หลักการในการเพิ่มประสิทธิภาพการสื่อสารข้อมูลผ่านชั้นโปรแกรมประยุกต์ โดยใช้วิธีการบิตทอร์เรนต์ในระบบการแชร์ไฟล์แบบเพียร์ทูเพียร์

ณัฐที ปิ่นทอง

The Novel Principle of Application Layer Traffic Optimization By BitTorrent-Like Peer to Peer File Sharing Systems

Nattee Pinthong

สาขาวิชาคอมพิวเตอร์ศึกษา คณะครุศาสตร์ มหาวิทยาลัยราชภัฏราชนครินทร์ ฉะเชิงเทรา 24000
Department of Computer Education, Faculty of Education, Rajabhat Rajanagarindra University, Chachoengsoa 24000, Thailand
Corresponding author, E-mail address: nattee@ieee.org

บทคัดย่อ

ในช่วงเวลาที่ผ่านมา บิตทอร์เรนต์ได้กลายมาเป็นกลไกที่สำคัญต่อการกระจายข้อมูลผ่านเพียร์ทูเพียร์ในวงกว้าง ในขณะที่การศึกษาวิเคราะห์ประสิทธิภาพของบิตทอร์เรนต์ยังไม่ดีเท่าที่ควร ในบทความฉบับนี้ได้มีการนำเสนอหลักการในการเพิ่มประสิทธิภาพการสื่อสารข้อมูลผ่านชั้นโปรแกรมประยุกต์ โดยใช้วิธีการบิตทอร์เรนต์ในระบบการแชร์ไฟล์แบบเพียร์ทูเพียร์ โดยจากการศึกษาล่าสุดชี้ให้เห็นว่าความสัมพันธ์ระหว่างเพียร์แต่ละเพียร์ในบิตทอร์เรนต์สามารถเพิ่มประสิทธิภาพอัตราการดาวน์โหลดที่สูงขึ้นได้ อันเนื่องมาจากกลไกที่สำคัญในบิตทอร์เรนต์ ยกตัวอย่างเช่น ระบบกลไกของทิกฟอร์แท็ก กลไกของออปติมิสติกอันโช๊กกิ้ง กลไกของแอนติสนับลิ้ง และกลไกพีซซีเล็กชั่น เป็นต้น โดยกลไกที่กล่าวมาทั้งหมดสามารถทำงานร่วมกันได้เป็นอย่างดีอยู่แล้ว ซึ่งในบทความฉบับนี้มีวัตถุประสงค์เพื่อ 1) นำเสนอแนวคิดในการที่จะทำให้โหนดในแต่ละจุดของการเชื่อมต่อในบิตทอร์เรนต์ทำงานผ่านโทโพโลจีแบบอันเดอร์ไลอิ้งของระบบเครือข่ายได้ดีขึ้นกว่าเดิม 2) ทำการปรับปรุงอัลกอลิทึมของบิตทอร์เรนต์แบบเดิมแทนที่ด้วยอัลกอลิทึมแบบใหม่ที่ชื่อว่าแทรกเกอร์โลคอลไลซ์อัลกอลิทึม โดยใช้พื้นฐานของหลักการออโตโนมัสเอเอสฮอป 3) จากนั้นจะทำการศึกษาความครอบคลุมถึงประสิทธิภาพในการเปรียบเทียบอัตราของระยะเวลาในการดาวน์โหลดไฟล์ และระยะเวลารวมทั้งหมดของการดาวน์โหลดไฟล์โดยในละแต่วิธีการ โดยได้มีการนำแบบจำลองการทำงานแบบเพียร์ซิมมาใช้ ซึ่งผลการจำลองข้อมูลแสดงให้เห็นว่า หลักการแบบแทรกเกอร์โลคอลไลซ์อัลกอลิทึม ให้ผลลัพธ์ที่ดีกว่าการทำงานของระบบบิตทอร์เรนต์รูปแบบเดิม นอกจากนี้ในบทความยังชี้ให้เห็นว่ากลไกนี้สามารถลดการสื่อสารข้อมูลที่ไม่จำเป็นได้อย่างมีประสิทธิภาพทั้งระบบเครือข่าย

คำสำคัญ: ระบบเครือข่ายเพียร์ทูเพียร์ การเพิ่มประสิทธิภาพในการสื่อสารข้อมูล โพรโทคอลบิตทอร์เรนต์

Abstract

In recent years, BitTorrent has emerged as a very scalable peer-to-peer file distribution mechanism. While early measurement and analytical studies have verified BitTorrent's performance. In this work present The Novel Principle of Application Layer Traffic Optimization By BitTorrent-Like Peer to Peer File Sharing Systems. Recent studies suggest that the long-term relationships among BitTorrent peers be explored to enhance the downloading performance. In such systems, the mechanisms like Tit-for-Tat, Optimistic Unchoking, Anti-Snubbing and Various Piece Selection Strategy have been working very well. The purpose of this work are 1) To study the analysis of BitTorrent protocol and its networking infrastructure through the concept of offering some guidelines to make BitTorrent node aware of the topology of underlying networks. 2) To modify BitTorrent's original algorithms and replace them with Tracker Localized Algorithms based on autonomous system (AS) hops. And 3) to conduct a comprehensive performance comparison of the average

download time of peers for original BitTorrent algorithm and the Tracker Localized algorithm, Then to conduct a comprehensive performance comparison of the time of all peers breaking downloading for original BitTorrent algorithm and the Tracker Localized algorithm based on the PeerSim simulation. The simulation result shows that, with our scheme the nodes in BitTorrent-like P2P systems have better sense of the topology of their underlying networks, and can interact more efficiently.

Keywords: Peer-to-Peer, Application Layer Traffic Optimization, BitTorrent Protocol

Introduction

Peer-to-Peer (P2P) networks have emerged as a successful architecture for content sharing over the Internet. BitTorrent (BT), the most popular P2P application, has attracted significant attention from network operators and researchers for its wide deployment. Since the Peer-to-Peer has been proposed, many kinds of Peer-to-Peer protocols were implemented. The BitTorrent has been the most successful protocol for Peer-to-Peer file sharing since it discovered in 2001 (Cohen, 2003). More and more users are continually joining such systems and more resources are being made available, which attracts even more users to join. P2P applications include file sharing systems such as Napster, Kazaa, eDonkey and BitTorrent. The ever-increasing traffic not only brings enormous revenues to Internet Service Providers (ISPs) but also poses a significant engineering challenge to ISPs by building overlay networks (Urvoy, et al., 2006) that are oblivious to the underlying Internet topology and routing, P2P systems allow network resources irrationally utilized and P2P traffic always starves the bandwidth of other applications like web traffic.

In fact, many previous research have demonstrated that many of the requested P2P objects were downloaded from peers outside the local network more than locally available. That increases the cost of ISPs therefore why BitTorrent-Like P2P file sharing systems bring so much inter-ISP traffic is that BitTorrent-like P2P file sharing systems always use random methods to choose neighbors and do not concern the impact brought by data transmission between inter-ISP peers which may lead to performance degrading of the whole network. The opposing reaction of ISPs will occur.

Meanwhile, the inter-ISP links are always the network bottleneck of heavy congestion. Consequently, the download rate of P2P systems may be influenced and then the function will be weakened. All of this may result in serious performance degrading of the whole systems. One key aspect of performance in BitTorrent-Like P2P file sharing systems is the download rate that is achieved by participating peers. Then try to find the relation of download speed with various parameters like upload bandwidth, swarm size, start-time, first-chunk-time etc.

There are many solutions such as estimating topology information focusing on predicting network distance in terms of latency, constructing some entities to feed back the underlying topology information to overlay and etc. In this paper, conducted a comprehensive performance comparison the average download time of peers for original BitTorrent algorithm and the Tracker Localized algorithm, We perform event-based simulations to evaluate the performance of our algorithm with BitTorrent protocol. The results also show that our scheme can improve the performance of P2P file sharing systems.

Research Methodology

1. The BitTorrent

BitTorrent (BT) is a very scalable Peer-to-Peer protocol for large scale content-distribution over the Internet. BitTorrent works by splits file into pieces, where users connected to each other directly to upload and download portions of a large file (called as a piece) from other peers who have also downloaded either the file or parts of it and delivering the chunk in a non-sequential manner, the hashes of which are included in a torrent file. All of these torrent files are stored in a server.

BitTorrent differentiates between two types of peers: leeches and seeds. Leeches are peer that only have some or none of the data while seed are peers that have all the data but stay in the system to let other peers download from them. Thus seeds only perform uploading while leeches download pieces that they do not have and upload pieces that they have. A tracker in BitTorrent traces participation peers in a swarm. The downloading process is as follows. Every peer that participates in sharing a file is member of a swarm, which is tracked by a tracker, and multiple swarms associated to a single file can coexist in parallel. The set of all swarms and thus all peers sharing the same file is referred to as a torrent. If a new peer wants to download a file, it first connects with the tracker to get a peer- list which has the file it wants to get. Then the downloader can connect with these peers and download pieces in parallel from the peer in peer list. If the downloader completes the download process it begins to upload the file to other peers for free.

BitTorrent implements a set of algorithms that balances the content distribution load among a swarm of peers and overlay mesh network of peers. Each swarm is managed by a centralized process a tracker. The tracker does not host any content but maintains metadata about it. As leech enter the swarm they first connect to the tracker. The tracker returns a random list of peers that have the content, each leech then randomly selects a subset of that list as its neighbors and initiates requests to set up connections with these neighbors. This mechanism provided faster download whereas in normal P2P systems, the peer downloads from a single peer alone and the download speed is limited.

Instead of downloading directly from the server, each leech requests pieces from all the peer it is connected to. The Tit-for-Tat (TFT) policy (Qiu, et al., 2004) in which downloader give upload preference to peers that provides high download rates, thus creates an incentive for peers to upload their data to other. The downloader periodically updates the connected peers to fine better neighbors. The downloader users a Rarest First policy (Pouwelse, et al., 2005) to determine which piece to request next.

The BitTorrent employs a Tit-for-Tat policy to deter free riding, when the peers behave selfishly and use the swarm only to download pieces without making any contribution to the swarm. The peers exchange data on a Tit-for-Tat policy in the swarm of peers interested in the same file (Vu, et al., 2010), the neighbors of a peer that provide the most data are allowed to request piece in return or we call unchoking. The Tit-for-Tat policy construct thus creates an inventive for peers to upload their data to others. Once a peer has completed to download of a file, it can continue to seed it by using its upload capacity to serve the file to other for free (Liu, et al., 2010).

The piece selection strategy and peer selection strategy are very important in BitTorrent file sharing systems. They can affect the performance of BitTorrent systems enormously. The main piece

selection algorithms of BitTorrent are Strict-priority, Endgame mode, Rarest first and Random first piece (Xia, et al., 2010).

The main peer selection strategy (Legout, et al., 2006) are Tit-for-tat, Optimistic unchoking, Upload only and Anti-snubbing.

2. The Application Layer Traffic Optimization (ALTO)

P2P systems ensure that popular contents are received by peers in the overlay. However, most P2P systems usually employ an arbitrary peer selection policy that ignores the underlying Internet topology, establishing connections between randomly chosen subsets of cooperating peers from around the world. Most distributed hash tables (DHT) use greedy forwarding algorithms to reach their destination, making decisions that may not turn out to be globally optimized. This naturally leads to the ALTO (Seedorf, et al., 2009) problem: how to best provide the topology of the underlying network while at the same time allowing the requesting node to use such information to effectively reach the node on which the content resides. Thus, it would appear that P2P networks with their application layer routing strategies based on overlay topologies are in direct competition against the Internet routing and topology. To solve the ALTO problem is to build network coordinate systems which embed the network topology into a low dimensional coordinate space and enable network distance estimations based on latency (Gurbani, et al., 2009).

3. ALTO for BitTorrent-Like P2P File Sharing Systems

According to choke/unchoking algorithm, peers choose the peers whose download or upload rate is the fastest as their nearest peers in BitTorrent system. However, the peer selection algorithm of BitTorrent is random. So the chance of peer selecting external networks is large. This probably the table is 100 or less than 100, the tracker sends the table to the request peer directly. If the number leads to much unnecessary inter-ISP traffic. To overcome these problems, we introduce the generation of AS topology and the enhance algorithms in the following. The Generation of AS Topology: To make the traffic of BitTorrent stay in the local network by using ALTO approach, we need to adjust some methods in BitTorrent to know some information of underlying topology. We can obtain a peer's AS number through its IP address and some public AS searching institutions. So it's not difficult to know which AS an IP address belongs to. After collecting such information, we know a map table of peers' IP addresses and its corresponding AS numbers. Next we will show how to obtain an integrated AS topologic map through this map table. If all BitTorrent peers belong to N ASes, Select one peer randomly from every AS, then we run traceroute between any two peers to obtain their end-to-end route on IP layer. Next, we find all peers' AS numbers in this route through AS-IP map table. Thus an end-to-end route passes through several ASes linearly. Then treat the consecutive IP addresses in the same AS as an AS node. Thus, we convert IP layer's route to AS's route and then obtain an integrated AS topologic map. At last, with Floyd algorithm, we could get a table with shortest hops of any two ASes and save it in trackers.

4. Tracker Localized Algorithm

In order to make trackers concern the local characteristics, to adjust the random select algorithm of peers. When receive a request from a peer, the tracker sorts AS hops of the request peer to each candidate peers first. The candidate peers usually exist in a table and they are managed by the trackers. To suppose the default number of candidate peers in the table is 100. If the number of candidate peers in is more than 100 and the AS number of the 100th peer is different from the 101st's, the tracker returns

the first 100 peers. If the AS number of the 100th peer is the same as the 101st's, To take down the AS number of the 100th peer and suppose its AS number is n, then remove the peers whose AS numbers are n until the AS number of the 100th peer is different from the 101st's, then the tracker returns the first 100 peers. At last, the request peer asks for files from the peers that the tracker returns directly.

Results Discussion

1. Experimental Setup

This work used the Java-based PeerSim as our simulation platform. PeerSim is a Peer-to-Peer simulator. It has been designed to be both dynamic and scalable. The engines consist of components which may be 'plugged in' and use a simple ASCII file based configuration mechanism which helps reduce the overhead. PeerSim can also work in two different modes: cycle-based or event-based. The cycle-based engine is based on a very simple time scheduling algorithm and is very efficient and scalable. However, it has some limitations. PeerSim can achieve a network consisting of 10^6 nodes using the cycle-based engine. It can also simulate the topology and characteristics of networks precisely.

Firstly, We generate 1024 nodes randomly on the topology that PeerSim simulates, and distribute them in AS0, AS1, AS2, AS3, AS4 AS5 and AS6. The AS topology is shown in Figure 1. 7 of these 1024 nodes are core nodes which only take charge of transmitting data. The bandwidth of links between the core nodes is evenly fluctuant between 100Mbps and 200Mbps. The other 1017 nodes are edge nodes. In these edge nodes, 512 nodes are peers, some of these 512 peers are seeds (the size of the shared file is 50M). Then we let a different number of peers download the file randomly every 10 seconds.

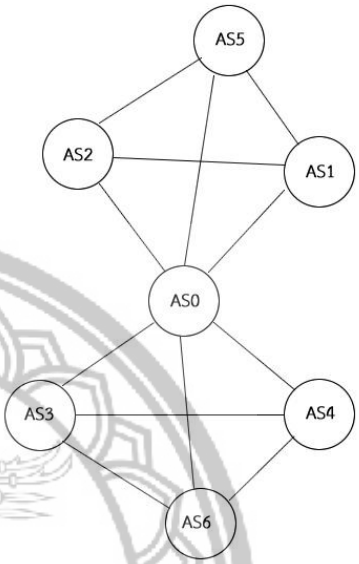

Figure 1 The AS topology in our simulations.

In order to add more authority to our experiment results, we set several scenes including 16 peers, 32 peers, 64 peers, 128 peers, 256 peers, 512 peers and 1024 peers and take more than 5 rounds in every scene. In each round, all conditions including the position of nodes, the bandwidth of links, the delay of links, the seeds and the download events are all generated randomly in some special range except for factors such as the number of nodes and peers, the size of files, the number of edges, the number of ASes and AS topology. Finally, to average the experimental results obtained from the scene which the number of download peers is N and then compare them with that of the original BitTorrent algorithm.

2. Performance that Users Care About

We compare the performance that users care about in this part; these performance metrics include

the average download time of peers and the breaking time of peers. The results are presented as follows.

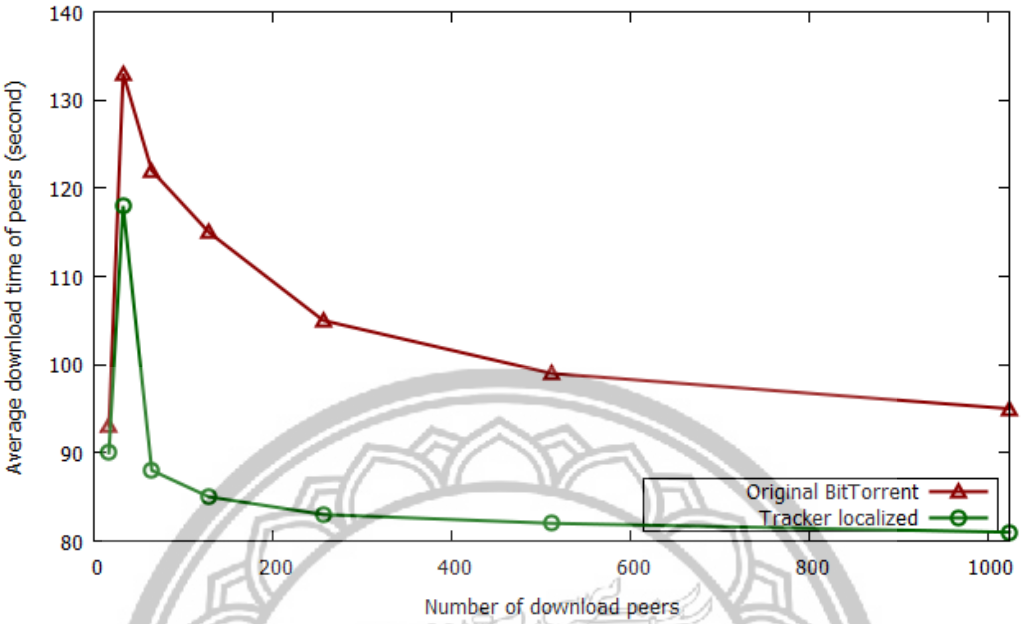

Figure 2 Average download time of peers for Original BitTorrent algorithm and the Tracker Localized algorithm.

Figure 2 shows the average download time of peers for each localized algorithm and the original BitTorrent algorithm. We can conclude that when we use the Tracker localized algorithm, the download time of peers decreases to the largest extent, but when there are few download peers, it doesn't decrease but increase a bit. That is because peers select neighbors randomly to transmit data in original BitTorrent. When there are a few download peers, the bandwidth of inter-AS links is large enough so that the download rate is faster.

However, after the trackers are localized, the peers always select neighbors locally. Because the bandwidth of local links is smaller than that of inter-AS links, the data transmission rate is slower on local links. With the increase of download peers, the peers in the original BitTorrent still select neighbors randomly which may lead to congestion on the inter-AS links and performance degrading of the whole network. The Tracker localized algorithm makes the traffic localized and utilizes bandwidth effectively.

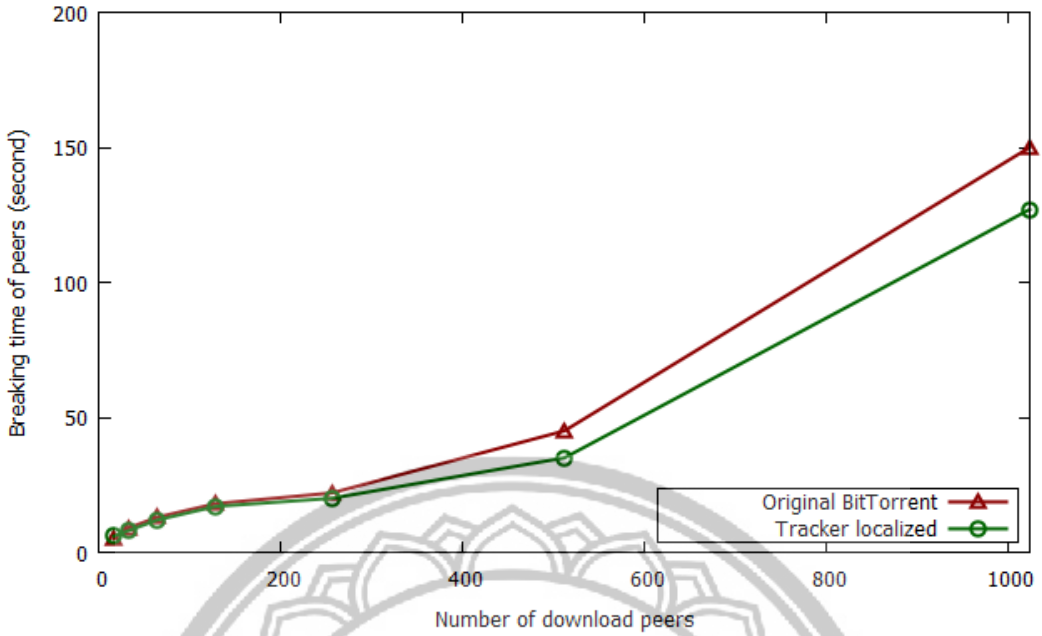

Figure 3 The time of all peers breaking downloading for Original BitTorrent algorithm and the Tracker Localized algorithm.

If peers can't find some pieces of a file during the process of downloading, then the downloading will be in an interrupted state until the peers have received the pieces. Figure 3 shows the time of peers breaking downloading. We can see that the breaking time of peers is the least when using the Tracker localized algorithm.

However, the breaking time is the most when using Tracker localized algorithm. With the increase of download peers, the improvement for breaking time of all localized algorithms is getting more and more obvious.

Conclusions

We studied in this paper the working of BitTorrent Protocol and various mechanisms used to achieve optimal performance by the protocol. In this paper we investigated the impact of number of peers on download rate through PeerSim simulator for The Novel Principle of Application Layer Traffic Optimization By BitTorrent-Like Peer to Peer File Sharing Systems. Firstly, we proposed a novel approach to make BitTorrent node aware of the topology of the underlying networks by modify BitTorrent's original algorithms and replace them with Tracker Localized Algorithms based on autonomous system (AS) hops. After that to conduct a comprehensive performance comparison the average download time of peers for original BitTorrent algorithm and the Tracker Localized algorithm, Then to conducted a comprehensive performance comparison the time of all peers breaking downloading for original BitTorrent algorithm and the Tracker Localized algorithm based on the PeerSim simulation. The simulation result shows that, with our scheme the nodes in BitTorrent-like P2P systems have better sense of the topology of their underlying networks and can interact more efficiently.

In order to perfect our scheme, we still need to do a lot of work in the future. For instance, we can add some other strategies to our scheme, we can consider the file fragmentation, we can predict the users next tendency for what kind of resources they will be

interested by analyzing their behaviors and so on. All of this needs us to do deeper research.

References

Bharambe, A. R., Herley, C., & Padmanabhan, V. N. (2005). *Analyzing and improving bittorrent performance.* Retrieved February, 4, 2014, from http://scholar.google.co.th/scholar?q=Analyzing+and+improving+bittorrent++performance&btnG=&hl=th&as_sdt=0%2C5

Cohen, B. (2003, June). Incentives build robustness in BitTorrent. *Incentives build robustness in BitTorrent.* Retrieved February, 4, 2014, from http://scholar.google.co.th/scholar?q=Incentives+build++robustness+in+BitTorrent&btnG=&hl=th&as_sdt=0%2C5

Gurbani, V. K., Hilt, V., Rimac, I., Tomsu, M., & Marocco, E. (2009). A survey of research on the application-layer traffic optimization problem and the need for layer cooperation. *Communications Magazine, IEEE, 47*(8), 107-112.

Legout, A., Urvoy-Keller, G., & Michiardi, P. (2006, October). Rarest first and choke algorithms are enough. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (pp. 203-216). N.P.: ACM.

Liu, J., Wang, H., & Xu, K. (2010). Understanding peer distribution in the global internet. *Network, IEEE, 24*(4), 40-44.

Pouwelse, J., Garbacki, P., Epema, D., & Sips, H. (2005). The bittorrent p2p file-sharing system: Measurements and analysis. In *Peer-to-Peer Systems IV* (pp. 205-216). Berlin: Springer Berlin Heidelberg.

Qiu, D., & Srikant, R. (2004, August). Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review* (Vol. 34, No. 4, pp. 367-378). N.P.: ACM.

Seedorf, J., Kiesel, S., & Stiemerling, M. (2009, September). Traffic localization for P2P-applications: the ALTO approach. In *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on* (pp. 171-177). N.P.: n.p.

Urvoy-Keller, G., & Michiardi, P. (2006, April). Impact of inner parameters and overlay structure on the performance of bittorrent. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings* (pp. 1-6). N.P.: n.p.

Vu, L., Gupta, I., Nahrstedt, K., & Liang, J. (2010). Understanding overlay characteristics of a large-scale peer-to-peer IPTV system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), 6*(4), 31.

Xia, R. L., & Muppala, J. K. (2010). A survey of BitTorrent performance. *Communications Surveys & Tutorials, IEEE, 12*(2), 140-158.