1-61208-133-5

acknowledged and they are retransmitted in order to increase the probability of their arrival. TCP, by offering reliability, provides a faster delivery of updates in case of a network failure which is not needed in the case of TSUP. Lower overhead is considered here more important that fast retransmission. Thus TSUP implements a timer-driver retransmission, as opposed to data-driven used by TCP [10].

Periodically sent packets, the keep-alive mechanism, acknowledgements and retransmissions contribute to the low reliability needed in TSUP. They help exceed the drawbacks of the UDP transport protocol, and also give a more efficient communication than a TCP one.

B. Tracker Awareness

Tracker communication is conditioned by awareness. For this purpose, in the current version of the protocol, each tracker is configured statically with a list of other communicating trackers. Each element of the list represents a *link* which is identified by the tracker host name (URL or IP address) and port. Other parameters for the link may be configured; some of them may be specific to the implementation. If the virtual connection establishment process is successful, the link becomes a virtual connection, which is conserved with keep-alive packets (HELLO packets).

A future version of the protocol will incorporate the design of a tracker discovery mechanism capable of generating the list of communicating trackers for each tracker dynamically, with the benefit of scalability and reduction of the administrative overhead.

C. Tracker Networks

To improve TSUP's scalability, trackers may be grouped together in networks named **tracker networks**. Connections in all tracker networks are full mesh. Two networks are connected with the aid of **border trackers** (see Figure 2).

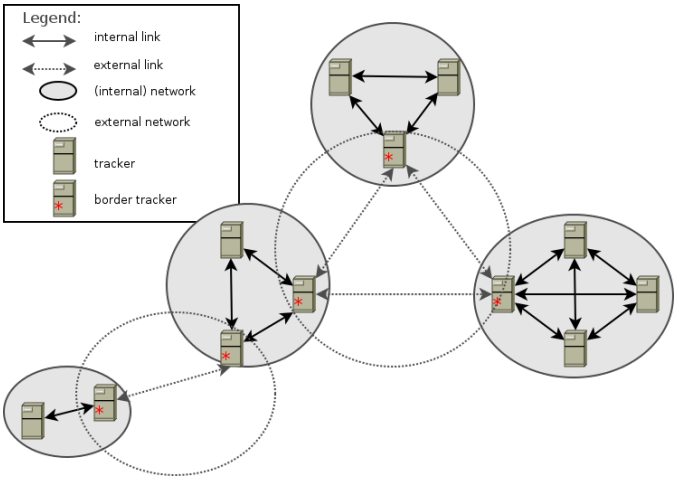

Figure 2. Tracker Networks

To configure a topology which contains multiple networks, each link of each tracker must be set as an **internal link** or an **external link** (see Figure 2). Trackers connected with internal links are part of the same *tracker network*; trackers connected with external links are part of other networks. However, the ones from the first category may also be classified as belonging to an **internal network** and the ones from the second category as being from an **external network**. A complete graph, using internal links as edges is an *internal network*, and a complete graph with external links as edges is an *external network* (see Figure 2). A tracker which has both internal and external links is a border tracker. Peer information received from an internal network originates from **internal peers** while information received from an external network originates from **external peers**. Peers connected to a tracker with TCP, via the BitTorrent protocol, are called **own peers**. The links between trackers in a network, whether internal or external, must be full mesh.

In order to use a scalable and low resource consuming communication within a network, trackers are organized in groups depending on the unified swarm. Therefore a tracker may belong to more than one group at the same time, the number of groups it belongs being equal with the number of swarms present on that tracker. Each group contains a **swarm leader** responsible for sending peer updates to peers in the group. The other group members, instead of sending updates to other members on a full mesh graph, it sends updates to the swarm leader on a tree graph, reducing updating overhead. These updates propagate to other peers in the group.

In accordance to graph theory, the number of updates sent in a swarm without the swarm leader mechanism (full mesh) is computed by using the formula below:

$$UPDATES_{fullmesh} = 2 \cdot \frac{n(n-1)}{2} = n(n-1) \quad (1)$$

The number of updates sent within a swarm using the swarm leader mechanism (tree) are:

$$UPDATES_{swarmleader} = 2(n-1) \quad (2)$$

As may be observed from the formulas above the complexity decreases from $O(n^2)$ in a full mesh update scheme to $O(n)$ with the swarm leader scheme.

Each swarm contains two swarm leaders, one for the internal network (which sends updates through internal links), called *internal swarm leader* and one for the external network (which communicates updates through external links), named *external swarm leader*.

As connections are full mesh in an internal network, the internal peers (received from other trackers from the internal network) are distributed to other internal trackers only by the internal swarm leader and in no other circumstance by another tracker. Through analogy, in an external network, peers (received from other trackers from the external network) are distributed to other external trackers only through the external swarm leader. On the other hand, internal peers are distributed to external trackers and external peers to the internal trackers.

 ISBN: 978-1-61208-133-5

Own tracker peers are distributed both to the internal and the external network.

Swarm leaders are automatically chosen by trackers during the election process which is started periodically. There are metrics used in order to choose the most appropriate leader. The first and most important one prefers as swarm leader a tracker which possesses the smallest *number of swarm leader mandates*. The *number of mandates* is the number of swarms where a tracker is swarm leader. This balances the load of the trackers – as the number of mandates of a tracker increases, its load also increases. In the current version of the protocol the grouping of trackers into networks and the selection of border trackers is done manually (statically) by the system administrator.

When two network trackers A_1 and A_2 are connected indirectly through other network trackers B_j, if A_1 and A_2 use a common swarm and B_j doesn't use this swarm, then the A_i trackers cannot unify unless the border trackers are specified in the configuration. This happens because the configured border trackers must unify with any swarm, although they do not have peers connected from that swarm.

Grouping trackers in networks increases system scalability, but also network convergence time. The update timers can be set to a lower value for border trackers to limit convergence overhead. The system administrator should opt between scalability and convergence and adapt the protocol parameters to the specific topology.

IV. Implementation Details

TSUP is currently implemented in the popular *XBT Tracker* [9], implemented in C++. The extended TSUP capable tracker was dubbed *XBT Unified Tracker*.

The original tracker implements an experimental UDP BitTorrent protocol known as *UDP Tracker*. Because TSUP also uses UDP and communication takes place using the same port, TSUP-specific packets use the same header structure as the UDP Tracker, enabling compatibility.

XBT Tracker uses a MySQL database [8] for configuration parameters [7] and for communication with a potential front end. XBT Unified Tracker adds parameters for configurations that are specific for TSUP and uses a new table in order to remember links with other trackers and their parameters. *Tracker awareness*, as described in III-B, is implemented in the database.

Besides the HTTP *announce* and *scrape* URLs, the original tracker uses other web pages for information and debugging purpose. The unified tracker adds two extra information web pages for *monitoring*. The *trackers web page* shows details about every link and the state of the connection for that link. For every swarm, the *swarms web page* shows the list of peers and the list of trackers connected for that swarm.

V. Experimental Setup

TSUP testing activities used a virtualized infrastructure and a Peer-to-Peer testing framework running on top of it. We were able to deploy scenarios employing various swarms, ranging from a 4-peer and 1-tracker swarm and a 48-peer and 12-tracker swarm. Apart from testing and evalution, the infrastructure has been used to compare the proposed tracker overlay network with classical swarms using a single tracker and the same number of peers. We will show that a unified swarm has similar performance when compared to a single tracker (classical) swarm.

In order to deploy a large number of peers we have used a thin virtualization layer employing OpenVZ [5]. OpenVZ is a lightweight solution that allows rapid creation of virtual machines (also called containers). All systems are identical with respect to hardware and software components. The deployed experiments used a single OpenVZ container either for each tracker or peer taking part in a swarm. A virtualized network has been build allowing direct link layer access between systems – all systems are part of the same network; this allows easy configuration and interraction.

The experiments made use of an updated version of hrktorrent [2], a lightweight application built on top of libtorrent-rasterbar [4]. Previous experiments [13] have shown libtorrent-rasterbar outperforming other BitTorrent experiments leading to its usage in the current experiments. The experiments we conducted used a limitation typical to ADSL2+ connections (24 Mbit download speed limitation, 3 Mbit upload speed limitation).

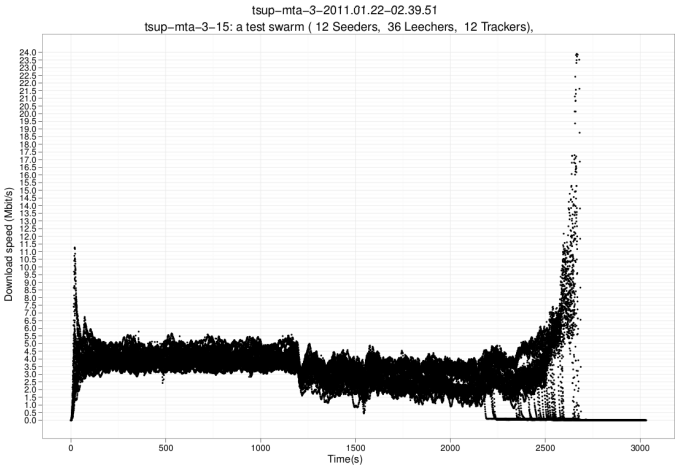

Figure 3. Sample Run Graphic

An automatically-generated sample output graphic, describing a 48 peer session (12 seeders, 36 leechers, 12 trackers) sharing a 1024 MB file is shown in Figure 3. The image presents download speed evolution for all swarm peers. All of them are limited to 24 Mbit download speed and 3 Mbit upload speed.

All peers use download speed between 2 Mbit and 5 Mbit on the first 2000 seconds of the swarm's lifetime. As the leechers become seeders, the swarm download speed increases rapidly as seen in the last part of the swarm's lifetime, with the last leechers reaching the top speed of 24 Mbit.

 ISBN: 978-1-61208-133-5

VI. Scenarios and Results

In order to test the overhead added by TSUP to BitTorrent protocol, we have made a set of test scenarios which compare the average download speed for a swarm with unified trackers and for another swarm with just one non-unified tracker, but the same number of leechers and seeders. Each test scenario is characterized by the shared file sizes, the number of peers and, in the case of tests with unified trackers, by the number of trackers. We shared 3 files of sizes 64MB, 256MB and 1024MB. In the test scenarios with unified trackers for each file we tested the swarm with 1, 2, 4, 8 and 12 trackers. On each tracker there were connected 4 peers, from which 1 is a seeder and 3 are leechers. So, for example, in a scenario with 8 trackers there are 8 seeders and 24 leechers, totalizing 32 peers. Having 3 files and 12 trackers in the biggest scenario we needed to create 36 .torrent files, because for each shared file we made a .torrent file for each tracker. In the corresponding test scenarios with non-unified trackers, there is just one .torrent file for each shared file. We varied the numbers of seeders and leechers connected to the tracker so that they have the same cardinality with the corresponding unified trackers scenarios.

Each test scenario has been repeated *20 times* in order to allow statistical relevance. The average download speed was calculated as an average value from the 20 sessions.

Results may be seen in the table from Figure 4, which depicts the results for each file size, in the top (64MB), middle (256MB) and bottom part of it (1024MB), respectively. For each of this two situations the mean download speed ("mean dspeed") and relative standard deviation ("rel.st.dev.") is depicted. In the right part, titled "perf. Decrease" (performance decrease), shows the percent of download speed decrease induced by the overhead of the TSUP. In the left side of the table the number of seeders and leechers for each scenario is shown. The percentage value for download speed decrease is computed using the standard formula:

$$d = 100\% \cdot \frac{ds_{SingleTracker} - ds_{UnifiedTrackers}}{ds_{SingleTracker}}, \quad (3)$$

where *ds* is an abbreviation from mean download speed.

All positive percentage values from the "perf. decrease" header mark a decrease of performance caused by TSUP overhead. A negative download speed decrease percentage shows that there is an increase instead of a decrease.

The unification which takes place in XBT Unified Tracker introduces an overhead in the BitTorrent protocol in comparison to XBT Tracker. In theory the performance decrease must always be positive. But there are situations where the percentages are negative, which could suggest that TSUP increases the speed, thus reducing the download time. But this performance increase is not due to TSUP, but is caused by another fact. In all scenarios, in the Single Tracker experiments, at the beginning all peers are started almost simultaneously, creating a flash crowd. So in this situation the communication between peers will start immediately, but when multiple trackers are unified, the TSUP imposes a delay before each peer finds out of all the others, because of the convergence time. It is known that sometimes it is better when peers enter the swarm later [12], explaining the presence of negative values for the performance decrease.

		Single Tracker			**Unified Trackers**			
seeders	*leechers*	*mean dspeed (KB/s)*	*rel.st.dev. (%)*	*trackers*	*mean dspeed (KB/s)*	*rel.st.dev. (%)*	*perf. decrease (%)*
$size = 64MB$							
1	3	337.73	1.28	1	335.11	2.87	0.78
2	6	472.27	0.87	6	396.55	16.64	16.03
4	12	475.13	1.87	4	463.42	12.60	2.46
8	24	476.10	6.06	8	497.35	11.10	-4.46
12	36	470.53	9.61	12	496.48	11.48	-5.52
$size = 256MB$							
1	3	358.72	0.45	1	356.55	1.07	0.6
2	6	476.59	0.26	6	407.55	14.67	14.49
4	12	477.56	0.45	4	477.45	10.73	0.02
8	24	492.40	6.64	8	500.56	8.49	-1.66
12	36	486.04	8.33	12	494.93	12.38	-1.83
$size = 1024MB$							
1	3	365.27	0.19	1	365.57	0.31	-0.08
2	6	477.47	0.15	6	437.09	9.12	8.46
4	12	477.54	0.18	4	466.03	5.93	2.41
8	24	423.71	6.19	8	418.69	8.73	1.18
12	36	407.71	4.47	12	418.76	6.27	-2.71

Figure 4. Tracker Networks

From results in Figure 4 several conclusions are drawn. The TSUP overhead becomes more insignificant, on the first hand, when the number of peers increases (and proportionally the number of seeders) and, on the other hand, when the file size increases. When the overhead is insignificant, the percentages have lower values. TSUP convergence time causes

 ISBN: 978-1-61208-133-5

the avoidance of a flash crown at the beginning of each scenario, thus inducing a small performance increase. Starting from 8 seeders and more TSUP performance decrease becomes smaller that the performance increase caused by avoiding the flash crowd. That is why some performance decrease values are negative. The relative standard deviation is generally increasing with the number of peers, but is decreasing when the file size increases. The values can be considered normal, taking into account the number of peers that are part of a swarm.

Due to the small values of performance decrease and relative standard deviation, we concluded that TSUP overhead is insignificant for small to medium-sized swarms (less than 50 peers) which share big files (1GB). BitTorrent is generally used for sharing large files and TSUP allows the increase of swarms size; these two factors come as an advantage for this technology.

Swarm unification increases the number of peers for a shared file, but this fact does not always grant a bigger download speed, as it can be seen in Figure 4. However, having a swarm with a bigger number of peers has three advantages. First of all increases the chance that more seeders will later be available and a big proportion of stable seeders increases download speed. The second reason is that bigger swarms increase shared file's availability by making redundancy. The third reason is that a bigger swarm is more attractive for new users, giving the possibility of creating a big social network, which is an important thing these days.

VII. Conclusion and Further Work

A novel overlay network protocol on top of BitTorrent, aiming at integrating peers in different swarms, has been presented. Dubbed TSUP (Tracker Swarm Unification Protocol), the protocol is used for creating and maintaining a tracker network enabling peers in swarms to converge in a single swarm. Each initial swarm is controlled by a different tracker; trackers use the overlay protocol to communicate with each other and, thus, take part in a greater swarm.

We have used an OpenVZ-based Peer-to-Peer testing infrastructure to create a variety of scenarios employing an updated version of the XBT Tracker, dubbed XBT Unified Tracker. The protocol incurs low overhead and overall performance. The unified swarm is close to the performance of single-tracker swarm consisting of the same number of seeders and leechers with the benefit of increased number of peers, which boosts download speed. The increased number of peers provides the basis for improved information for various overlays (such as social networks) and allows a healthier swarm – given enough peers, if some of them decide to leave the swarm, some peers will still take part in the transfer session.

Experiments have involved different swarms, with respect to the number of peers and trackers, and different file sizes using simulated asymmetric links. The table at the end of the article shows a summary of results, which prove the fact that the protocol has a low overhead.

At this point each tracker uses a statically defined preconfigured list of neighboring trackers. One of the main goals for the near future is to enable dynamic detection of neighboring trackers and ensure extended scalability. We are currently considering two approaches: the use of a tracker index where trackers' IP/host addresses and ports are stored or the use of a completely decentralized tracker discovery overlay similar to DHT's discovery methods.

As proof of concept, our test scenarios have focused on homogeneous swarms. All peers in swarms are using the same implementation and the same bandwidth limitation. All peers enter the swarm at about the same time, with some delay until swarm convergence, in case of the unified tracker protocol. We plan to create heterogeneous swarms that use different clients with different characteristics. The number of seeders and leechers in initial swarms are also going to be altered and observe the changes incurred by using the unification protocol.

Acknowledgment

This paper is suported from POSCCE project GEEA 226 - SMIS code 2471, which is co-founded through the European Found for Regional Development inside the Operational Sectoral Program "Economical competivity improvement" under contract 51/11.05.2009, and from the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/6/1.5/S/19.

Special thanks go the the P2P-Next [6] team who is working enthusiastically to deliver the next generation peer-to-peer content delivery platform. Their dedication, professionalism and vision are a constant factor of motivation and focus for our work.

References

[1] DHT & PEX. http://lifehacker.com/5411311/bittorrents-future-dht-pex-and-magnet-links-explained. [Online, accessed 31-March-2011].

[2] hrktorrent. http://50hz.ws/hrktorrent/. [Online, accessed 31-March-2011].

[3] ipoque Internet Studies. http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009. [Online, accessed 31-March-2011].

[4] libtorrent (Rasterbar). http://www.rasterbar.com/products/libtorrent/. [Online, accessed 31-March-2011].

[5] OpenVZ. http://wiki.openvz.org/. [Online, accessed 31-March-2011].

[6] P2P-Next. http://www.p2p-next.org/. [Online, accessed 31-March-2011].

[7] XBT Configuration Options. http://www.visigod.com/xbt-tracker/configuration. [Online, accessed 31-March-2011].

[8] XBT Table Documentation. http://www.visigod.com/xbt-tracker/table-documentation. [Online, accessed 31-March-2011].

[9] XBT Tracker by Olaf van der Spek. http://xbtt.sourceforge.net/tracker/. [Online, accessed 31-March-2011].

[10] H. Balakrishnan. Lecture 3; Coping with Best-Effort: Reliable Transport. 2005.

[11] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking up Data in P2P Systems. *Commun. ACM*, 46:43–48, February 2003.

[12] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Understanding and Deconstructing BitTorrent Performance. *Technical Report MSR-TR-2005-03, Microsoft Research*, 2005.

[13] R. Deaconescu, G. Milescu, B. Aurelian, R. Rughiniș, and N. Țăpuș. A Virtualized Infrastructure for Automated BitTorrent Performance Testing and Evaluation. *International Journal on Advances in Systems and Measurements*, 2(2&3):236–247, 2009.

 ISBN: 978-1-61208-133-5