# CTracker: a Distributed BitTorrent Tracker Based on Chimera

Raúl JIMÉNEZ, Björn KNUTSSON
*Kungliga Tekniska högskolan, Isafjordsgatan 39, Stockholm, 164 40, Sweden*
*Tel: +46 8 790 42 85, Fax: +46 8 751 17 93, Email: rauljc@kth.se; bkn@kth.se*

**Abstract:** There are three major open issues in the BitTorrent peer discovery system, which are not solved by any of the currently deployed solutions. These issues seriously threaten BitTorrent's scalability, especially when considering that mainstream content distributors could start using BitTorrent for distributing content to millions of users simultaneously in the near future.
In this paper these issues are addressed by proposing a topology-aware distributed tracking system as a replacement for both centralized and Kademlia-based trackers.
An experiment measuring most popular open BitTorrent trackers is also presented. It shows that centralized trackers are not topology aware. We conclude that an ideal topology-aware tracker would return peers whose latency to the requester peer is significantly lower than of a centralized tracker.

## 1. Introduction

The BitTorrent protocol [1] distributes digital content using the resources every participant offers. Every participant is called a peer and a swarm is a set of peers participating in the distribution, i.e., downloading and uploading of a given content.

Nowadays the most popular swarms on the most popular BitTorrent public trackers hold a few tens of thousand peers. BitTorrent usage is continuously increasing and with the participation of legal content distributors we can expect the usage to skyrocket. When content providers start distributing popular TV shows and movies through BitTorrent we should not be surprised to have several millions of peers participating simultaneously in a single swarm.

Big players are already moving towards on-line content distribution by using different peer-to-peer and hybrid systems, for instance, BBC with its successful iPlayer [2]. Furthermore, among others, BBC and the European Broadcaster Union participate in the P2P-Next project [3]. P2P-Next is a Seventh Research Framework Programme project, which aims to become the standard for on-line content distribution using the BitTorrent framework.

The BitTorrent tracker is a key component of the BitTorrent framework. A tracker is set up in order to track the participants within a swarm. Every peer willing to join the swarm contacts the tracker and requests a list of peers participating in the swarm. Then this peer will contact the peers in the list in order to download/upload content from/to them.

Unfortunately, there are three major open issues that threaten the reliability of the tracking system. (1) The tracker is a single point of failure. When a tracker fails the current members of the swarm are not affected but no other peer will be able to join. (2) The tracker faces a scalability issue since it is only able to handle a finite number of peers based on the processing power and bandwidth available. (3) The third issue is locality, and it is more subtle: when you contact the tracker, you will receive a random subset of the available peers. This means that while a local peer may exist, you may only be notified of peers on

other continents, resulting in both higher global bandwidth consumption and lower download speed.

The single point of failure issue has been addressed by distributing the tracking tasks among the peers. There are two implementations both based on a DHT (Distributed Hash Table) called Kademlia [4]. Several problems have, however, been found on both of them [5] and there is no visible effort towards fixing them because they are not considered critical but a backup mechanism should the tracker fail.

The scalability issue also affects the distributed trackers because the small set of nodes that are responsible for tracking a specific swarm (8 or 20 nodes in the current implementations) will receive every query. Kademlia partially distributes this responsibility by caching part of the address list on the nearby nodes. This caching feature, however, is not good enough. Although the caching nodes help replying requests, the core nodes must still keep track of every single peer in the swarm.

The locality issue is a consequence of the equality of the peers. From the tracker's point of view there is no discernible difference among peers, therefore it is not able to return a list consisting of the “best” peers, but rather just a random set of peers. In order to improve locality, the tracker could use different heuristics such as geolocation services but that would imply an extra overload. Probably that is the reason why trackers lack this feature.

This paper proposes a system that addresses the three issues described in this section.

## 2. Objectives

The main objective of this paper is to present a design for a topology-aware scalable distributed tracker based on Chimera, which addresses the issues explained in the introduction. In addition, we will outline some additional benefits of our proposed design.

We also show, through our simple experiment, that there is ample room for improvement. The difference between the ideal tracker and the current centralized tracker implementations is large enough to justify the research on this topic and the replacement of the current tracking system.

## 3. System Overview

We have undertaken a study of the behavior of the BitTorrent protocol and extensions, and the currently available DHT technologies. Based on the results, we designed and implemented a prototype and compared its behavior with the existing BitTorrent implementations [6].

In this paper a different approach is suggested. Instead of designing, implementing and deploying a completely new protocol we propose to replace just the DHT system in the current BitTorrent framework. We consider that, by being BitTorrent backwards compatible, this DHT replacement can be implemented and deployed more easily, increasing drastically the probability of a large-scale deployment.

Furthermore, we are considering, together with the Tribler research team, to integrate Chimera's key properties into the existing Kademlia-based DHT. If this is possible, the new solution would be fully backwards compatible with Mainline DHT clients. This task, however, is out of the scope of this paper and regarded as future work.

### *3.1 Distributed Tracker within the BitTorrent Framework*

BitTorrent applications using a distributed tracker have two components: (1) a peer which uses the BitTorrent protocol to download/upload data from/to other peers and (2) a node that is a member of the DHT and performs the distributed tracker's tasks.

As stated in the introduction the existing implementations of distributed trackers fail to address the scalability and locality issues. We consider that a topology aware framework offers us the properties needed to address these issues. This framework would allow us to

spread small lists of topologically close peers among the nodes; contrary to assigning the task of tracking the whole swarm to a small set of nodes (one node plus a few replicas).

There is a framework called Chimera whose properties fulfill the requirements for such system.

### *3.2 Chimera's Routing Algorithm*

In this subsection a brief description of Chimera's behavior is given. Further information about Chimera is located in the Related Work section.

Chimera [7] is a topology-aware DHT overlay. It routes each message through a number of nodes until it reaches the destination node. Every node in the path processes the message and routes it to another node whose identifier is closer –i.e, more prefix matching bits– to the destination identifier. A node is a destination of a message when there is no node whose identifier is closer in the identifier space –node and destination identifiers might match but that case is very rare.

When routing a message, a node forwards it to the topologically closest node among the candidates. This behavior makes Chimera topology aware, especially during the first hops into the DHT; contrary to the current BitTorrent DHT's behavior, where hops are randomly long all the way to the destination.

Furthermore, intermediate nodes can cache and retrieve results, a key property used by our design, which will be explained in depth later.

### *3.3 Integrating Chimera as Distributed BitTorrent Tracker*

Nodes can send two kind of messages: announce and find_peers. Every message is routed according to a modified version of Chimera's routing algorithm that is explained along this section.

An announce message contains a [IP, port number] pair and its destination is a swarm identifier. This message announces that this peer is participating in a swarm and where it can be contacted by other peers. Every node in the path stores the [peer, swarm] pair and routes the message. There is no reply for this message.

A find_peers message is addressed to a swarm identifier and it is created by a node looking for peers participating in the swarm. Every node in the path checks whether it has information about the swarm. If there is a list of peers for that swarm, that list is sent to the requester. Otherwise the message is forwarded to the next node.

So far the scalability related to the centralized tracker and locality issues have been addressed, allowing intermediate nodes to return small lists of topologically-close peers. This is still not good enough, however, since the destination of a very popular swarm –say 10 million peers– will receive every single announce message and keep track of every peer.

Solving this issue is the main contribution of this paper and it justifies replacing the current DHT used in the BitTorrent framework.

### *3.4 Scalability Improvement over the Current Distributed Trackers*

Chimera's routing algorithm can be modified to forward only a limited number of announce messages. In this way, destination nodes tracking a few tens of peers will keep track of every peer in the swarm but when the swarm reaches 10 millions of peers this node will only track a limited number of peers (the topologically closest peers).

In the modified routing algorithm there are two new parameters m and n where $n \geq m$. The parameter m is the maximum number of announce messages to be forwarded per swarm and n is the maximum number of peers stored in a swarm list. The swarm list is ordered by the distance –network latency– from the node to the peers stored in the list. These parameters can be calculated independently by every node and might be dynamic depending the node's configuration and workload. For instance, a powerful node which

wants to store every announcement received might set n to infinite, however, it must be more careful setting m in order to keep the DHT bandwidth overhead low.

Every node in the path of an announce message measures the latency to the new peer and tries to add it to the swarm list. If the list length is already n and the new peer is not closer than any other in the list then the message is dropped. If the list length is between *n* and m, the new peer will be added to the list, but the message will only be forwarded if the new element is inserted among the m lowest latency peers. Lastly, if the list is shorter than m elements, the message is forwarded following the original Chimera routing algorithm.

One may think that the fact a high latency node (e.g, satellite connections) can be isolated by dropping its announce messages is a design flaw. If this node happens to join a busy swarm and the next node in the Chimera overlay has already a list with n elements, the message will be dropped and there will be no reference to its participation in the whole DHT. Unfortunately for this node, that is exactly what is desired; close nodes are easy to find and the far-away ones are not. This node can, however, always send find_peers messages and discover other peers, therefore there is no risk of total isolation. In a sense, it would have the same effect as a peer behind a NAT or firewall, where the peer can establish connections to others but cannot be contacted by other peers.

### *3.5 Additional Benefits*

Not only will this system improve tracker's scalability, but it can also decrease costs for ISPs. Being able to find topologically close peers, peers can easily discover other peers within the same ISP, thus reducing inter-ISP traffic. Furthermore, ISPs could offer users incentives to decrease inter-ISP traffic even further (e.g., by increasing link speed in connections within the ISP's network and/or setting up an easy to find peer offering cached content).

This is not a minor benefit, since BitTorrent traffic represents an important fraction of the total Internet traffic [8] and some ISPs are trying to control this by caching, throttling or banning BitTorrent traffic, in order to reduce costs and impact on other traffic [9-10].

## 4. Centralized Tracker Versus Topology-Aware Decentralized Tracker

In this section, the results from a small-scale experiment show how topology (un)aware the most popular BitTorrent centralized trackers are. Then, these results are compared to an ideal topology-aware decentralized tracker.

The BitTorrent specifications [11] do not specify how many peers a tracker should return as a response to a peer's query, nor how these peers should be selected. It is believed that most of the tracker implementations return a list of random peers, i.e., trackers are not topology aware.

In our experiment, the most popular torrent files in Mininova.org were downloaded. Since mininova offers torrent files tracked by different trackers, several tracker implementations are analyzed at once. The results show, however, that there are no discernible differences among different trackers regarding topology awareness.

A total of 79 swarms were analyzed. Every 5 minutes a request was sent to every tracker, obtaining 79 lists of peers. Then, the latency to these peers was measured by using tcptraceroute to every peer in the list. This process was repeated 5 times.

One of the most interesting findings is that most of the peers were not reachable on the port announced to the tracker. The suggested explanations are: peers no longer on-line, NATs, and firewalls; and has been reported by other experiments on BitTorrent [12]. In total, 11578 reachable peers were measured; around 150 peers per swarm (i.e., 30 reachable peers per request on average.

In Figure 1 the average latency to the peers is plotted. For every swarm there are five points and two curves, where five points represent the average latency to the peers in the

list returned to each request. One curve represents the average latency to every peer measured within the swarm while the other shows the average latency to the *x* lowest latency peers in the swarm.

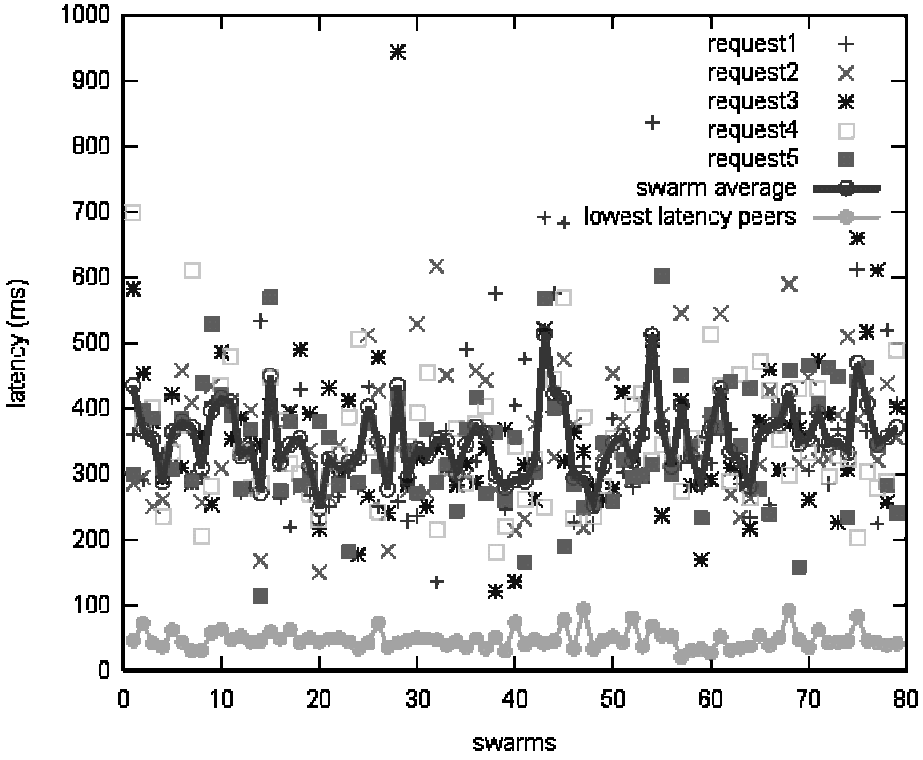


*Figure 1: Latency measurements to peers participating in popular BitTorrent swarms*

This last curve represents the ideal list of peers that a topology aware tracker should return and it is calculated as follows. The tracker returns 5 lists of torrents as response to our 5 requests. We check the reachability of peers in every list and calculate the average number of reachable peers per request which will be called *x*. If the tracker were ideally topology-aware it should have returned just the *x* lowest latency peers whose average is plotted in the figure forming the "lowest latency peers" curve.

The figure shows that the average latency for different requests is as random as we can expect, when assuming that trackers return lists of random peers. It also shows that the difference between the average latency to every peer returned (swarm average) and to the ideal list (lowest latency peers) is between 5 and 10 times. While the former backs our initial assumption, the latter shows a large room for improvement in BitTorrent trackers.

Our measurements so far have confirmed our hypothesis, but we are continuously working to study larger swarms, and we will also start monitoring swarms from multiple vantage points.

## 5. Related Work

In this section background information about DHT and Chimera is given.

### *5.1 Distributed Hash Tables*

Several structured lookup protocols [13] have been studied in order to choose one that offers the characteristics this system needs. Chord [14] is one of the most well-known DHT. Actually, Kademlia [4] is a Chord derivation used in BitTorrent. Although these protocols provide a distributed lookup system, they do not offer topology awareness.

On the other hand, Tapestry [15] is a structured lookup protocol that provides this characteristic. Moreover, its implementation in C –called Chimera– is flexible enough to be

adapted to our needs. In this project, Chimera was chosen as lookup overlay, whose description will be explained next.

### *5.2 Chimera*

As an implementation of Tapestry, Chimera routes messages from one node to a destination's root. In Chimera, each node has a unique identifier. Each node has several routing tables, with references to its nearest neighbors within a level.

A link belongs to a level, depending on the length of the shared prefix of the identifiers of the two nodes involved. For instance, let identifiers in hexadecimal and 4-bit levels, node 6E83 has links of level 4 to nodes whose identifier are 6E8*, level 3 to nodes 6E** and so on. In fact, this is similar to IP routing.

A message from one node to another is routed choosing the highest level link in each step. Since each step routes the message through a greater level, the maximum number of hops is $\log_{\beta}(N)$, where the identifiers are expressed in base $\beta$ and N is the length of the identifier. Moreover, since each node routes the messages through its nearest neighbor in that level, the paths are deterministic and topologically aware. At the last hop, the message's and the node's identifiers match, then the message is delivered.

Each object –torrent identifier– has a unique identifier as well. When any node wants to perform an operation over an object (publish, unpublish, lookup, etc.), the message is routed to the block's identifier. Since most likely there will not be a node matching it, the message will reach its destination's root. This is the node whose identifier is the closest to the block's one.

Then, a publish(objectID) is delivered to the objectID's root and this node stores all the references to objectID. In the path, each node which forwards the publish messages also stores the references. A lookup message will be routed in the same manner, however, at any hop it will reach a node which stores a list of references –list of peers participating in the swarm. This node can stop the lookup and return its list of references. This list will be shorter than the root's one and these references were likely published by the closest nodes to the requester.

The main difference between Chimera and Tapestry is that Tapestry reaches the node that actually published an object, while Chimera only routes the messages. Because of the aforementioned property it was possible to use Chimera in this design.

## 6. Conclusions

In this paper a design of a topology-aware distributed BitTorrent tracker has been presented. This design addresses three key open issues in the current BitTorrent tracker system. We explained how the scalability of the whole system is improved drastically by removing the existence of hot spots in the DHT. This is a desirable property nowadays but it will be absolutely necessary when mainstream content providers offer their content on the BitTorrent framework in the near future.

ISPs will play a key role in P2P content distribution. This design provides mechanisms to reduce inter-ISP traffic, improving user experience (lower lookup latency and increased download speed), without increasing dramatically the ISP's costs.

Our experiment has shown how the most popular open BitTorrent trackers behave and how far their results are from an ideal topology aware system. This backs our initial hypothesis that there is a large room for improvement and encourages us to implement the described system in order to measure its performance.

Future work includes modifying Tribler [16], a BitTorrent-based content distribution framework developed by a research team at Delft University, integrating this design, and comparing performance against the current unmodified version.

## Acknowledgment

The research leading to these results has received funding from the Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 21617.

## References

[1] B. Cohen. Incentives build robustness in BitTorrent. In Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA, June 2003.
[2] BBC. iPlayer. http://www.bbc.co.uk/iplayer/ (April 2008)
[3] P2P-Next. http://www.p2p-next.org/ (April 2008)
[4] P. Maymounkov and D. Mazieres. Kademlia: A peerto -peer information system based on the xor metric. In Proceedings of IPTPS02, Cambridge, USA, March 2002. http://www.cs.rice.edu/Conferences/IPTPS02/.
[5] Scott A Crosby and Dan S Wallach An Analysis of BitTorrent's Two Kademlia-Based DHTs Technical Report TR-07-04, Department of Computer Science, Rice University, June 2007.
[6] R. Jiménez. Ant: A Distributed Data Storage And Delivery System Aware of the Underlaying Topology. Master Thesis. KTH, Stockholm, Sweden. August 2006.
[7] CURRENT Lab, U. C. Santa Barbara. Chimera Project. http://current.cs.ucsb.edu/projects/chimera/ (April 2008)
[8] A. Parker. The true picture of peer-to-peer filesharing, 2004. http://www.cachelogic.com/. (April 2008)
[9] TorrenFreak. Virgin Media CEO Says Net Neutrality is “A Load of Bollocks”. http://torrentfreak.com/virgin-media-ceo-says-net-neutrality-is-a-load-of-bollocks-080413/ (April 2008)
[10] The Register. Californian sues Comcast over BitTorrent throttling. http://www.theregister.co.uk/2007/11/15/comcast_sued_over_bittorrent_blockage/ (April 2008)
[11] Bram Cohen. The BitTorrent Protocol Specification http://www.bittorrent.org/beps/bep_0003.html (April 2008)
[12] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "A measurement study of the bittorrent peer-to-peer file-sharing system," Tech. Rep. PDS-2004-007, Delft University of Technology, Apr. 2004.
[13] Frank Dabek, Ben Zhao, Peter Druschel, and Ion Stoica. Towards a common API for structured peer-to-peer overlays. In IPTPS '03, Berkeley, CA, February 2003.
[14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. Technical Report TR-819, MIT, March 2001.
[15] Ben Zhao, John Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, April 2001. 55
[16] J.A. Pouwelse and P. Garbacki and J. Wang and A. Bakker and J. Yang and A. Iosup and D.H.J. Epema and M. Reinders and M. van Steen and H.J. Sips (2008). Tribler: A social-based peer-to-peer system. Concurrency and Computation: Practice and Experience 20:127-138. http://www.tribler.org/