Fewest Common Hops (FCH): An Improved Peer Selection Approach for P2P Applications

Humaira Ijaz
University of Innsbruck, Austria
University of Sargodha, Pakistan
Innsbruck, Austria
Email: humaira.ijaz@uibk.ac.at

Sadia Saleem
School of Computer Science
University of Manchester, UK
Manchester, UK
Email: saleems@cs.man.ac.uk

Michael Welzl
Department of Informatics
University of Oslo, Norway
Oslo, Norway
Email: michawe@ifi.uio.no

***Abstract*—Underlay-unawareness in P2P systems can result in sub-optimal peer selection for overlay routing and hence poor performance. The majority of underlay aware proposals for peer selection focus on finding the shortest overlay routes by selecting the nearest peers according to proximity. However, in case of multiple and parallel downloads, if the underlay paths between a downloader and its selected nearest peers share a bottleneck, this can cause congestion, leading to performance deterioration instead of improvement. This effect was neglected in previous work because, in today's Internet, the bottleneck is usually not shared as it is the end user's access link. This is no longer the case in more modern scenarios, e.g. with FTTH or with upcoming in-network caching techniques such as DECADE. We propose an improved peer selection approach for P2P applications called Fewest Common Hops (FCH) that ensures proximity based node selection having maximum path disjointness. It is a client based, infrastructure independent heuristic to optimize download time for multiple and parallel downloads in P2P content distribution applications. Simulations show that, even when FCH is implemented in the simplest possible fashion (using only traceroute), it can significantly decrease the download time.**

***Keywords*-fewest common hops; peer Selection; path disjointness; multiple downloads;**

I. Introduction

Peer to Peer (P2P) systems were developed initially for file sharing, e.g. Napster, Gnutella but later they have become popular for content sharing, media streaming, telephony applications etc. P2P system is a virtual or logical network of nodes, often called overlay, because it is formed at the application layer, on top of a physical network. This underlying physical network is called underlay. All P2P systems have a certain deliberate ignorance about the underlay that allows them to form an entirely new network, which is completely under the control of the application it is designed for. Hence, many P2P systems perform their own routing function at the application layer called overlay routing, thereby allowing end nodes to choose routing paths themselves.

Although this application level peer selection gives flexibility to P2P applications to choose paths, underlay-unawareness and continuous change in underlay properties such as bandwidth and loss rate can result in sub-optimal peer selection and overlay routing that generates a large amount of unnecessary traffic [1][2] – a significant waste, given that P2P filesharing is the dominant traffic type of the Internet [3]. As with all Internet traffic, it can cause congestion and performance deterioration, which can thereby limit the scalability of the overlay, resulting in customer dissatisfaction. All these problems stress the use of underlay awareness for peer selection.

To overcome this problem, many overlay optimization mechanisms have been proposed that use underlay information for peer selection. Some are based on passive or active probing, and some approaches strive for using ISPs or third parties to provide underlay information for node selection. However, neither ISPs nor other third parties (e.g. other P2P applications) might cooperate because there are issues of security, privacy, storage, and continuous maintenance of information. The majority of underlay-aware proposals for peer selection is therefore focused on finding shortest overlay routes by selecting nearest nodes according to proximity information [4][5]. Apparently this selection could yield better performance [5][6] – but proximity based node selection alone is not sufficient. If the selected shortest overlay routes are shared between peers, this can result in congestion on shared bottlenecks and ultimately in suboptimal performance. In order to overcome this problem, there is therefore a need to find the non-shared paths both at access and core networks for peer selection.

Shared bottlenecks can exist both at the access and core level due to the increase in Internet usage and advancement in Internet technology. Fiber to the home (FTTH) is an example of advancement in technology that provides high speed at access links i.e. 100 Mbit/s or its variants in countries like japan, where it will become gigabit FTTH by 2020 [7]. The largest operators have limited capacities in terms of tens of gigabits, approximately having thousands of gigabit FTTH customers, making a shared bottleneck at the core level a genuine possibility. Deployment of the IETF DECADE standard could become another cause for shared bottlenecks, as it strives to let P2P applications operate on caches that are located beyond the typical last-mile bottleneck. Finally, in developing countries, there is a trend of sharing one Internet connection among many users, e.g. in an Internet cafe, causing shared bottlenecks between close nodes at the access level. All these problems stress the importance of path disjointness for peer selection.

Path disjointness means that selected overlay paths should

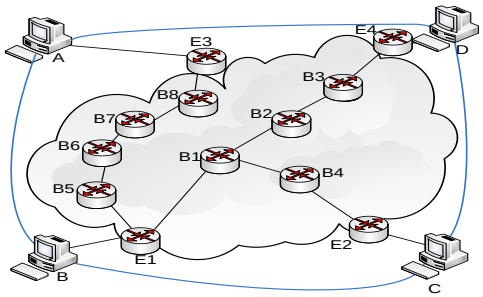

Fig. 1. An example of proximity based peer selection

have a minimum number of common intermediate hops. If selected peers are nearest according to proximity but do not have disjoint paths, there is a chance that a part of their shared path becomes a bottleneck, which means that the downloads negatively affect each other. Figure 1 shows an example overlay network of four nodes {A,B,C,D} connected with a network of backbone routers $\{B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8\}$ through edge routers $\{E_1, E_2, E_3, E_4\}$. Suppose node B wants to perform multiple downloads and it is using a proximity based peer selection algorithm, e.g. Pastry [4].

For the first download, node B performs a lookup and finds two nodes C and D storing replica of the required file. Node B selects node C for download because it is nearest (4 hops away, as compared to the 5 hops needed to reach node D). For the second download, node B performs another lookup and finds two nodes A and D storing replica of the file. Node B selects node D because it is nearest according to proximity. However, link $E_1 - B_1$ is shared between two downloads because both C and D would send data through this common link simultaneously. This can result in congestion and an increased download time.

To address the above problem, we have designed, implemented, and evaluated a simple yet efficient peer selection heuristic for multiple and parallel downloads, called Fewest Common Hops (FCH). FCH enables a node to select a peer that is nearest according to proximity but has maximum path disjointness. To do so, a client peer uses traceroute to all available candidate peers and stores the resulting path topology from the client to these candidate peers. We pick traceroute because it is the simplest possible mechanism that can indicate path disjointness; more sophisticated measurement methods could be used in future work. The number of traceroutes is equal to the number of replica used in the P2P system; the storage and time effort is therefore $O(m)$, where m is the number of replica. Afterwards, the traceroute results are compared to select the peers that have the fewest routing hops and maximum path disjointness with paths that are already selected for downloads (in case of multiple downloads).

FCH is meant for efficient network resource utilization in P2P content distribution applications where the setup latency is non-critical (like file downloads, not real-time streams where users want an immediate reaction). However, one can still apply FCH for real-time streams later and switch between peers if that gives a benefit. FCH runs on the application layer of the client, and does not require any third party information. Moreover, it does not contradict existing peer selection algorithms, such that any existing peer selection algorithms could be enhanced with it.

The remainder of this paper is structured as follows: Section 2 critically analyses existing peer selection algorithms designed for P2P systems to identify open issues that need further research. Section 3 presents an overview of our FCH peer selection algorithm, Section 4 outlines the simulation setup and presents experimental results. Section 5 concludes.

II. Related Work

Overlay optimization mechanisms for proximity-based peer selection in P2P applications can be categorized as follows:

A. Grouping/Clustering of nodes

The approaches in this category use proximity information for arranging nodes into groups. P2P applications can get this information by estimating the network position of peers with reference to a) landmarks (e.g. GNP [8]) b) IP-Prefixes (e.g. [9]) and c) super peers (e.g. [10]), via network measurements like Time-to-Live (TTL) or Round-Trip Time (RTT) between peers, or via geographical position of nodes (e.g. [11]). According to [12] network positioning shows noticeably worse performance as compared to results shown in [8].

B. Use of ISPs

Another approach is to use ISPs or third parties to get information such as topology, bandwidth, loss rate from the underlay and provide it to P2P clients. Xie et al. [13] proposed provisioning of underlay information as a service by a third party. P2P clients can use this information for peer selection. The IETF has also recently formed the Application Layer Traffic Optimization (ALTO) working group that provides underlying network information to P2P clients with the help of ISPs or third parties [14]. Similarly, [15], [16] investigated node selection with the help of an ISP "oracle" according to ISP preferences. In [17] a gametheory framework is proposed that facilitate design of ISPs cooperative policies for P2P applications to reducing inter-domain traffic. For maximizing path independence, [18], [19] exploit path redundancy and use multi-homing at endpoints and at the stub network level. These approaches have issues related to privacy, storage, continuous maintenance, bandwidth consumption, and continuous probing. Efforts are therefore being made to solve these problems.

C. Probing based Overlays

A few probing based P2P applications [4], [20] use passive probing to get the proximity information for node selection whereas [21], [22] use a combination of active and passive probing for topology discovery, node selection, monitoring the functioning and quality of Internet paths, route failure detection and recovery. ROR [23] uses continuous periodic beaconing of normal and pre-calculated backup paths for fast failure detection.

In these applications, each peer would ideally send probing messages to all other peers, store this information, and then

do a lookup on that database to select a peer. The probing cost then reaches up to $O(n^2)$, and hence some restrictions must be in place. Regarding the time effort, the overhead is in the order of O(log n) if there are O(log n) routing messages, and if it is a constant additional effort on top of every routing message that is sent. These probing techniques are also used for calculating alternate paths for construction of fault tolerant overlay networks [21], [23].

These probing based overlays can be realized for small or medium sized P2P systems because the discovery of the topology or alternate paths requires a lot of active and passive probing messages and their maintenance also requires continuous probing which generates a large amount of unnecessary traffic, limiting the application's scalability. Redundancy is another issue in these probing based P2P systems because the network itself already has a complete picture of the network from different network vantage points. To rediscover this information by probing is a waste of time and resources. Further, none of these algorithms investigate the path sharing needed by multiple downloads or parallel downloads.

III. FCH

FCH is a simple and scalable peer selection algorithm, based on proximity and maximum path disjointness for multiple and parallel downloads. It ensures that a peer chosen for download among candidate peers is nearest and has maximum path disjointness with peers that are already selected for downloads by the same client. It has three main components:

1) Whenever a download is initiated, the *Path Topology Extractor (PTE)* infers the path topology between the client and the candidate peers. The path topology is just a list of all intermediate routing hops between the client and the candidate peer. To obtain this path topology, PTE uses pair-wise traceroute [24]. However, the design is flexible enough to use any end-to-end measurement tool to find the underlay topology. PTE stores the path topology of every candidate peer as a separate topology set $S_i : i = 1, ..., n$ and provides these topology sets to Path Topology Comparator.
2) The *Path Topology Comparator (PTC)* compares the path topology set of every candidate peer with path topology sets of peers that are already selected for downloading. Then PTC selects the path topology set that has the fewest common hops with path topologies of these selected peers in case of multiple or parallel downloads. Let us consider the scenario of multiple downloads.
   Case-1 (first download): PTC checks the cardinality of the topology sets and the set having the smallest cardinality is selected as a set S. S is transferred to the peer selector module. PTC also puts nodes of set S into set SN which is used by PTC to determine path disjointness for future downloads, where SN contains topology sets of all selected peers.
   $minDistance = min(|S_1|,|S_2|, ...|S_n|)$
   $S = \{S_k: |S_k| = minDistance\}$
   $SN \Longleftarrow S$
   Case-2 (all later downloads): In this case, PTC compares every set $S_i : i = 1, ..., n$ with set SN to find the number of common elements.
   $cNodes_i = \{x: x \in SN \cap S_i\}$ where $i = 1, ..., n$
   PTC then uses the number of common intermediate routing hops between the candidate peer and already selected peers $|cNodes_i|$to select the set S with the smallest number of common hops.
   $minHops = min(|cNodes_1|, |cNodes_2|,...|cNodes_n|)$
   $S = \{S_k: |S_k| = |cNodes_k| = minHops\}$
   where $S_k \in S_i$
   $SN = \{SN \cup S\}$
   In both cases, after the path topology comparison, PTC will send the set S to the Peer Selector.
3) The *Peer Selector (PS)* chooses the peer from the set S. It receives S from PTC and compares the nodes of the set S with all candidate peers. In this way, the nearest candidate peer having maximum path disjointness with other, already selected peers is chosen for downloading.

A. Example

Let us now consider how FCH works in the scenario described in Figure 1.

Case-1 (first download): For Node B to select peers in case of two downloads, FCH operates as follows:
PTE extracts the path topology of the candidate peers C and D and store them in two sets of nodes, S_1 and S_2, respectively.
$S_1 = \{E_1, B_1, B_4, E_2, C\}$
$S_2 = \{E_1, B_1, B_2, B_3, E_4, D\}$
PTC checks the cardinality of sets S_1, S_2 and the set S_1 having smallest cardinality is selected as set S.
$|S_1| = 5$, $|S_2| = 6$; $minDistance = min(5, 6)$
$S = \{S_1: |S_1| = 5\}$
$S = \{E_1, B_1, B_4, E_2, C\}$
The set S is transferred to the Peer Selector module. PTC also copies nodes of set S into set SN which is used by PTC to determine path disjointness for future downloads.
$SN \Longleftarrow \{E_1, B_1, B_4, E_2, C\}$
Case-2 (second download): PTC obtains the sets S_1, S_2 representing the path topology of the two candidate peers D and A from PTE.
$S_1 = \{E_1, B_1, B_2, B_3, E_4, D\}$
$S_2 = \{E_1, B_5, B_6, B_7, B_8, E_3, A\}$
Afterwards, it compares the sets S_1, S_2 with the set SN to find the number of common elements.
$SN = \{E_1, B_1, B_4, E_2\}$
$cNodes_1 = \{E_1, B_1\} = S_1 \cap SN$
$cNodes_2 = \{E_1\} = S_2 \cap SN$
Here, $|cNodes_1|$ represents the number of common intermediate routing hops between S_1 and SN and $|cNodes_2|$ represents the number of common intermediate routing hops between S_2 and SN. PTC then uses $|cNodes_i|$ to select the peer with the fewest common hops.
$minHops = min(|cNodes_1|, |cNodes_2|) = min(2, 1)$

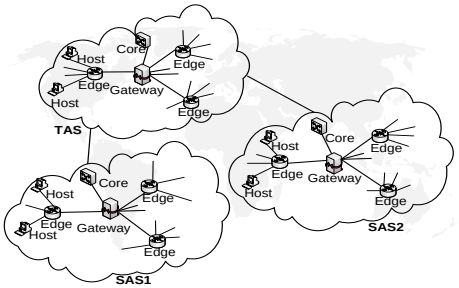

Fig. 2. The TSA topology

$S = \{S_k: |S_k| = |cNodes_k| = minHops\}$
$SN = \{E_1, B_1, B_4, E_2, C\} \cup S_2$
$SN = \{E_1, B_1, B_4, E_2, C, B_5, B_6, B_7, B_8, E_3, A\}$
In both cases, after the path topology comparison, PTC will send S to PS and add the elements of set S in SN to determine path disjointness for future downloads. For the first download PTS compares nodes of set S with nodes C and D and finds and selects node C. For the second download, it selects node A after comparing the nodes of set S with nodes A and D.

IV. Performance Evaluation

The distributed behavior of large scale overlays makes their development and evaluation highly complex. To overcome this complexity, simulations have proven indispensable for the design, development and evaluation of overlays. We have therefore implemented and evaluated a prototype for FCH using simulations. Since we expect the problem that FTH solves – shared bottlenecks in the network – to become significant in the future due to e.g. FTTH and DECADE, it is hard to additionally carry out a credible real-life experiment. We have carried out a series of PlanetLab tests with the shared bottleneck detection tool in [25] and found that, as accepted, there are no shared bottlenecks *inside* the network, as the bottleneck is always the hosts's acces link. However, in the future we plan to at least use Real Network Support provided by Oversim to carry out an emulation based experiment, as a compromise between a real-world environment and simulation.

A. Simulation Setup

We used the OMNET++ based Overlay Simulation Framework "OverSim" [26] that encompasses the simulation models of all network layers from MAC to application layer. It allows the design, evaluation and comparison of different P2P networks at a large scale with a choice of different underlying Network Models. We selected the ReaseUnderlay because it generate a realistic Internet like network topology with realistic bandwidths, packet delays, and packet losses and can also generate self-similar background traffic. . We used a transit-stub topology available in Rease called "TSA", shown in Figure 2. TSA has 1 transit domain, 2 stub domains and 9 autonomous systems with realistic data rates similar to the GÉANT topology model and the FTTH Internet model in Japan [7]. Fifteen routers are arranged hierarchically. The transit domain has one transit/core router to connect stub routers of the stub domains by a 10Gbps/20ms traonsit-stub link. Every stub domain has a stub router to connect with a transit router of the transit domain and a gateway router to connect with three autonomous systems via a 1Gbps/1ms STUB-AS link. Every gateway router is connected with three access routers via a 155Mbps/1ms GW-AS link. The end hosts/nodes are connected with this access router by an access link with a datarate of FTTH, i.e. 100Mbps, and delay 5ms.

We also repeated all experiments with a TSI topology (using INETUnderlay), where 6 backbone/core routers are connected by a 10Gbps/1ms NE-PON channel and four of these backbone routers are connected with access routers (each representing an autonomous system) by a 1Gbps/30ms G-PON channel. End hosts are connected to the access router via a 100Mbps/10ms E-PON (Ethernet PON) channel [27]. Since the results were similar, we only present the results that we obtained with the more sophisticated model (TSA using ReaseUnderlay).

LCH is implemented as an extension of the Distributed Hash Table (DHT) based P2P system "Pastry" that exploits physical network proximity by applying certain heuristics in its routing tables. We conducted an experiment in which we took 14 sets of nodes, consisting of 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500 nodes respectively. In our experiment end nodes are running Pastry [4] to form an overlay. Pastry in Oversim sends put requests to put data/replica on different nodes. Similarly, download requests are sent randomly from X nodes (downloaders) to X nodes (sources/replica). For Pastry the lookup requests of a client always finds a set of numerically closest nodeIds storing the replica and selects a node from this set that is closest in the network according to a proximity metric. The proximity metric is the RTT from ping.

Our data points show the cases where there was one downloader doing multiple downloads from different sources/replica but having a shared path between these sources/replica. We measured the RTT from the downloader to FCH-based selected source and Pastry-based selected source. Not all sources selected by Pastry are nearest according to proximity; we have therefore restricted our observations to only those Pastry based sources that were actually nearest in terms of the proximity metric for comparison with FCH based source selection. For both kinds of selections, we took the average of results of 100 download requests seen at all the destinations (downloaders) for every set. The number of replica was also varied from 1-6, i.e. the experiment was repeated six times for every set and again the average of these results was taken. We compared the performance of Pastry with and without applying FCH and observed that FCH can significantly reduce the download time as compared to Pastry-based node selection. For evaluating the performance of FCH, the average download time of FCH-based source selection is plotted against average download time of Pastry-based source selection for all sets of nodes as shown in Figure 3. The red line in the graph shows the download time with Pastry-based nearest source selection and the black line shows the download time with FCH based source selection.

Based on 100 measurements, the average download time of 10-70 nodes with FCH-based source selection is always at least 22% less than with Pastry-based source selection. The

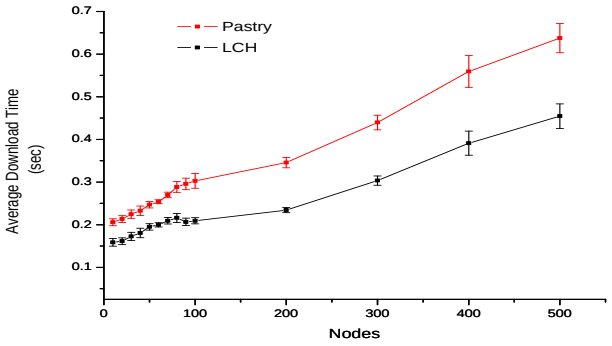

Fig. 3. FCH-based vs Pastry-based source selection

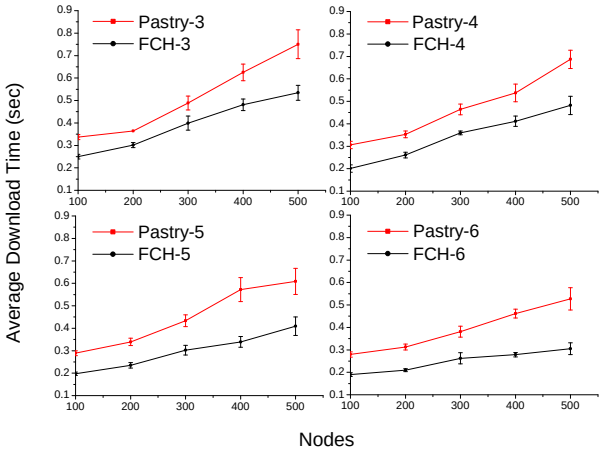

Fig. 4. FCH-based vs Pastry based random node selection

download time increases with the number of nodes but the performance of FCH also improves because the gap between two lines becomes wider after 70 nodes. With 500 nodes, FCH reduces the average download time by 29%.

We also evaluated how the performance depends on number of replica. For this, we observed the average download time of 20 download requests in 6 separate cases, i.e. with 1-6 replica, against sets of nodes i.e. 100-500. The cases of 1-2 replica look uninteresting because FCH only began to take effect from 3 replica onward; so the cases of 3-6 replica are shown in Figure 4. The red lines in these graphs show the average download time of Pastry-based nearest source selection and black lines show the average download time of FCH based source selection. Pastry-3, FCH-3 refers to the case of 3 replica, Pastry-4, FCH-4 shows the download time for 4 replica and so on. The average download time based on 100 measurements of FCH-3 is 22% less than Pastry-3. Performance of FCH increases with number of replica – for FCH-6 the average Pastry-6 download time is reduced by 33%.

V. Conclusion and Future Work

We have developed the FCH peer selection algorithm based on proximity and maximum path disjointness, which will become increasingly relevant as the Internet moves towards faster access links (e.g. FTTH) and incorporates more in-network caching techniques such as DECADE. We have implemented and evaluated a prototype for FCH using Pastry and Oversim. Results show a significant decrease in download time as compared to Pastry. Our implementation of FCH assumes we have topology knowledge from traceroute;clearly, more sophisticated measurement methods for shared bottleneck detection could also be applied. ISPs or third parties can also apply this heuristic in combination with their own techniques for providing underlay information that helps in node selection to clients.

We have considered multiple data transfers in this work, where a client simultaneously downloads different contents from different sources, forming a one-to-many relationship. A single parallel download is a similar situation where the content is divided into different chunks, and the client gets each chunk from a different source. We therefore plan to study the applicability of FCH for parallel downloads. One particularly interesting case that we plan to investigate is parallel downloading with BitTorrent [28], which continuously measures the per-source download speed and accordingly adapts its choice of peers, but does not consider correlations between these sources as it would be needed in order to take shared bottlenecks in the underlay into account. Finally, we will examine the cases where $minDistance$ is small having large $minHops$ or $minDistance$ is large having small $minHops$. To design a peer selection policy that can balance both traffic and download time, we need to study all factors that can affect these two targets and derive an appropriate metric for peer selection.

References

[1] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 219–232, Apr. 2004.

[2] H. Zhang, J. F. Kurose, and D. F. Towsley, "Can an Overlay Compensate for a Careless Underlay?" in *INFOCOM*, 2006.

[3] http://www.ipoque.com/en/resources/internet-studies.

[4] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," 2001.

[5] Y. Liu, X. Liu, and L. Xiao, "Location-aware topology matching in p2p systems," in *Proceedings of IEEE INFOCOM*, 2004.

[6] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Topology-aware routing in structured peer-to-peer overlay networks," in *Future directions in distributed computing*. Springer-Verlag, 2003, pp. 103–107.

[7] http://en.wikipedia.org/wiki/internet_in_japan.

[8] T. S. E. Ng and H. Zhang, "Global network positioning: a new approach to network distance prediction," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 1, pp. 73–73, Jan. 2002.

[9] L. Garces Erice, K. W. Ross, E. W. Biersack, P. A. Felber, and G. Urvoy Keller, "Topology-centric look-up service," in *NGC'03, 5th International Workshop on Networked Group Communications, September 16-19, 2003 - Munich, Germany*, Munich, GERMANY, 09 2003.

[10] J. Yu and M. Li, "CBT: A proximity-aware peer clustering system in large-scale BitTorrent-like peer-to-peer networks," *Comput. Commun.*, vol. 31, no. 3, pp. 591–602, Feb. 2008.

[11] H. Doan, L. T. H. Hanh, and S. Andrew, "An optimised geographically-aware overlay network," in *INDIN '05*, 2005, pp. 372 – 377.

[12] D. R. Choffnes, M. A. Sánchez, and F. E. Bustamante, "Network positioning from the edge: an empirical study of the effectiveness of network positioning in p2p systems," in *INFOCOM'10*, 2010.

[13] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: provider portal for applications," in *SIGCOMM 2008*, 2008.

[14] J. Seedorf, S. Kiesel, and M. Stiemerling, "Traffic localization for P2P-applications: The ALTO approach," in *Proc. P2P*. IEEE, 2009.

[15] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P users cooperate for improved performance?" *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 29–40, Jul. 2007.

[16] S. James and P. Crowley, "IMP: ISP-Managed P2P," in *Proc. P2P*, 2010.

[17] V. Bioglio, R. Gaeta, M. Grangetto, M. Sereno, and S. Spoto, "A game theory framework for isp streaming traffic management," *Perform. Eval.*, vol. 68, no. 11, pp. 1162–1174, Nov. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.peva.2011.07.007

[18] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, "Best-path vs. multi-path overlay routing," in *Proc. IMC*. ACM Press, 2003.

[19] J. Han, D. Watson, and F. Jahanian, "Topology aware overlay networks," in *IEEE INFOCOM, 2005, VOL 4*, 2005, pp. 2554–2565.

[20] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," University of California at Berkeley, Berkeley, CA, USA, Tech. Rep., 2001.

[21] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. SOSP'01*, 2001.

[22] Y. Chen, D. Bindel, and R. H. Katz, "Tomography-based overlay network monitoring," in *Proc. ICM'03*, 2003.

[23] B. Y. Zhao, L. Huang, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz, "Exploiting routing redundancy via structured peer-to-peer overlays," in *in ICNP*, 2003, pp. 246–257.

[24] ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

[25] B. Y. Muhammad Murtaza Yousaf, Michael Welzl, "Accurate shared bottleneck detection based on svd and outlier detection," University of Innsbruck, Institute of Computer Science, Tech. Rep. DPS NSG Technical Report 1, August 2008. [Online]. Available: http://heim.ifi.uio.no/michawe/research/publications/

[26] I. Baumgart, T. Gamer, C. Hübsch, and C. P. Mayer, "Realistic underlays for overlay simulation," in *Proc. ICST SIMUTools '11*, 2011.

[27] http://argo-contar.com/news/pon-mitsubishi.pdf.

[28] http://www.bittorrent.org/beps/bep_0003.html.