2020

Reducing BitTorrent Download Time via Handshake-Based Switching

Elliotte Kim

Nova Southeastern University, elliotte_kim@hotmail.com

Reducing BitTorrent Download Time via Handshake-Based Switching

by

Elliotte Kim

A dissertation submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy in Computer Science

College of Computing and Engineering
Nova Southeastern University

February 14, 2020

We hereby certify that this dissertation, submitted by Elliotte Kim conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Greg Simco

Gregory E. Simco, Ph.D.
Chairperson of Dissertation Committee

4/6/2020
Date

Francisco J. Mitropoulos, Ph.D.
Dissertation Committee Member

4/6/2020
Date

Sumitra Mukherjee

Sumitra Mukherjee, Ph.D.
Dissertation Committee Member

4/6/2020
Date

Approved:

Meline Kevorkian

Meline Kevorkian, Ed.D.
Dean, College of Computing and Engineering

4/6/2020
Date

College of Computing and Engineering
Nova Southeastern University

2020

An Abstract for a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Reducing BitTorrent Download Time via Handshake-Based Switching

by
Elliotte Kim
February 2020

Peer-to-peer networking overcomes the single point of failure and bandwidth limitations inherent to the centralized server model of file-sharing. It is both a popular means of sharing digital content and a major consumer of internet traffic, with BitTorrent being the most-used protocol. As such, significant research has gone into improving peer-to-peer performance in order to reduce both download times and networking costs. One aspect that can affect performance is the client's selection of peers to download from, as the time spent downloading from even a single poor-performing peer can impact the overall download duration.

A recent peer selection strategy explored having a client use historical knowledge acquired through third-party sources, as well as its own first-hand experience with previously visited peers, as a means of selecting likely good-performers, coupled with a peer switching strategy that replaced peers whose post-selection downloads exhibited poor performance contrary to what historical knowledge suggested in order to limit the time spent downloading from said poor-performers Though this tactic demonstrated reduced download times compared to various past works, it still suffered from poor peer selection due to its historical knowledge not necessarily reflecting the current state of the peers.

This work introduced and examined an enhancement to this hybrid peer selection and switching strategy by adding current intelligence regarding a peer's available bandwidth, all the while avoiding the additional network costs associated with performing on-the-fly probing or querying techniques utilized by other peer selection strategies to benchmark prospective peers. With such on-the-fly knowledge about a peer's current bandwidth availability, this new enhanced strategy quickly replaced poor performers without waiting for downloads to be performed and subsequently benchmarked, resulting in reduced overall peer-to-peer download times.

The results of adding this pre-download peer switching enhancement demonstrated improved download performance, particularly in early file transfer runs. However, as more runs occurred and the benefits of the original strategy's historical knowledge became more pronounced, the time savings gained from this new enhancement diminished.

Acknowledgements

First, I would like to thank all my professors at Nova Southeastern University, whose classes helped expand my range of knowledge and improve my research and writing skills. In particular, I wish to thank:

- Dr. Gregory Simco, my supervisor, for all the guidance, feedback, and advice you have provided me regarding this dissertation. The idea for my research was conceived from one of your class assignments. I learned so much from you, and that experience helped make the rest of my doctoral studies so much easier.
- Dr. Francisco Mitropoulos and Dr. Sumitra Mukherjee, for always teaching your classes with such gusto and for serving on my dissertation committee. Your input has been invaluable to me.

Next, there are a number of people from the Massachusetts Institute of Technology that helped start me on this journey that I must thank:

- Myron "Fletch" Freeman, Dr. Duane Boning, and Dr. Eric Grimson, for providing me with that initial opportunity at EECS. You made my dream of MIT possible and gave me the flexibility to pursue it. I will forever be grateful to you all.
- Dr. Jim Bales, for always being available to give me advice and encouragement. Thank you for continuing to be a sounding board for me, including on this work, well after I was your student.
- Ayida Mthembu and Donna Friedman, for being the best deans I could ever ask for. You always looked out for my well-being and made sure I not only survived, but also flourished in my academic pursuits.
- Dr. Megumi Ando, for going above and beyond any other TA I have ever met. I could not have survived MIT without you. You inspired me to pursue a doctorate, and continue to inspire me to this day.

Finally, I would like to thank my friends and family for all their love and support over the years. None of this would have been possible without them.

Table of Contents

List of Figures

Figures

List of Tables

Tables

Chapter 1

Introduction

Reducing the impact of poor-performing peers can result in faster peer-to-peer file downloads (Ren, Liu, Zhou, Tang, Ci & Wang, 2013). The recently developed Hays & Simco (2017) hybrid peer selection and peer switching strategy demonstrated shorter download times compared to other prior works by coupling advanced knowledge, used to avoid selecting potential poor-performers, with a choke, used to limit the impact of actual poor-performers selected.

This work details the creation and subsequent testing of a new peer-switching enhancement that, when coupled with Hays & Simco's strategy, further reduced overall BitTorrent download times. It accomplished this by providing the client with on-the-fly bandwidth performance data about peers, without incurring the additional network overhead typically inherent to the gathering of such information for other peer selection (Hsiao et al., 2011; Li, 2012; Ying & Basu, 2006) or peer switching strategies (Chiu & Eun, 2008; Lehrfeld & Simco, 2010). In so doing, the added download efficiency achieved from this work can save more time for consumers and cut more costs for both consumers and ISPs alike.

The remainder of this chapter provides background regarding this paper's research, as well as describes the problems that this work attempted to solve and the issues that it had to contend with. Chapter 2 provides a review of literature that covers Hays & Simco's work, as well as other relevant prior techniques researched. Chapter 3 describes the premise and methodology this research used for augmenting Hays &

Simco's strategy. It also outlines the simulation environment used to conduct experiments, as well as success criteria. Chapter 4 provides the test results that both validated the simulation environment and affirmed this work's goal of having its new enhanced strategy achieve reduced overall download times compared to Hays & Simco's original approach. Based on these observed results, Chapter 5 details the conclusions drawn from this paper's research, including the strengths and weaknesses of the enhanced Hays & Simco strategy. Chapter 5 also discusses what other peer-to-peer strategies could benefit more from this work's enhancement than Hays & Simco's hybrid strategy.

Background

With the scalability, redundancy, and failover that it provided, peer-to-peer networking became a popular distributed application architecture, used for such purposes as file-sharing, instant messaging, content delivery, and even digital crypto-currencies. A significant portion of internet traffic has been attributed to peer-to-peer communications, with file-sharing via the BitTorrent protocol making up a majority of this activity (Schulze & Mochalski, 2009).

Unlike the traditional client-server networking model, which consists of clients communicating solely with a central server, peer-to-peer networking is comprised of peers that can share their resources directly with each other. A peer can act as both client and server simultaneously, requesting resources from others while offering out what resources it has at the same time (He, Dong, Zhao, Wang & Qiang, 2016).

Peer-to-peer has 2 major advantages over traditional client-server: scalability and robustness. In the client-server model, adding more clients causes resource contention by

dividing up the server's network bandwidth, thus making each client's download time longer. But peer-to-peer networking can overcome said bottleneck, having the client download different pieces of the desired fire from various peers simultaneously and assemble them together. This allows a client to aggregate, to the limits of its own download capabilities, the service capacities of the various peers it communicates with (Lua, Crowcroft, Pias, Sharma & Lim, 2005), rather than strictly compete with others clients for a single server's resources (Chiu, 2010). As such, so long as the client has not saturated it download bandwidth, adding more peers that can contribute their respective upload bandwidths in order to utilize the client's available download bandwidth can actually improve file transfer performance (Qiu & Srikant, 2004).

For example, in the traditional client-server model where the client has a 50 Mbps connection and the server has a 20 Mbps connection, at best the download rate the client can achieve would be 20 Mbps, less so if said server is sharing its bandwidth with other clients. However, in the peer-to-peer model, a client could, for example, download from 100 peers simultaneously, each with a 1 Mbps connection. Even if only half of each peer's bandwidth is available, this would allow the client to achieve an aggregate download rate of 50 Mbps and take full advantage of its download bandwidth.

Peer-to-peer network performance, typically measured from the client's perspective by how long it takes to download a file, is impacted by various factors, including the service capacities of both client and peer's respective internet connections, the number of network hops between client and peer, physical distance, contention for a peer's bandwidth by other clients, even peer-to-peer specific bandwidth throttling enforced by either the ISP or the peer itself.

A peer suffering from a slow internet connection, high latency, or heavy network contention may be a poor-performer to download from (Xie, Yang, Krishnamurthy, Liu & Silberschatz, 2008). Unfortunately, peer-to-peer systems are prone to poor peer selection, since they operate at the application layer (Hays & Simco, 2017), constructing overlaying network topology instead of having the underlying network topology information available. Poor peer selection can make peer-to-peer networks less efficient (Magharei, Rejaie, Rimac, Hilt & Hofmann, 2014), hampering clients with longer download times (Ren, Liu, Zhou, Tang, Ci & Wang, 2013). As such, significant research has gone into trying to reduce the impact of poor-performing peers and improve peer selection.

Some performance-improving techniques researched, described in the next chapter, probed peers to gauge their data transfer performance or performed queries that measure the latency between client and peer in order to prune potential poor-performers that demonstrated low available bandwidth or high network latency (Li, 2012; Hsiao, Hsu & Miao, 2011), Such techniques helped clients make better-informed decisions regarding which peers to select, though at the cost of introducing additional network overhead in performing such on-the-fly probes or queries on top of the time spent establishing a peer-to-peer connection and performing the download. As such, on-the-fly intelligence-gathering approaches involved a trade-off, ideally cutting more download time by providing better-performing peers than the cost added in collecting said information.

The work of Hays and Simco (2017) explored using prior knowledge for selecting peers. In this strategy, historical service capacity and network locality data, collected in advance and stored locally to the client so as to avoid the additional on-the-fly network

overhead associated with peer-probes or infrastructure-queries that could eat into performance, were used to make better-informed decisions in selecting peers, while a peer switching technique was employed to limit the impact of poor-performers that got through peer-selection by replacing them after observing their download performance firsthand.

Problem Statement

Hays and Simco's advanced knowledge-based peer selection strategy attempted to predict the service capacity and locality of peers by using historical information collected in advance and stored locally to the client. However, this approach still suffered from poor peer selection caused by incomplete or out-of-date information that no longer reflected the current state of a peer's performance, resulting in longer download times.

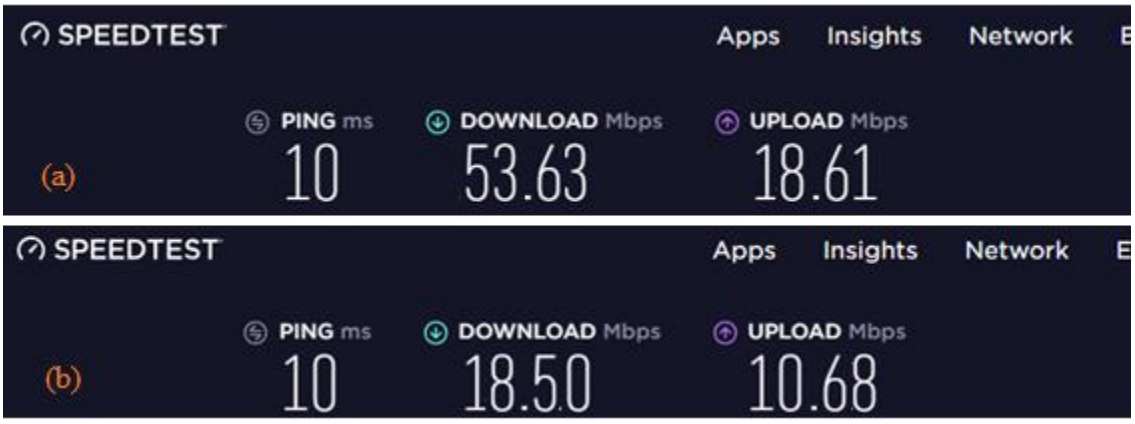

Figure 1: (a) Example of SpeedTest detecting upload speed of almost 20 Mbps.

(b) Almost 11 Mbps upload from same PC with BitTorrent running.

The upload speed data gathered by OOKLA's web-based SpeedTest utility did not reflect any peer-to-peer specific bandwidth throttling limits that were configured on a peer itself. For example, in Figure 1a, the home cable modem connection for a user

supported upload speeds of about 20 Mbps, as measured by SpeedTest. However, as illustrated in Figure 2, the user could have configured his or her Vuze BitTorrent application to limit uploads to only 50 Kbps. In this example, the 20 Mbps service capacity as detected by SpeedTest could have led advanced knowledge peer selection to conclude that this node was likely a good candidate, whereas in reality the 50 Kbps throttling limit set on the BitTorrent application might have made this one a poor choice.

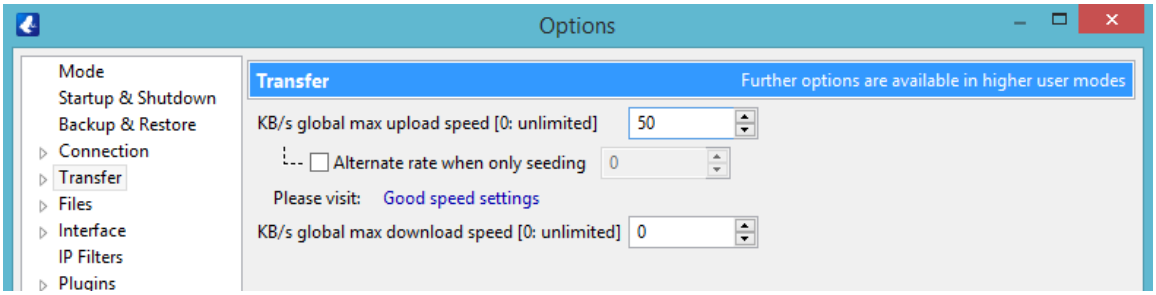

Figure 2: Example of PC from Figure 1 with 50 Kbps BitTorrent upload throttle.

Since a peer's available bandwidth depends on the utilization demands of all the other applications that share its network connection, changes in these demands may not be reflected in historical knowledge. For example, SpeedTest data regarding a node's bandwidth capacity could have been gathered when there were no other networking demands made on it (see Figure 1a), yet at the moment a client wanted to select said node as a peer for downloading, it could have already been uploading to multiple other BitTorrent clients (see Figure 1b). Even a peer with high service capacity when under no load could end up being a poor-performer if it was already under heavy network contention (Bolliger, Gross, & Hengartner, 1999).

Chunk-based peer switching, discussed in the next chapter, helped reduce the impact of poor-performing peers by limiting the amount of time a client spent with any

poor-performing peers that might have been picked by whatever peer-selection strategy was leveraged, rather than having the client maintain slow download connections with poor-performers until the entire file was transferred (Lehrfeld & Simco, 2010). Chunk-based switching built on top of time-based switching's practice of replacing selected peer every 5 minutes regardless of performance by adding bandwidth benchmarking and replacement of selected peers that did not meet some threshold transfer rate that occurred every minute. As such, choke-based switching retained the benefits of time-based switching, but achieved performance gains by potentially identifying selected poor-performing peers and replacing them after only 1 to 4 minutes instead of replacing all peers after 5 minutes (Chiu & Eun, 2008; Lehrfeld & Simco, 2010).

However, peer-switching strategies still entailed potentially spending time downloading from poor-performers prior to their replacement. Furthermore, it was possible for a replacement peer to be a worse performer than the one switched out, as this depended on the peer-selection strategy used to select said replacement. In the case of Hays & Simco's approach (2017), given that the historical knowledge it drew upon for initial peer selection was, as mentioned above, susceptible to out-of-date or incorrect intelligence, so too was subsequent peer selection invoked in replacing poor-performers. For example, a prospective peer's network load could have increased due to higher demand from other clients, or its throttling settings could have been enabled or changed. As such, under Hays & Simco, a poor-performing peer could become switched out for a false positive, a peer that historical knowledge suggested was a high-capacity peer, but was currently a poor-performer.

Furthermore, since the SpeedTest data did not reflect a peer's current bandwidth utilization or throttling settings, Hays and Simco's approach (2017) suffered from some of its initial selection and replacement peers being unwitting poor-performers, with realized transfer rates differing significantly from what prior knowledge suggested. In such a scenario, while prior knowledge was useful in identifying past poor performers to consider avoiding, the list of peers initially thought to be good performers could have actually contained a number of false positives.

Research Goal

This research set out to improve upon the hybrid advanced knowledge-based peer selection and choke-based peer-switching strategy researched by Hays and Simco (2017). Like the original, this new approach collected and locally-stored prior knowledge about peer service capacity and network locality from such historical data sources as OOKLA's SpeedTest and MaxMind's GeoLite ISP-mapping databases. This intelligence was gathered in order to reduce and sort the list of available peers down to a ranked list of potential good performers for selection, establish connections to and download from selected peers, and update its prior service capacity knowledge with observed download performance for subsequent re-ranking. Periodic replacement of the worst-performing peers with potentially better ones was also conducted (see Figure 3 below).

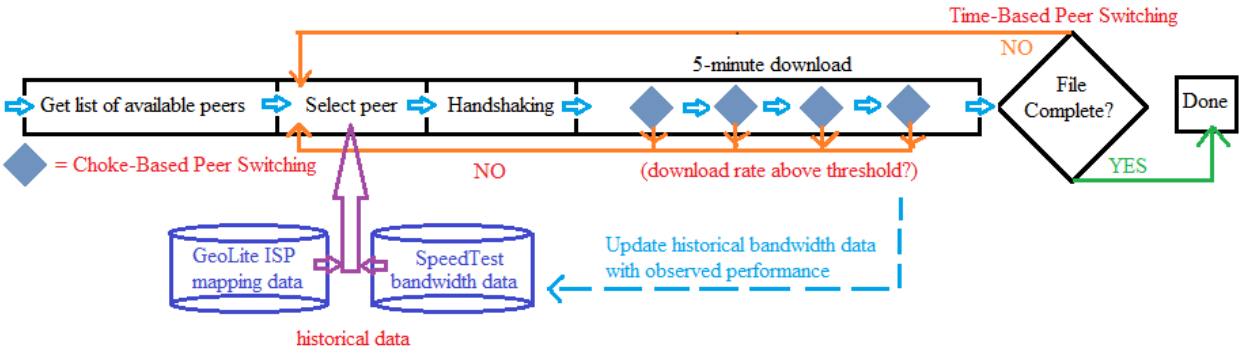

Figure 3: Lifecycle of a peer-to-peer connection using Hays & Simco's strategy.

However, unlike the original, this new strategy incorporated on-the-fly performance information about candidate peers to further prune those deemed likely poor-performers. As such, it gained the benefits of peer-probing, accounting for such peer service capacity factors as current network load and bandwidth throttling limits that would otherwise not have been accounted for by advanced knowledge historical data alone.

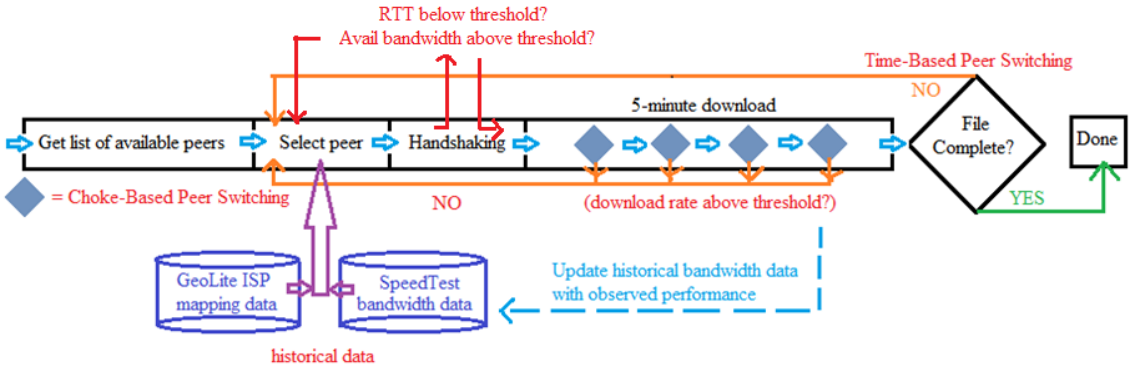

Figure 4: Lifecycle of a peer-to-peer connection using proposed strategy.

Furthermore, as shown in Figure 4, said information was provided by the peer to the client during the connection-establishment or handshaking phase of the client-peer connection lifecycle, rather than after the connection was already established and some amount of downloading was performed, as was the case with both peer-switching strategies and bandwidth-benchmarking peer-probing strategies.

In keeping with the goals of the original work, this research avoided adding additional network overhead in gathering the on-the-fly performance information by leveraging BitTorrent network activity already inherent to Hays and Simco's approach.

By adding the benefits of peer-probing to Hays and Simco's strategy at an earlier stage in the peer connection lifecycle, this research aimed to further reduce the peer-to-peer download times yielded by its predecessor by facilitating the pruning of poor-performers without waiting for downloads to be subsequently performed and benchmarked.

Relevance and Significance

Taking up over 20% of the world's network utilization (Bindal, Cao, Chan, Medved, Suwala, Bates & Zhang, 2006), BitTorrent became a leading consumer of internet traffic (Schulze et al., 2009). With people sharing files that easily hit gigabyte sizes and BitTorrent having such a large audience ("BitTorrent and µTorrent", 2012), improving peer selection could make downloads more efficient by having clients connect to peers that would reduce cross-ISP traffic and provide higher data transfer rates, thus reducing costs for ISPs while saving time for consumers.

Peer probing-based selection likely provided the most accurate information about a peer's immediate performance. By establishing a download connection and transferring some probing data from the peer, whether it was generic benchmarking packets whose intervening time gaps were measured (Hsiao et al., 2011) or a chunk of the desired file itself whose download completion time was denoted, (Li, 2012), the client could gauge the peer's current transfer rate. This measurement specified the probe's immediate

download performance, which reflected ISP line capacity, distance and latency, utilization, and throttling. However, since a probe was itself a data download, its completion time was also subject to the transfer performance between client and peer. As such, probing a poor-performing peer to gauge its download performance would take longer to complete than for a high-performing one.

Hays & Simco's (2017) advanced knowledge peer selection strategy avoided adding on-the-fly network overhead by relying on locally-stored historical information. While such data could account for locality and past service capacity, it failed to consider such transient factors as current peer network contention, where the peer could, for example, have much more of its present upload capacity allocated to other clients compared to when the historical data from SpeedTest's assessment or the client's previous observation was recorded, and user-configured throttling, where SpeedTest's benchmarking was not hampered by bandwidth-limits set on BitTorrent-specific network traffic, which could result in poor peer selection.

By taking advantage of current peer download performance information provided during the handshake phase of the connection lifecycle, the client did not need to wait for downloads to be performed in order to benchmark and subsequently prune poor-performing peers, resulting in faster downloads

Barriers and Issues

Knowledge acquisition about peers has typically incurred a cost, generally in the form of additional network overhead. This overhead could take the form of pings, traceroutes, peers probes, and network infrastructure lookups. Despite providing

intelligence that could improve peer selection and decrease download times, such overhead could cut into some of this time savings.

Hays & Simco's strategy addressed this overhead issue by moving the intelligence-gathering cost from when a peer-to-peer download actually occurred to a point in time beforehand and stored this information locally. However, mistakes could arise as such advanced knowledge failed to account for the here and now.

As Hays & Simco (2017) noted, it was difficult, if not impossible, to test peer selection strategies in the real world. Either a BitTorrent application needed to be modified or a new one created that utilized the proposed strategy. It then needed to be deployed to multiple computers across the internet. Files-sharing performance needed to be tested, yet the system had to be isolated to avoid contamination of experiments from the outside. A consistent environment was important for scientific testing and evaluation, making the internet a difficult setting to conduct experiments in (Hays & Simco, 2017).

As such, a simulation environment was needed, capable of representing peers with diverse conditions, such as varying ISP service capacities, network utilization loads, and possible network throttling.

Summary

This research set out to improve upon the hybrid advanced knowledge-based peer selection and choke-based peer switching strategy developed by Hays & Simco (2017), with the end goal of further reduced download times in peer-to-peer networks. It accomplished this by pruning selected poor-performing peers before downloading even commenced. By taking advantage of network activity that was already inherent in

BitTorrent using the original Hays & Simco strategy in order to gain on-the-fly peer performance information, this research's enhanced version avoided overhead costs that could have otherwise cut into any time savings gained.

Chapter 2

Brief Review of the Literature

Introduction

Since 1999, when Napster first popularized their use, peer-to-peer networks have become one of the most prominent ways of sharing content across the internet (Grizzard et al., 2007), accounting for 43% to 70% of world-wide network traffic, depending on the location. Of this activity, BitTorrent has been the most popular protocol (Schulze et al., 2009), ranging between 15 million and 27 million active nodes daily (Wang & Kangasharju, 2013) and over 150 million users each month ("BitTorrent and μTorrent", 2012). Even Microsoft integrated into Windows 10 a peer-to-peer client for receiving and sharing operating system updates and software patches (Newman, 2015), thus increasing the use of peer-to-peer technology by another 500 million active devices (Bott, 2017).

Considering the prevalence of peer-to-peer applications, it is no surprise that much research has gone into improving and mitigating peer selection, as even a single peer could influence the overall performance of a file-share, ultimately impacting the client's time spent waiting for a download to complete (Ren et al., 2013).

Random-Based Peer Selection

In random peer selection, the client would indiscriminately choose a peer to connect to from a list of those available that host the desired file, handshake with said chosen peer to establish a connection, and download the file, as depicted in Figure 5.

Since this approach was robust and simple to implement (Traverso et al, 2015), it became the most commonly-used selection strategy (Sherman, Nieh & Sten, 2009).

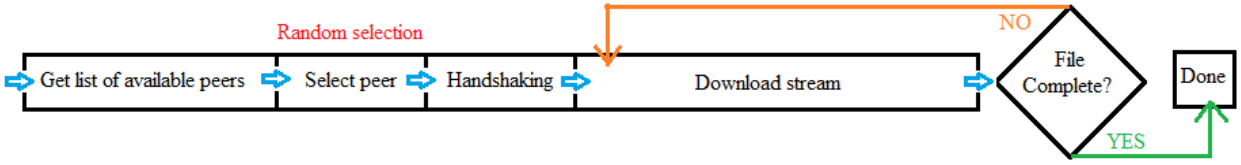

Figure 5: Lifecycle of a peer-to-peer connection using random-based peer selection

However, since the random selection strategy did not do anything to reduce the chance of picking poor-performing peers, using said approach often resulted in slow downloads. Because of this, most peer-to-peer applications using random selection improved on their performance by complementing the strategy with such techniques as parallel downloading or peer-switching in order to reduce the time spent downloading from poor-performing peers (Magharei, Rejaie, Rimac, Hilt & Hofmann, 2014).

Parallel Downloads

Parallel downloading entailed the client connecting to multiple peers simultaneously in order to download a file, thereby aggregating the collective available upload bandwidths of each connected peer to the limit of what the client's own download bandwidth was able to accommodate. Said file was divided into chunks, one equally-sized chunk per connected peer.

For example, in the case of five connected peers each chosen by random peer selection, a 1000 MB file was divided into five 200 MB chunks that the client downloaded from each peer simultaneously. As such, how long the overall download took to complete was determined by the time spent retrieving a chunk from the slowest connected peer (Chiu & Eun, 2008).

Peer-Switching Strategies

Peer switching, on the other hand, involved the replacement of a peer after it was already selected, a connection was established, and some amount of file transfer was performed. The selection of a replacement peer generally called for the same strategy used in the initial peer selection to be leveraged again.

Furthermore, peer switching strategies could be performed in conjunction with parallel downloading. As such, a client could, for example, initially have selected 10 peers to download from simultaneously, and replaced one or more of those peers along the way, depending on criteria that was set by the peer switching strategy used.

Random Chunk-Based Switching

Like parallel downloading, the random chunk based-switching strategy divided a file into equally-sized chunks. In this case, however, numerous small chunks were produced. The client downloaded each chunk individually and sequentially, with each chunk assigned to a randomly-selected peer (Chiu & Eun, 2008). As illustrated in Figure 6, the cycle of peer selection, handshaking, chunk downloading, and peer switching was repeated until the entire file was downloaded.

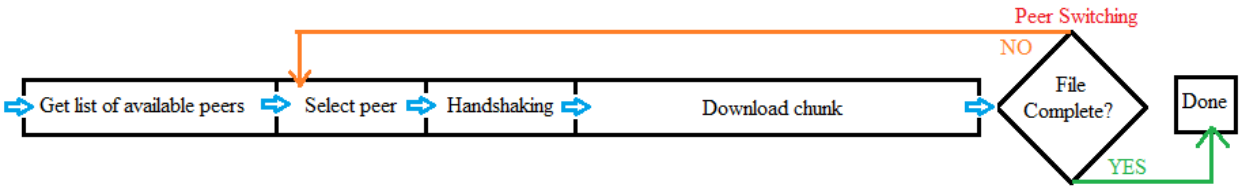

Figure 6: Lifecycle of a peer-to-peer connection using chunk-based peer switching

Since chunks were intentionally small in size, the time spent with a particular peer was limited, hopefully keeping the impact of poor-performing peers to a minimum.

However, a particularly poor-performing peer could still result in a chunk taking a long time to download (Chiu & Eun, 2008).

Time-Based Switching

Time-based switching used the time spent downloading from a selected peer rather than the completion of a chunk download as the basis for a peer switching strategy (Chiu & Eun, 2008). As show in Figure 7, the client selected a peer to download from and established a download connection. After retrieving as much of the desired file as possible within a five minute span of time, the client dropped the peer and selected a new one to continue downloading from. This process was repeated until the entire file was retrieved.

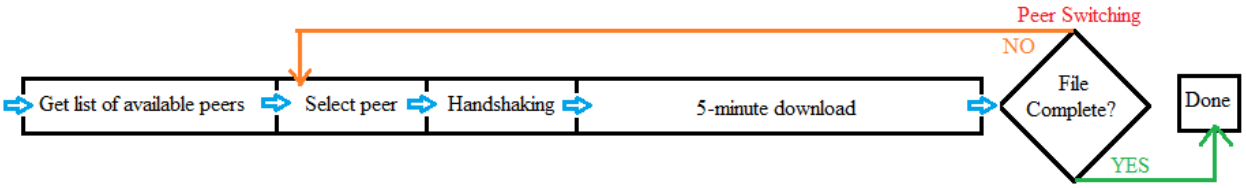

Figure 7: Lifecycle of a peer-to-peer connection using time-based peer switching

Compared to chunk-based switching, time-based switching reduces the effect of a poor-performing peer by limiting its impact to a 5 minute download window instead of waiting for a chunk to complete, which could take longer. As such, time-based switching has demonstrated shorter download times (Chiu & Eun, 2008). However, it is still possible for a selected peer to transfer very little data during its allotted five minute connection.

Choke-Based Switching

Choke-based switching (see Figure 8 below) took the time-based switching strategy a step further by allowing peer connections to be dropped ahead of their allocated durations. The realized rate of transfer between client and peer was compared to some calculated choking threshold value. Every minute, download rates were checked. Peers that did not meet said threshold were subsequently dropped and replaced with newly-selected ones (Lehrfeld & Simco, 2010).

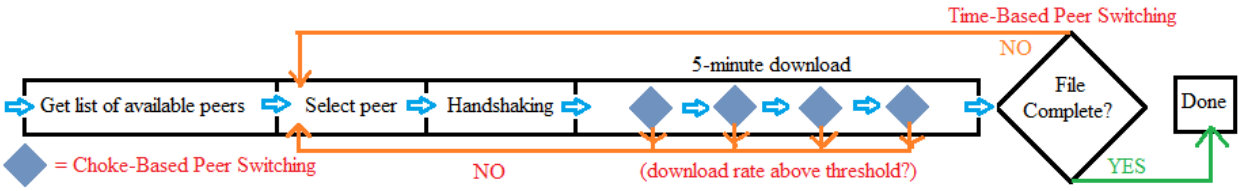

Figure 8: Lifecycle of a peer-to-peer connection using choke-based peer switching

By further reducing the time spent with poor-performing peers, the time-based switching strategy augmented with choke-based switching demonstrated faster downloads than time-based switching alone (Lehrfeld & Simco, 2010). A slow peer's individual impact could be confined to a single minute rather than five, though the selection strategy used could still end up replacing a peer with another poor-performer.

Other Peer Selection Strategies

Various types of peer selection strategies have been developed, intent on outperforming the download performance of random selection. Parallel downloading became a mainstay for any peer-to-peer application. Some selection strategies benefitted from peer-switching, either by working in conjunction with a replacement strategy or by incorporating some aspect of peer-switching into their peer selection behavior. Still other

selection strategies relied on using historical knowledge or other intelligence-gathering methodologies in order to weed out poor-performing peers or limit their impact.

Peer Probing-Based Selection

Selection strategies that performed bandwidth-benchmarking downloads or probes on candidate peers for on-the-fly performance information could help clients make better informed decisions regarding which peers to share from, resulting in faster downloads.

HP Algorithm

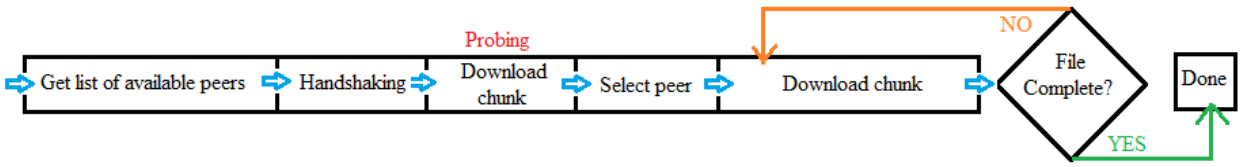

Figure 9: Lifecycle of a peer-to-peer connection using HP probe-based peer selection

The High-capacity Peer (HP) algorithm was an example of a peer probing-based peer strategy that combined peer selection with a form of chunk-based switching (Li, 2012). HP probed available peers to determine their respective throughputs. It accomplished this by establishing a download connection and requesting a file chunk from each available peer in order to gauge transfer performance. Once these probes were completed and high-capacity peers were identified, the remaining chunks were downloaded from the selected good-performing peers while poor-performing peers were pruned, as shown in Figure 9.

HP did not guarantee reduced download times compared to random selection. Case in point, a file could be so small that there are as many available peers as there are

chunks, resulting in all file chunks being received during the probing process (Li, 2012). HP was arguably more a peer switching algorithm rather than a peer selecting one, since a peer was dropped from further use only after a chunk of the desired file had already been downloaded from it. No peer pruning was conducted prior to initiating chunk downloads from each peer due to probes. Additionally, on its own, HP did not account for any drop in a selected peer's download performance that could occur after the probing phase was completed, meaning that a client could have end up spending much of its time downloading from poor-performing peers without a switching strategy accompanying it.

Query-Based Selection

Various selection strategies leveraged on-the-fly queries to peers or networking infrastructure in order to ascertain such download performance characteristics as locality and latency for selecting peers.

Ying & Basu's Traceroute-Based Strategy

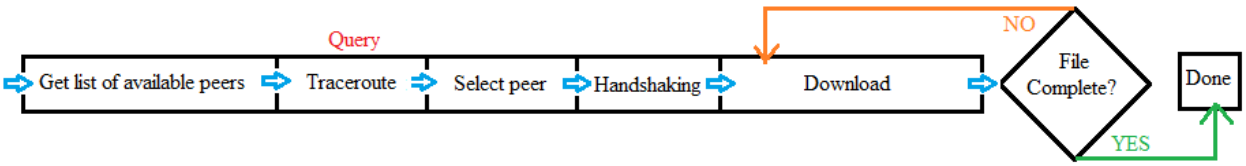

Figure 10: Lifecycle of a peer-to-peer connection using Ying & Basu's peer selection

Researchers from the University of Alberta proposed a query-based peer selection algorithm that leveraged traceroute, a popular tool used to map out network topology, detect and diagnose routing problems, and describe the path, the number of hops, and the round-trip time (RTT) for each hop between two devices on the internet (Mao, Rexford, Wang & Katz, 2003). In this approach, the client coordinated with a tracker to obtain a list of peers to connect from. Using traceroute information collected from every peer,

candidates with RTTs and hop counts larger than some maximal threshold value were removed from the list (Ying & Basu, 2006).

This traceroute query phase helped the selection phase of the lifecycle (see Figure 10) prune peers that had high latency or were located far away from the client prior to establishing download connections with them. However, this strategy could still select poor-performing local, low-latency peers with low bandwidth availability, as well as introduce additional network overhead in conducting traceroute queries to candidate peers, which could become appreciable when the list of available peers was large.

Historical Knowledge-Based Selection

While probe and query-based peer selection strategies attempted to gain on-the-fly knowledge about peers in order to weed out and replace selected poor-performers, some strategies leveraged prior knowledge in order to eliminate peers before any connection attempt to them was even initiated.

Varvello and Steiner's Traffic-Localizing Selection

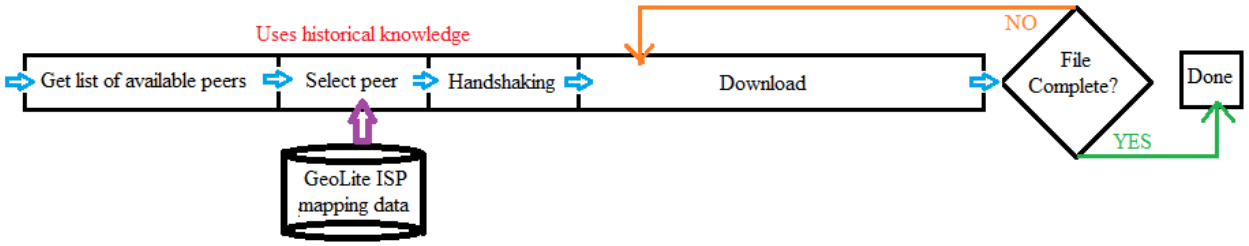

Figure 11: Lifecycle of a peer-to-peer connection using Varvello & Steiner's peer selection

Bell Labs' Varvello and Steiner (2011) explored a historical-knowledge-based traffic-localizing strategy using MaxMind's GeoLite database for peer selection (see Figure 11). The benefit of having the client identify and download from localized peers

(peers that share the same ISP as the client) was that there was no cross-ISP traffic involved in their interaction, which generally resulted in fewer network hops, less latency, and avoidance of inter-ISP throttling (Pacifici, Lehrieder & Dán, 2016). As such, localized peers were likely better performers for the client and a cheaper cost to the ISPs.

In their approach, a single client created 256 distinct logical entities that joined a peer-to-peer network. Called sybils, these entities were assigned node IDs close in proximity to the desired file's info hash. By taking advantage of BitTorrent's distributed hash table (DHT) network topology, sybils were always made aware of those peers hosting the file, as well as any requesting it. Using the GeoLite data to look up ISPs from IP addresses, sybils could choose to work only with localized peer-sets. If too few existed, external peers could also be included (Varvello & Steiner, 2011).

While the prior knowledge used by this strategy could ensure that localized peers were prioritized for selection by the client, it did not guarantee that selected peers were good-performers. On its own, the strategy did not account for the latency or available upload bandwidth of those selected peers.

Hybrid Strategies

Some peer selection strategies utilized multiple features from the approaches mentioned earlier, combining their respective benefits to improve download performance.

Adaptive and Efficient Peer Selection

Adaptive and Efficient Peer Selection (AEPS) used a query-rank-then-probe model (Hsiao et al., 2011). The client sent a query to each of the candidate peers and

waited for their reply, measuring the round-trip time for each. Peers were rank based on those results, giving priority to candidates with the smallest RTT values. Priority candidates were then probed for their available bandwidth (ABW) verification by having each send some number of generic data packets to the client. The client measured the time gaps between the probe packets it received from a peer and used that to calculate its available bandwidth. AEPS selected those peers that meet a certain threshold (see Figure 12).

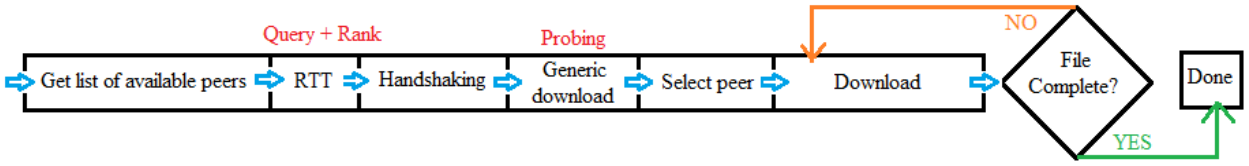

Figure 12: Lifecycle of a peer-to-peer connection using AEPS

Though AEPS outperformed methods that were solely random-based, RTT-based, or ABW-based (Hsiao et al., 2011), it did introduce additional network overhead by measuring RTT from a large list of initial peer candidates and by performing ABW verification probes. Since every available peer was queried for their RTT in order to prune candidates, a candidate-rich environment resulted in the client sending messages to and waiting for responses from numerous peers. Furthermore, ABW verification probes could run slowly, particularly when gauging peers with good round-trip times but little available bandwidth throughput. And, like the HP Algorithm, on its own, AEPS did not account for changes in download performance. Should a selected peer's transfer rate decline after the probing phase, the client could end up being stuck connected to poor-performer.

Hays and Simco's Peer Selection

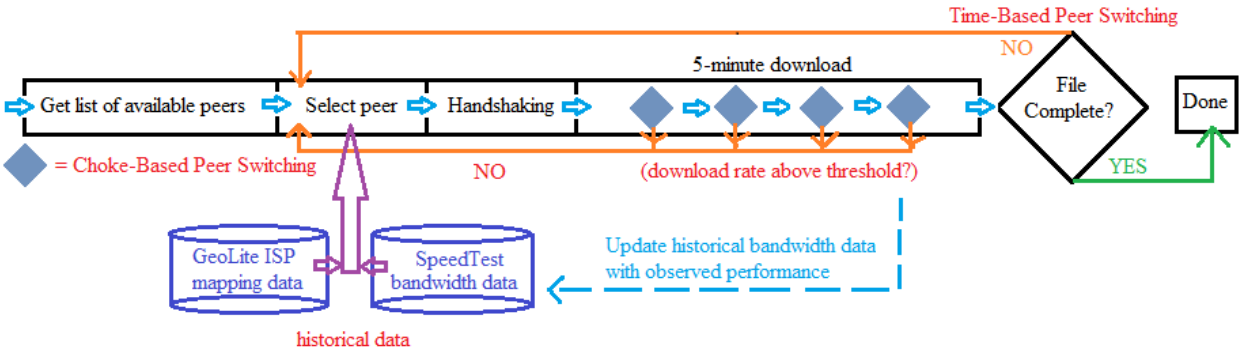

Figure 13: Lifecycle of a peer-to-peer connection using Hays & Simco's strategy.

Like Varvello & Steiner's historical knowledge-based strategy, the research by Hays and Simco (2017) used MaxMind's GeoLite database, as well as OOKLA's Speedtest data, thus gaining locality and bandwidth service capacity information in order to make informed peer selection decisions. Such knowledge was acquired ahead of time and stored locally, thereby avoiding additional on-the-fly network overhead associated with queries or probes being introduced during the actual peer selection or file download processes.

Using the GeoLite data, the algorithm divided the list of available peers into 2 smaller lists, one containing those peers sharing the same ISP locality as the client, the other containing the rest (Figure 14, steps 5 & 6). The peers on both lists were sorted based on historic service capacity, either from the initial OOKLA data acquired or from first-hand downloading experience subsequently observed and saved by the client (Figure 14, step 7).

```
Beforehand…
Step 1:   If prior peer-to-peer downloads were performed
          then      go to Step 4
          else      go to Step 2
Step 2:   Retrieve GeoLite and SpeedTest data and store them locally
Step 3:   Set peer's historical performance value to its SpeedTest value
When ready to start peer-to-peer downloading…
Step 4:   Retrieve list of available peers hosting desired file
Step 5:   Use GeoLite data to determine the ISP of available peers from their IP addresses
Step 6:   If available peer shares same ISP as client
          then      add to List A
          else      add to List B
Step 7:   Sort peers in List A and in List B by their SpeedTest value in descending order
Let a = size of List A, b = size of List B, and n = number of allowed parallel downloads
Step 8:   If n <= a
          then      selected peers = first n peers in List A
          else      selected peers = all the peers in List A plus the first (n – a) peers in List B
Let counter c = 0, incrementing by 1 every second
Let z = 300, representing the 5 minutes allocated for time-based peer switching in seconds
Let threshold = choke-based switching threshold
Step 9:   Handshake with selected peers to establish download connection
Step 10: Start downloading a file chunk from each selected peer
Step 11: While c < z
Step 11a:         If c is a multiple of 60 and peer's observed performance < threshold
                  then      update peer's historical performance value with its observed performance
                            re-sort List A and List B rankings
                            switch peer with highest ranked peer available
                            handshake with replacement peer to establish download connection
                            download remaining part of file chunk from replacement peer
Step 11b:         If peer's file chunk finished downloading
                  then      update peer's historical performance value with its observed performance
                            re-sort List A and List B rankings
                            switch peer with highest ranked peer available
                            handshake with replacement peer to establish download connection
                            retrieve another chunk from the highest ranked peer available (can be same peer)
          End while
Step 12: Update selected peers' historical performance with their observed performances
Step 13: Re-sort List A and List B rankings
Step 14: Replace selected peers with the n highest ranked peers available (can be same peers)
Step 15: If all file chunks are completely downloaded
          then      DONE
          else      go to Step 8
```

Figure 14: Hays & Simco's Advanced Knowledge-Based Peer Selection Strategy

Peer selection first pulled from the shared-ISP list. If said list had fewer peers than the number of parallel downloads allowed, the other list was also utilized (Figure 14, step 8). The performance of parallel downloads from selected peers was monitored, the

locally stored service capacity information about those peers was revised (Figure 14, step 12), and both list rankings were updated (Figure 14, step 13). Time-based and choke-based peer switching were employed to prune worst performers in hopes of finding better performing replacements (see Figure 13). As such, this made Hays & Simco's approach an integrated peer selection, peer replacement, and parallel download strategy.

Both Hays & Simco's (2017) and Varvello & Steiner's (2011) approaches drew on prior knowledge in order to identify peers that were located in the same ISP as the client. Fortunately, the mapping between IP address and ISP rarely changed over time, making the knowledge gained from MaxMind data consistently useful in referencing peers.

The peer's current available service capacity, on the other hand, could change from one moment to the next as demand on the peer's resources by other clients changed. Prior knowledge did not reflect a peer's current bandwidth utilization. A peer capable of 20 Mbps uploads could experience heavy load, thus having little of its bandwidth available for a new client. Even a peer with a high SpeedTest transfer rate could end up being a poor-performer if it was experiencing heavy network contention.

Furthermore, bandwidth rates recorded by SpeedTest did not reflect throttle settings configured on a peer, since the web-based data transferred between a PC and SpeedTest to gauge performance would not restricted by a throttle setting configured on that PC's BitTorrent application. For example, a peer's network connection could have been rated at 20 Mbps uploads according to SpeedTest. However, current throttle settings could have limited BitTorrent uploads to just 50 Kbps. In this scenario, the 20 Mbps

value reported by SpeedTest could have resulted in advanced knowledge-based peer selection to incorrectly presume that a peer was a good performer, when in fact the 50 Kbps throttle setting could have made it a poor one

As such, Hays & Simco's strategy, in using past historical bandwidth availability data for peer selection, lacked current, up-to-date service capacity information that on-the-fly probing techniques could provide to help prune poor performing peer. Instead, it relied on choke-based peer switching's one minute benchmarking interval to deal with poor-performers that were selected.

Summary

This literature review denoted several different strategies developed to improve the download performance of peer-to-peer networks above that of using mere random selection, with each approach having its own set of advantages and disadvantages.

Some selection strategies performed benchmarking download probes to assess a peer's current bandwidth availability (Li, 2012), while others relied on queries or past knowledge to make assumptions about a peer's current desirability (Ying & Basu, 2006; Varvello & Steiner, 2011). Switching strategies, on the other hand, attempted to reduce the impact of poor-performing peers that made it through the selection process by curtailing the amount of time spent downloading from them (Chiu & Eun, 2008; Lehrfeld & Simco, 2010). And hybrid strategies emerged that combined aspects of other strategies in order to further reduce download times (Hsiao et al., 2011; Hays & Simco, 2017).

However, despite these improvements over random selection, lack of accurate, up-to-date bandwidth information have led selection strategies that rely on queries or

historical knowledge to inadvertently choose poor-performers, while probing and peer-switching techniques have themselves been fettered by slow peers.

The new strategy described in this research built on the work of Hays & Simco (2017), adding on-the-fly intelligence regarding a peer's current bandwidth availability in order to facilitate the replacement of poor-performers without waiting for the requisite downloads inherent to conventional peer-switching techniques, and without introducing a separate query or probe phase to the connection lifecycle.

Chapter 3

Methodology

Introduction

The goal of this research was to introduce a new strategy that could complement other strategies to further reduce the average peer-to-peer download completion time of a file for a BitTorrent client. To accomplish this goal, this research endeavored to improve upon the approach of Hays & Simco (2017).

Hays & Simco's (2017) work was chosen as a basis to complement as it was a recently-developed strategy that already demonstrated reduced download times over prior works by taking advantage of historical knowledge-based selection to both localize peers and rank them based on past performance, and choke-based switching to limit the amount of time spent with a selected poor-performing peer. As such, being able to further improve upon Hays & Simco's (2017) download performance was a challenge that, if this research's strategy was successful, would not only make the Hays & Simco's (2017) work even more cost-efficient for ISPs and convenient for consumers, but could also result in similar or even greater reductions in download times when complemented with less-advanced strategies.

The remainder of this chapter makes note of the premise upon which this paper's research was based upon, discusses how on-the-fly bandwidth availability intelligence was gained and leveraged to quickly replace peers, how its integration into the hybrid work of Hays & Simco (2017) produced a faster-downloading strategy, how a simulation

environment was used to conduct the study, and how data produced in said experiment was analyzed and download performance was measured.

Premise

In reviewing the literature, this paper observed that having peer-switching strategies further limit the amount of time a client spent downloading from a newly selected poor-performing peer (going from the time spent downloading a complete chunk, as in the case for chunk-based switching, down to a 5 minute interval, as in the case for time-based switching (Chiu & Eun, 2008), and likewise from a 5 minute interval down to a 1 minute benchmarking interval in going from time-based to choke-based switching strategies (Chiu & Eun, 2008; Lehrfeld & Simco, 2010)) resulted in further reduced overall download completion times.

As such, this research operated on the premise that overall download completion times could be further reduced if the length of time the client spent downloading from newly selected poor-performing peers could be cut down to zero. In other words, if poor-performing peers that made it through the selection phase could be quickly replaced without waiting for some amount of downloading from them to occur (as in the case with past peer-switching strategies (Chiu & Eun, 2008; Lehrfeld & Simco, 2010)), then this could further reduce the negative impact of poor-performing peers on BitTorrent peer-to-peer downloads and improve the overall download performance experienced by the client.

If this research was to be capable of immediately replacing newly-selected peers that were poor-performers without waiting for benchmarking downloads to be performed,

as in the case of choke-based switching (Lehrfeld & Simco, 2010), then this new strategy needed to provide the client with on-the-fly intelligence regarding a selected peer's current available bandwidth in order to be able to make such determinations, which past strategies were only able to typically garner through probing downloads (Li, 2012; Hsiao et al., 2011) or by making less accurate assumptions about through historical knowledge (Varvello & Steiner, 2011) or round-trip latency measurement (Ying & Basu, 2006; Hsiao et al., 2011).

However, this paper's research introduced a new form of peer-switching that gained probe-like intelligence regarding a peer's currently available bandwidth without the inclusion of a separate benchmarking download phase, provided the client with said intelligence in a query-like manner from the peer without the addition of separate query networking traffic to BitTorrent normal functionality, and allowed the client to replace selected poor-performing peers prior to ever downloading from them, thereby further limiting interactions with poor performers and reducing overall download times.

Handshake-Based Peer Switching

To facilitate the acquisition of on-the-fly intelligence regarding a peer's available bandwidth, which could have been be used by a BitTorrent client to assess and replace newly-selected peers that were poor-performers prior to downloading from them, this paper's research introduced the concept of handshake-based switching.

If a normal BitTorrent client wanted to download a file from a peer, a TCP connection would first be established. Via this connection, the client would send a handshake message to the peer, which would then reply with its own handshake message

(Erman, Ilie & Popescu, 2005). If the client received this response and the peer ID and info hash embedded within the handshake message were what the client expected, then the connection would stay open and the actual file download would begin. Otherwise, the client would drop the connection ("Bittorrent Protocol Specification", 2017).

Note that the BitTorrent handshake message, besides containing a peer id and info hash, has 8 reserved and unused bytes, all set to zero after the fixed headers ("The BitTorrent Protocol", 2017). Bram Cohen, designer of BitTorrent, indicated that these reserved bytes could be used to change the protocol's behavior ("Bittorrent Protocol Specification", 2017).

As such, this paper's research took advantage of these reserved and unused bytes by having peers encode their currently available bandwidth estimate into their handshake message response to the client. This allowed the client to gauge a newly-selected client's available bandwidth without needing to wait for some amount of benchmarking download to be performed between the client and peer, as in the case with probes or choke-based switching, and immediately replaced poor-performers that did not meet a chosen bandwidth threshold (see Figure 15).

Additionally, since this intelligence regarding a peer's available bandwidth was encoded into the handshake message, network traffic which was already inherent to both standard BitTorrent and BitTorrent using Hays & Simco's (2017) hybrid strategy, this new enhancement did not introduce additional wait time for the client that would otherwise have been created by the addition of a separate peer-querying or peer-probing phase in order to gain information regarding a peer's available bandwidth.

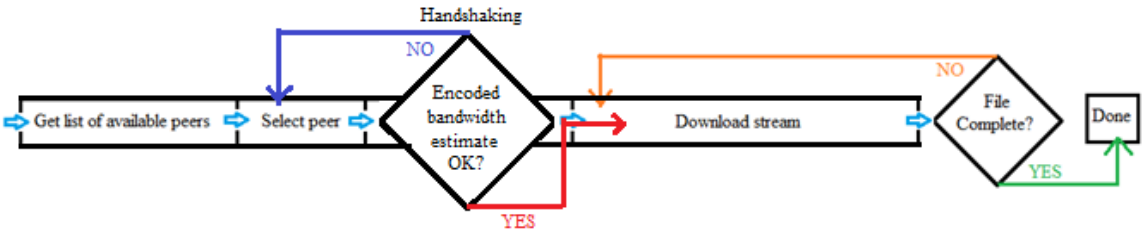

Figure 15: Lifecycle of a peer-to-peer connection using Handshake-based peer switching

A modified BitTorrent application was needed in order for a peer to estimate its current bandwidth availability to be able to send to the client via its handshaking message response.

On a peer, this modified application performed network benchmarking at startup to measure its maximum realized upload throughput, prior to accepting request from others and thus before other nodes began contending for its resources. Such an upload benchmarking feature already existed in Vuze, a popular BitTorrent application, which served as a similar basis for this paper's research. Vuze's benchmarking test entailed the client interacting with auto-speed servers to gauge the median transmission rate of a torrent file over the course of 20 seconds after a few seconds of ramp up to full speed, with results that varied from the actual rates by about 10 percent ("Speed Test FAQ", 2010).

This startup measurement was then used as the basis for the peer's maximum service capacity value. Then, by monitoring its current bandwidth usage during BitTorrent file-sharing, the application running on a peer estimated its available service capacity on-the-fly by subtracting its current upload utilization from its maximum throughput value. The peer could update its maximum service capacity value on occasion, such as when its current file-shares were uploading at a higher rate than said

value, or by performing additional benchmarks when utilization appeared to be low. The peer's modified BitTorrent application was also tasked with encoding its current available bandwidth estimate into its handshaking message response to any clients attempting to initiate a peer-to-peer connection with it. This maximum service capacity value was utilized and updated over the course of a BitTorrent application instance's entire runtime, providing available bandwidth estimates in servicing multiple different torrents either simultaneously or in tandem.

On a client, the modified application was tasked with checking the aforementioned reserved bytes for an estimated available bandwidth encoding within handshake message responses from peers and immediately terminated connections with those peers whose estimates fell below a chosen threshold in order to prevent their respective content downloads from initiating. As such, the client then leveraged whatever peer-selection strategy it relied on in order to replace those dropped peers until it had achieved its configured simultaneous number of peers setting value. Similarly, the client also took advantage of any other peer-switching strategy or strategies to address slowdowns in peer performance that occurred after content downloads had begun.

Since this handshake-based peer-switching strategy eliminated a poor-performing peer immediately before content transfer began, it applied not only to initial peer selection, but also to any subsequent selection of peers that took place because of peer-switching that was invoked by this handshake-based strategy or any other peer-switching strategy it was complemented with.

Another benefit of embedding estimated available bandwidth into the BitTorrent handshake message response was that modified applications would still work normally with standard BitTorrent. A client running regular BitTorrent could still download from a peer running a modified version, as the client could simply disregard the peer's bandwidth embedding in its normally reserved and unused bytes of the handshake message response, thus interacting with the peer per the standard BitTorrent protocol. Similarly, a modified client could still download from a standard peer by not utilizing handshake-based switching when the bandwidth estimate embedding was absent.

Handshake Switching-Enhanced Hays & Simco

While the handshake-based approach described above could merely be complemented with Hays & Simco's hybrid strategy to perform faster peer-switching of newly-selected peers, it should be reiterated that Hays & Simco's (2017) strategy also took advantage of locally-stored bandwidth data, collected either from SpeedTest or subsequently updated through first-hand download performance observations, to help rank peers for possible selection. As such, in order to gain the most benefit from this research's handshake-based peer-switching strategy, the handshake-embedded bandwidth estimates were also integrated with Hays & Simco's strategy to update its locally-stored historical data, so that not only was peer-switching performance improved, but also the peer-selection process as well.

As in Hays & Simco's (2017) strategy, MaxMind data was used by this handshake-enhanced integration to separate the available peers list into 2 sub-lists, those that shared the same ISP as the client and those that did not. Both sub-lists were then

initially sorted and ranked based on the OOKLA SpeedTest data. Once an initial set of peers was selected, the client went through the process of BitTorrent handshaking in order to establish connections with those selected peers.

Similar to how choke-based switching replaced connected peers not meeting some calculated threshold, handshake-based switching also replaced poor-performers. But, while a client with choke spent time slowly downloading from a poor performing peer before switching, a minute in the case of Hays and Simco's experiments (Hays & Simco, 2017), handshake-based selection immediately replaced a poor-performing peer that made it through the selection process based on its provided bandwidth availability estimate and immediately updated the client's locally-stored historical knowledge regarding a peer's service capacity accordingly, as shown in Figure 16, so that those poor-performers could be re-sorted for future selection. If this updating of the locally-stored data with the handshake-encoded bandwidth estimates was not performed, then this would have ran the risk of having the client immediately reselecting the poor-performing peer since its ranking would have been unchanged.

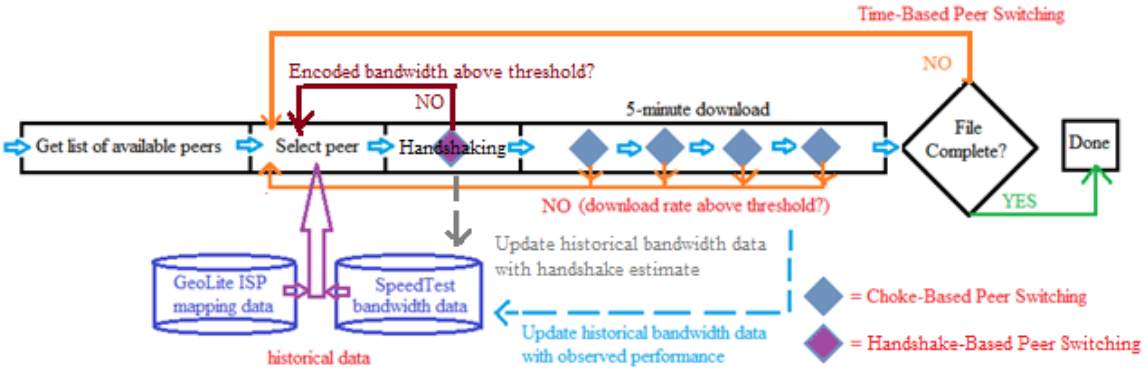

Figure 16: Lifecycle of Handshake-Enhanced Hays & Simco peer-to-peer connection.

```
Beforehand…
Step 1:   If prior peer-to-peer downloads were performed
          then      go to Step 4
          else      go to Step 2
Step 2:   Retrieve GeoLite and SpeedTest data and store them locally
Step 3:   Set peer's historical performance value to its SpeedTest value
When ready to start peer-to-peer downloading…
Step 4:   Retrieve list of available peers hosting desired file
Step 5:   Use GeoLite data to determine the ISP of available peers from their IP addresses
Step 6:   If available peer shares same ISP as client
          then      add to List A
          else      add to List B
Step 7:   Sort peers in List A and in List B by their SpeedTest value in descending order
Let a = size of List A, b = size of List B, and n = number of allowed parallel downloads
Step 8:   If n <= a
          then      selected peers = first n peers in List A
          else      selected peers = all the peers in List A plus the first (n – a) peers in List B
Let counter c = 0, incrementing by 1 every second
Let z = 300, representing the 5 minutes allocated for time-based peer switching in seconds
Let threshold = choke-based switching threshold
Step 9:   Handshake with selected peers to establish download connection
Step 9a:  If peer's handshake-embedded estimated performance < threshold
          then      update peer's historical performance value with its handshake-embedded estimate
                    re-sort List A and List B rankings
                    switch peer with highest ranked peer available
                    handshake with selected peer
                    repeat Step 9a
          else      go to Step 10
Step 10:  Start downloading a file chunk from each selected peer
Step 11:  While c < z
Step 11a:           If c is a multiple of 60 and peer's observed performance < threshold
                    then      update peer's historical performance value with its observed performance
                              re-sort List A and List B rankings
                              switch peer with highest ranked peer available
                              handshake with replacement peer to establish download connection
                              download remaining part of file chunk from replacement peer
Step 11b:           If peer's file chunk finished downloading
                    then      update peer's historical performance value with its observed performance
                              re-sort List A and List B rankings
                              switch peer with highest ranked peer available
                              handshake with replacement peer to establish download connection
                              retrieve another chunk from the highest ranked peer available (can be same peer)
          End while
Step 12:  Update selected peers' historical performance with their observed performances
Step 13:  Re-sort List A and List B rankings
Step 14:  Replace selected peers with the n highest ranked peers available (can be same peers)
Step 15:  If all file chunks are completely downloaded
          then      DONE
          else      go to Step 8
```

Figure 17: Handshake-enhanced Hays & Simco Strategy

All other aspects of the proposed strategy mirrored Hays & Simco's original work (see Figure 17). Both utilized parallel downloads. To address drops in transfer performance after handshaking had been performed and downloading had commenced, time-based peer switching, running at 5 minute intervals, replaced the worst performing peer connection with the highest ranked available peer. Hays & Simco's research tested the inclusion and exclusion of choke-based switching running at 1 minute intervals, so this paper did so as well for comparison. The client's locally-stored prior knowledge was updated with the observed transfer rates from selected peers that made it past handshake-based switching, and used for subsequent peer selections.

Simulation Environment

Real-world testing of peer-to-peer strategies was difficult given the logistical challenges and uncontrollable nature of the internet (Hays & Simco, 2017). Indeed, various prior works, including those of Chiu & Eun (2008), Lehrfeld & Simco (2010) and Hays & Simco (2017) all utilized simulation environments in order to assess their respective strategies.

In order to evaluate the addition of handshake-based peer switching in reducing download times, particularly of those achieved by Hays & Simco's hybrid strategy, a simulation environment was created that accounted for such file transfer performance factors as locality, ISP service capacity, bandwidth throttling, and network utilization. The simulation environment tested and compared the overall BitTorrent download performance when the following strategies were utilized: random selection with time-based switching, random selection with choke-based switching, Hays & Simco's advanced knowledge-based selection with time-based switching, Hays & Simco with

choke-based switching, and handshake-enhanced Hays & Simco with choke-based switching.

Simulated Download

Like Hays & Simco's (2017) single client experiments, this research simulated a BitTorrent client downloading a 150 MB file from a population of peers. For this implementation, the client allowed simultaneous parallel downloads from up to 4 selected peers. Additionally, the 150 MB file was divided into 20 evenly-sized chunks of 7.5 MB each. As such, at the start of a simulation run, before any downloading had commenced, the client had 20 inactive and incomplete (empty, in this case) chunks it needed to populate.

After a simulation run had commenced, as many as 4 chunks were actively downloading at any particular moment in time, with each chunk keeping track of which selected peer it had been assigned to, what the file transfer rate of its assigned peer was, and based on said rate and the amount of time spent transferring from the peer, how much of its 7.5 MB capacity had been filled.

To accommodate time and choke-based peer-switching strategies, a chunk did not have to be completed by a single peer. Rather, while a chunk could only be active with and assigned to one peer at a time, it could end up being active and inactive multiple times, accruing its content across multiple peers until completion. As such, each chunk maintained Boolean flags that denoted if it was active, in order to prevent accidental assignment to more than 1 peer at a time, and if it was complete, so as to prevent completed chunks from occupying one of the 4 selected peers. Thus, at the end of a

simulation run, all 20 chunks were in an inactive and complete state, representing a completed 150 MB file download with no active downloads from selected peers occurring.

Simulated Peers

At the start of the experiment, a population of 1000 peers were generated, of which only a maximum of 4 were selected by the client and downloaded from at any particular point in time. For ease of identification and tracking purposes, each peer was assigned a unique ID. In order to represent the various real world factors that could impact a peer's download performance, each peer was assigned a variety of attributes, including locality, maximum network upload capacity, and availability.

In order to accommodate Hays & Simco's (2017) strategy, which divides the list of available peers into 2 smaller lists based on ISP locality, each peer was randomly assigned a locality attribute value, designating whether it shared the same ISP locality as the client or not. Whereas Hays & Simco's experiments randomly assigned IP addresses to simulated peers, which were then subsequently looked up to determine respective ISP associations, this simulation environment simplified the process by skipping IP address assignment and instead assigned ISP locality directly. Such a simplification was viable since ISP lookup and localization of peers in Hays & Simco's strategy were advanced knowledge tasks performed well ahead of the peer-to-peer selection process and whose time spent were not factored into the download performance measurement of the strategy.

Each peer was randomly assigned a maximum capacity value, which represented what the peer's maximum upload rate was, not factoring in any throttling or network

contention caused by other nodes that attempted to download from it. Since throttling could curtail how much of a peer's maximum upload rate could actually be leveraged in a real-world scenario, a Boolean throttling enabled attribute was also randomly assigned, designating whether a peer was throttled. Similarly, a throttling capacity value was randomly assigned to each throttled peer, denoting said peer's maximum upload bandwidth capacity.

Since a peer on the internet could use some of its upload capacity for sharing with other clients, a percent available attribute was randomly assigned to each peer, signifying what percentage of its upload capacity was not being used. As such, the amount of upload bandwidth available for a peer was calculated by multiplying the percentage available attribute with the peer's upload capacity (either the maximum capacity or the throttling capacity if throttling was enabled). For convenience, this calculated value was also stored into a peer's actual performance attribute.

Each peer also had a known performance attribute associated with it. This was used to represent the locally-stored historical knowledge leveraged by the Hays & Simco (2017) strategy that was either initially based on SpeedTest data or subsequently from observed download performance measurement. This known performance attribute was then used by the simulation environment as a basis for re-sorting the 2 ranked peer sub-lists (those that were ISP local and those that were not) for subsequent peer selection.

However, for handshake-enhanced Hays & Simco simulations, selected peers that were pruned because they had available bandwidth estimates that fell below the threshold did not get the opportunity to have their download connection performance observed by

the client for subsequent updating of the locally-stored historical knowledge as per original Hays & Simco (2017). Therefore, the available bandwidth estimate embedded in a peer's handshake message response was also used to update the local-stored historical knowledge for subsequent peer ranking and selection, so as to prevent a pruned peer from being immediately re-selected.

Simulated Client

The simulated BitTorrent client was tasked with dividing the list of available peers based on their ISP locality into 2 smaller lists and then sorted said lists based on their known performance attribute, as per the original Hays & Simco (2017) strategy. The client selected 4 of the available peers to download from, choosing the highest ranked peers from the list of local peers first.

Since all BitTorrent connection attempts entailed a handshaking exchange to be performed between client and peer, whether or not available bandwidth estimates were embedded into the handshake message (Erman, Ilie & Popescu, 2005), this experiment arbitrarily assumed every newly selected peer incurred a 1 second overhead, to account for the handshake protocol's round-trip time cost.

Given that simulations running a strategy complemented with handshake-based peer-switching could result in more frequent peer replacements compared to a strategy without handshake-based switching, thus incurring more handshaking overhead costs, this research acknowledged that using such handshaking overhead cost in the experiment's overall download time calculations could put the new handshake-based switching strategy at a disadvantage. However, this research took the position that it was

better to underestimate the download speed improvement of the handshake-enhanced Hays & Simco strategy, rather than overestimate its performance benefits.

At one second intervals, the client was tasked with updating the active chunks' respective download progress status, based on their associated selected peer's actual upload rate. This was performed for each active chunk until its associated selected peer was dropped due to either time-based or choke-based peer switching, in which case the chunk was marked as inactive and kept in its incomplete state, or when the chunk in question was fully downloaded, at which point the chunk was denoted as being both inactive and complete.

Simulation Trials

This research defined a trial as a set of 10 simulated file download executions performed in tandem. Each run, like in Hays & Simco's research, simulated the peer-to-peer downloading of a 150 MB file, divided into 20 equally-sized 7.5 MB chunks, with each chunk keeping track of how much of its 7.5 MB content had been transferred.

At the start of a trial, a list of 1000 peers was randomly generated, with each peer being assigned an ISP locality, a maximum service capacity ranging from 25 Kbps to 375 Kbps that was used to represent the peer's initial known capacity value, whether bandwidth throttling was currently being used, and if so, what the bandwidth throttle was set to, and how much of the bandwidth capacity was currently available.

In the first simulated one-second interval of an execution run using random selection, the available peer list was shuffled, whereas it was sorted when using the other

selection strategies. Four available file chunks were then selected to simulate parallel downloads and assigned to the top four peers from the list.

For the handshake-based switching strategy, the handshake threshold was set to 200 Kbps, the midpoint of the maximum service capacity range. If a selected peer's estimated available bandwidth exceeded this handshake threshold, its chunk remained assigned to it. Otherwise, the chunk was unassigned from its peer, the peer's known capacity was updated with its estimated availability, and the peer list was re-sorted.

Each chunk assigned to a selected peer incremented its download progress by how much data would be transferred in 1 second based on said peer's available bandwidth value. When the chunk's progress hit 7.5 MB, the chunk was marked as complete, no longer available nor active, and unassigned from its peer.

In each subsequent simulated one-second interval, when there were fewer than 4 chunks actively downloading and other unfinished chunks remained, incomplete chunks were assigned to available peers from the top of the list until 4 chunks were actively downloading again. For the proposed strategy, handshake threshold checking was performed on new connections. Chunk progress was incremented and monitored for download completion.

The above simulated one-second interval process was repeated until all the chunks completed downloading, with a few caveats. At simulated 300-second intervals, time-based switching of the worst performing peer selected was performed, with the peer's known capacity value updated with its observed download performance.

At simulated 60-second intervals for those strategies using choke-based switching, selected peers performing below the 200 Kbps threshold had their chunks unassigned from them and their known available capacity updated with their observed download performance. Said underperforming peers were then subsequently replaced with the highest ranked available candidates.

In between each of the 10 runs within a trial, the 20 file chunks were reset, but the available peer list was not. This was performed in order to represent locally-stored prior knowledge persisting between executions for both Hays & Simco's approach and the new strategy. However, the available bandwidth value for each peer in the list was given a new value, reflecting transient changes to a peer's network contention and utilization.

Performance Evaluation

At the end of a simulated run, the total number of 1-second intervals required by the tested strategy to have all 20 chunks marked as both inactive and complete were measured. This represented the overall download time it took for BitTorrent to transfer the 150 MB file. 10 simulation runs were performed one after the other to form a single trial, so that later simulation runs in the trial using either Hays & Simco (2017) or handshake-enhanced Hays & Simco could benefit from knowledge gained in earlier runs within said trial.

The simulation code printed to the console the observed download times for each of the 10 runs for a simulated trial, which were subsequently recorded. Each simulated

trial was performed 100 times, in order to average out any random fluctuations observed and to calculate the expected download time for each of the 10 simulation runs.

Simulation Validation

In order to validate the simulation environment created for this paper's research, the cost-averaged download times for each simulation run for strategies using random selection with time-based switching, Hays & Simco with time-based switching, and Hays & Simco with choke-based switching, were examined.

Cost-averaged download times for random selection with time-based switching were considered to be consistent with Hays & Simco's (2017) own reported results if said times performed consistently across all simulation runs. In other words, this research expected that a graph charting average download times vs. simulation runs generally remained flat, not trending upward or downward over the course of each simulated run interval.

Thanks to its use of prior knowledge for peer selection, Hays & Simco's (2017) strategy using time-based switching was expected to exhibit decreasing average download times across successive simulation runs in order to be consistent with reported results and validate this portion of the simulation environment.

Likewise, Hays & Simco with choke-based switching was also expected to trend towards decreasing download times. However, said average download times were also generally expected to be smaller than those of Hays & Simco employing time-based switching at any particular simulation run, due to the use of choke.

Results

In order to help determine if this research's proposed handshake enhancement could improve upon Hays & Simco's download performance, the aforementioned download time averages for each simulated run were graphed. Said graph plotted average download time vs. simulation run for Hays & Simco's strategy with time-based switching, Hays & Simco's strategy with choke-based switching, and handshake-enhanced Hays & Simco with choke-based switching.

This research considered the enhancement strategy a success if the cost-averaged download times observed for handshake-enhanced Hays & Simco with choke-based switching were smaller than those of Hays & Simco's original strategy with choke-based switching at each simulated run interval.

Conclusion

Through the use of the aforementioned simulation environment running executions of both Hays & Simco's original advanced knowledge-based peer selection strategy and the handshake-enhanced version, this research hoped to see the new approach demonstrate smaller simulated download times than those seen from Hays & Simco alone, both with and without the use of choke, due to the pre-emptive pruning of poor-performing selected peers prior to initiating chunk download and the avoidance of non-inherent network activity for conveying performance information. The successful simulated demonstration of this research would justify the development, deployment, and real-world testing of a BitTorrent application that leverages this handshake-based peer switching.

Chapter 4

Results

Introduction

This research investigated the use of BitTorrent's handshake protocol as a conduit for peers to send estimated available service capacities to clients as a means of improving download times in Nicolas Hays & Gregory Simco's original research, both in regards to initial peer selection, as well as subsequent selection incurred during peer switching.

Prior works, like those mentioned in Chapter 2, have shown that lessening the time spent downloading from a poor-performing peer can reduce the overall peer-to-peer download time for a client (Chui & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). As such, one goal of this paper's research was to provide the client with on-the-fly intelligence regarding its selected peers' current estimated upload bandwidth availability, so that said client could completely avoid downloading from those peers whose estimates fell below its threshold by immediately pruning them.

One of Hays & Simco's design goals was to avoid introducing additional network overhead in gaining intelligence about peers. As such, their approach leveraged historical knowledge, gained ahead of time, so as to avoid additional on-the-fly overhead that could cut into the client's download performance (Hays & Simco, 2017). In keeping with this sentiment, another goal of this paper's research was to provide the aforementioned bandwidth availability estimates from selected peers to the client without incurring the additional on-the-fly network overhead typically associated with other intelligence

gathering approaches downloads (Li, 2012; Hsiao et al., 2011; Ying & Basu, 2006) by leveraging the network traffic already inherent in BitTorrent.

Chapter 3 described the new handshake-based enhancement to Hays & Simco's hybrid peer selection and peer switching strategy that was conceived during this paper's research in order to further improve upon the overall peer-to-peer download times achieved by Hays & Simco's original work. Chapter 3 also described the simulation environment that was created and used by this paper's research to evaluate the new handshake-enhanced Hays & Simco strategy and compare its overall download times with those of the original Hays & Simco work.

This chapter reviews the download times observed when recreating the strategies of prior works in order to demonstrate the validity of the simulation environment, as well as presents the results of the new strategy combining Hays & Simco's approach with handshake-based selection. A summary of the results concludes this chapter.

Simulation Validation

The console output for 10 simulation runs, measured in seconds, of a trial of random selection with time-based switching is shown in Figure 18.

```
Random Selection with Time-Based Switching
3978
3822
4701
3650
4080
3812
4264
4435
5143
4267
```

Figure 18: Console output for a random selection w/ time-based switching trial

From this console output and others like it, a large table was generated, which recorded the observed download times for each of the 10 simulated runs across the 100 trials using random selection with time-based switching.

Table 1, a subset of the aforementioned large table, shows the download times for the 10 simulation runs that were performed during the first 10 of the 100 trials conducted using random selection with time-based switching. For example, in the first trial column, the download times for the 10 simulation runs using random peer selection with time-based peer switching were 3978, 3822, 4701, 3650, 4080, 3812, 4264, 4435, 5143, and 4267 seconds, respectively.

	TRIAL	1	2	3	4	5	6	7	8	9	10
RUN											
1		3978	3956	4017	4076	5200	4079	5186	3956	5402	5460
2		3822	4705	4292	4035	4608	4372	4529	4656	4528	6414
3		4701	4319	6127	3937	4781	3164	4760	5552	5223	4483
4		3650	5156	3361	5203	4727	5437	4483	3727	3903	3690
5		4080	4872	3670	4517	3277	4238	4402	4515	5515	4607
6		3812	4965	4356	5991	5485	4287	4027	4216	4428	5165
7		4264	4027	4668	3803	5603	3947	3922	4997	4760	4360
8		4435	5139	4485	4401	4940	4570	4953	4049	5529	5162
9		5143	4165	4277	4623	4344	4032	5159	3753	3421	4869
10		4267	3938	5222	4810	5867	3711	6071	3807	3708	3896

Table 1: Random selection w/ time-based switching DL times for first 10 trials

Similar to Table 1, the corresponding download times results when the simulation environment was using random selection with time and choke-based switching, Hays & Simco's advanced knowledge-based selection with time-based switching, and Hays & Simco's advanced knowledge-based selection with time and choke-based switching are portrayed in Tables 2, 3, and 4, respectively.

	TRIAL	1	2	3	4	5	6	7	8	9	10
RUN											
1		3165	3066	2920	3037	2619	2672	2564	2604	3207	2685
2		2941	2454	2703	3124	2790	3288	2738	3054	2656	2601
3		3449	3504	2868	2864	2737	2635	3008	2872	2966	2686
4		3017	2534	2783	2490	3059	2708	3263	3032	3064	2161
5		2770	3048	2851	3062	3186	2809	2591	2802	3087	2972
6		3507	3089	2793	2524	3132	2684	2221	2594	2925	2854
7		2692	2619	2617	3680	3170	2241	2590	2962	3218	3090
8		2914	2856	3199	2913	2195	2828	3398	2882	2866	2786
9		2976	2735	2806	2637	2323	2444	2368	2895	2852	3012
10		2391	2549	2735	2386	2695	2550	2045	2606	2686	2349

Table 2: Random selection w/ choke-based switching DL times for first 10 trials

	TRIAL	1	2	3	4	5	6	7	8	9	10
RUN											
1		2108	1817	1762	1692	1644	1752	1325	2636	1873	2520
2		1762	1072	1531	1345	1636	2367	1253	1106	1224	1602
3		1529	1224	1279	1255	1349	2422	1826	1502	1026	2087
4		1019	1249	1141	1888	1213	1878	1708	1364	1221	1537
5		1032	1345	941	1234	1336	1288	2749	2283	1601	1635
6		1505	1376	1110	1182	964	1296	1288	1875	1929	1569
7		1569	1280	1519	1260	1246	1274	1561	1148	960	1095
8		1247	1204	1032	1053	1416	1125	1151	1279	960	1136
9		1648	961	1000	1032	1031	973	1009	1290	1024	1095
10		1569	966	989	1438	1239	1095	1045	1290	1018	1491

Table 3: Hays & Simco w/ time-based switching DL times for first 10 trials

	TRIAL	1	2	3	4	5	6	7	8	9	10
RUN											
1		1462	1052	1355	1238	1325	1183	1270	1204	1188	1226
2		986	984	960	1033	1211	998	1250	1051	1044	1035
3		1295	1108	975	960	1014	922	1032	1158	1032	1139
4		1130	1234	960	960	1059	1032	1004	1110	1116	1110
5		1326	1254	1110	1067	1032	1029	960	1052	1113	1010
6		1246	1110	1004	1188	1112	1032	1032	1095	1002	1031
7		1110	1122	960	1166	1032	1122	1054	1095	1025	1162
8		1125	1204	1180	1184	1032	1238	1032	1165	988	1000
9		1220	1155	1058	1095	974	1280	1037	1095	1015	986
10		1268	1176	960	1100	1033	1214	1143	1095	1074	1039

Table 4: Hays & Simco w/ choke-based switching DL times for first 10 trials

Table 5 shows what the average download time is for each of the 10 simulation runs across their respective experiment's 100 simulation trials, while Figure 19 presents these download time averages as a graph.

As was expected, the inclusion of choke-based peer switching, here set to 200 Kbps and at 1 minute intervals, substantially limited the impact of poor-performing peers, compared to time-based switching, set to 5 minute intervals. This was particularly evident in the case of random peer selection, where the use of choke-based peer switching reduced the download times achieved by time-based peer switching by 36.7%.

	STRATEGY	Random w/ Time-based Switching	Random w/ Choke-based Switching	Hays & Simco w/ Time-based Switching	Hays & Simco w/ Choke-based Switching
RUN					
1		4496.17	2801.53	1942.21	1241.75
2		4475.29	2847.43	1564.83	1085.16
3		4367.06	2813.79	1454.57	1062.12
4		4564.89	2814.07	1352.06	1065.9
5		4332.08	2843.01	1309.04	1070.79
6		4462.12	2817.88	1277.02	1072.32
7		4397.87	2841.81	1257.43	1072.66
8		4458.99	2806.39	1206.49	1086.41
9		4556.21	2807.48	1198.63	1077.76
10		4302.24	2742.1	1202.99	1082.1

Table 5: Random selection and Hays & Simco w/ time, choke-based switching average DL time

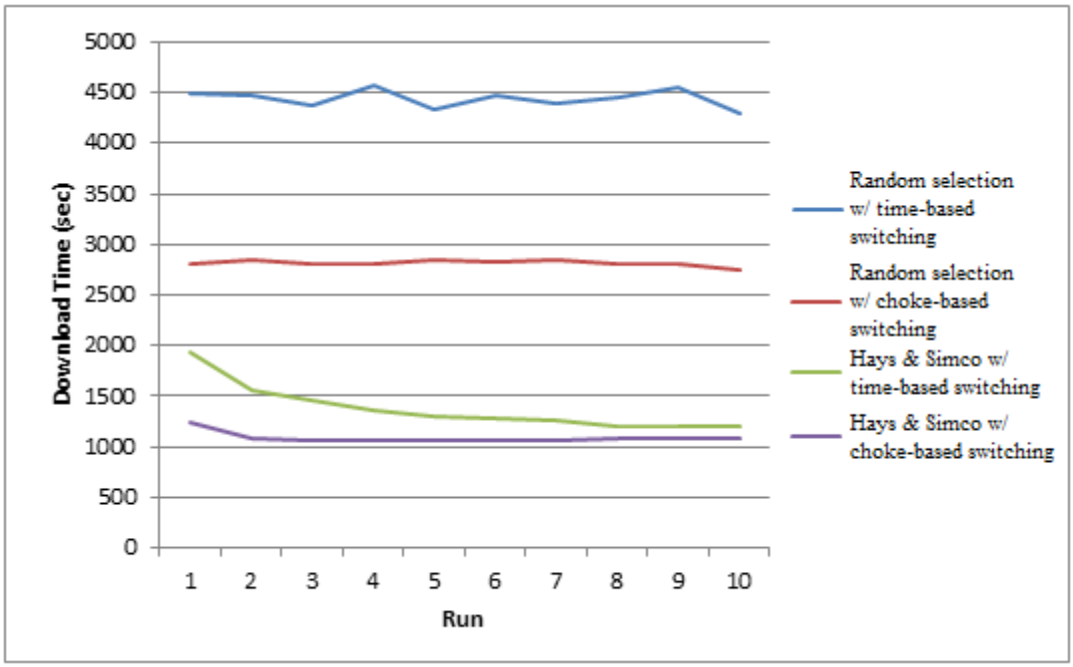

Figure 19: Random Selection and Hays & Simco w/ Time or Choke Switching DL Times

Hays & Simco's hybrid strategy also behaved as expected. Its use of advanced knowledge helped it avoid some poor performing peers that random selection could have otherwise hit, which accounted for a significant reduction in download times over random selection. Furthermore, Hays & Simco's download times generally improved over the course of a trial as the locally-stored prior knowledge regarding peers' service capacities got updated from one run to the next, which was also expected behavior.

As was the case in random selection, the addition of choke-based peer switching to Hays & Simco's strategy reduced download times over time-based switching alone. The average download time improved by 36.1% for the first run, 30.7% for the second, 27.0% for the third, and progressively tapered down to 10.0% by the tenth run, as the growing impact of prior knowledge updates reduced the effectiveness of the choke.

Based on these observed results adhering to expected behavior, this paper concluded that the simulation environment developed for this research was sufficiently validated and could be properly utilized for conducting experiments on the new handshake-enhanced Hays & Simco strategy.

Findings

Table 6 lists the observed download times for the first 10 trials' respective sequence of 10 simulation runs conducted using the new handshake-enhanced strategy. Table 7 presents the average download time across all 100 trials for the proposed methodology, while Figure 20 displays them as a line graph, as well as provides the corresponding download times for the two non-handshake-enhanced Hays & Simco approaches.

	TRIAL	1	2	3	4	5	6	7	8	9	10
RUN											
1		1070	994	1097	1049	938	960	885	981	1045	964
2		926	952	1062	981	860	909	1171	974	860	989
3		1006	938	960	1063	947	865	962	1026	1032	953
4		1032	922	960	1000	996	961	955	1010	1115	986
5		1027	922	1025	1045	921	1001	960	1032	1189	938
6		1146	935	1035	1134	860	1000	922	986	1093	1070
7		1078	1039	984	1144	1014	1000	922	1032	1077	998
8		1120	1056	1166	1032	960	998	946	1050	1038	1034
9		1178	966	1015	1201	960	980	1014	1110	1144	1033
10		1092	960	1050	1105	966	1120	986	1095	1130	1032

Table 6: Handshake-enhanced Hays & Simco DL times for first 10 trials

The download times observed for run #1 for the handshake-enhanced strategy were 15.8% smaller than that of Hays & Simco's strategy using choke-based peer switching. An 8.5% improvement in download times was observed in run #2, 5.7% in run

#3, and 4.7% in run #4. The difference in download times between the new handshake-enhanced strategy and Hays & Simco's original strategy with choke-based switching progressively decreased in successive simulated runs. While the handshake-enhanced strategy consistently outperformed its predecessor, its advantage diminished down to just 1.8% by the 10th and final run.

	STRATEGY	Hays & Simco w/ Time-based Switching	Hays & Simco w/ Choke-based Switching	Handshake-enhanced Hays & Simco
RUN				
1		1942.21	1241.75	1045.02
2		1564.83	1085.16	993.03
3		1454.57	1062.12	1001.76
4		1352.06	1065.9	1015.29
5		1309.04	1070.79	1025.36
6		1277.02	1072.32	1040.6
7		1257.43	1072.66	1043.81
8		1206.49	1086.41	1056.76
9		1198.63	1077.76	1061.86
10		1202.99	1082.1	1062.11

Table 7: Hays & Simco w/ time, choke, and handshake-based switching average DL times

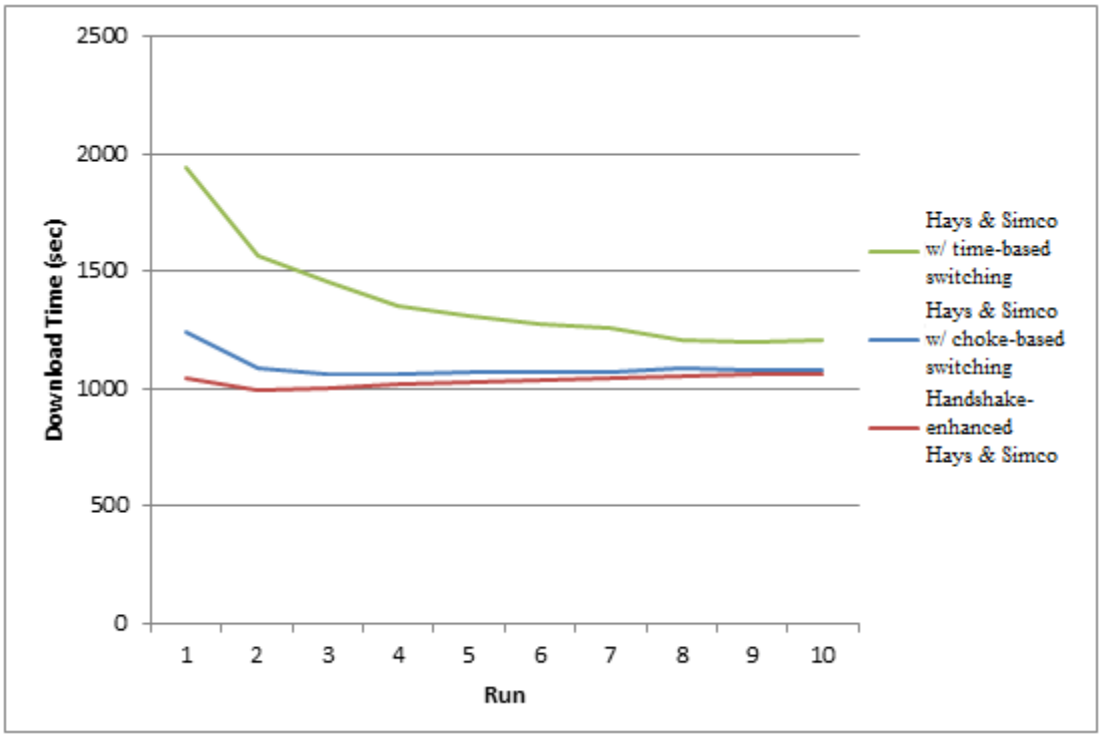

Figure 20: Results for Hays & Simco w/ Time, Choke, and Handshake-based Switching

Summary of Results

The results observed from the simulation environment experiments conducted showed that adding handshake-based peer selection to an approach that already leveraged Hays & Simco's advanced knowledge-based peer selection further decreased peer-to-peer file download times.

This performance improvement was particularly apparent when comparing the new strategy to Hays & Simco's approach without choke, as handshake-based selection effectively acted as an almost immediate and pre-emptive choke prior to fully establishing a download stream with a peer. On the other hand, Hays & Simco's approach using just time-based switching suffered from connections to peer-performers that could last as long as 5 minutes before switching peers.

Hays & Simco's approach using choke-based switching partially bridged the gap in performance. The use of choke-based switching reduced Hays & Simco's time with a selected poor-performing peer from 5 minutes down to potentially 1 minute. Furthermore, this reduced time with poor performing peers resulted in more frequent peer switches. Hence, the locally-stored prior service capacity knowledge used to select peers was updated more frequently, increasing the effectiveness of Hays & Simco's advanced knowledge strategy. As such, with each successive run within a trial, the combination of both choke-based switching and progressively-improving prior knowledge-based selection appeared to have worn away at the advantage gained from adding handshake-based selection, until after just 4 simulation runs the difference in download times between Hays & Simco's strategy with choke-based switching and its handshake-enhanced counterpart was less than 5 percent.

Chapter 5

Conclusions

Conclusions

The results obtained over the course of this paper's research supported the hypothesis that, without introducing any additional network overhead in the process, the overall download times achieved in the Hays & Simco (2017) advanced knowledge-based peer selection and choke-based peer-switching hybrid strategy could be further reduced by leveraging the reserved and unused bytes contained within the BitTorrent protocol's handshake message ("The BitTorrent Protocol", 2017) as a conduit for passing along a selected peer's estimated available service capacity to the client and using said information as a means of immediately replacing those peers whose estimates deemed them to be poor-performers.

As demonstrated by both time-based and choke-based peer switching strategies, reducing the amount of time the client spends downloading from a selected poor-performing peer, from the completion of a chunk down to a 5 minutes, and from 5 minutes down to as little as 1 minute, respectively, can reduce overall peer-to-peer download times (Chiu & Eun, 2008; Lehrfeld & Simco, 2010). The handshake-based peer switching strategy researched in this paper successfully reduced a BitTorrent client's overall download times even further by decreasing the time spent downloading from selected poor-performers to effectively zero, thus nearly mitigating their negative performance impact.

The new handshake-enhanced approach's download performance consistently exceeded those of its predecessor, Hays & Simco's hybrid historical knowledge-based peer selection and choke-based peer switching strategy, particularly in early file download runs when a client's locally-stored knowledge regarding the service capacities of peers was mostly derived from initial third-party data rather than more recently-acquired, first-hand experience (Hays & Simco, 2017).

However, as the client's first-hand experience updated the locally-stored data in subsequent runs, prior knowledge-based peer selection became more effective. As such, the client's selection of poor-performing peers became less frequent, thus diminishing the beneficial impact of faster peer switching.

Despite handshake-enhanced Hays & Simco's advantage over the original Hays & Simco strategy with choke-based switching becoming more and more negligible in latter experiment runs, the early run benefits leads this paper to conclude that the new approach serves as a suitable improvement to Hays & Simco's original research.

Implications

BitTorrent file transfers make up a substantial portion of the internet's overall usage (Bindal, Cao, Chan, Medved, Suwala, Bates & Zhang, 2006; Schulze et al., 2009). With its ability to make downloads more efficient, the research conducted in this paper has wide-ranging implications to the field of peer-to-peer networking.

Faster BitTorrent downloads not only can be more convenient for the client's user in regards to time spent waiting for a file transfer to complete, but can also save the client's user money, particularly for those clients hosted on metered networks. Since the

handshake message leveraged by this paper's new switching strategy for gathering intelligence about peers is network activity already inherent to BitTorrent, this research's enhancement does not introduce additional network overhead, thus avoiding increased delays and financial costs for the client.

This paper's research could also reduce the strain on peers. By having clients refrain from downloading from them, peers already contending with high demand and little available upload bandwidth remaining need not have to take on even more network load, particularly a load that would not be particularly satisfying for the client anyways.

. Given handshake-enhanced Hays & Simco's performance improvement was greatest during early simulation runs, this approach would be particularly effective for those who perform large BitTorrent downloads on occasion, rather than for prolific downloaders of smaller files.

Future Work

Though the addition of handshake-based peer switching was able to successfully demonstrate improved download performance over the original Hays & Simco strategy, this paper's new handshake enhancement may be better served as a complement to strategies that do not rely on locally-stored advanced knowledge, where successive downloads do not strengthen such strategies' peer selection performance that would otherwise cut into the benefits of service capacity-encoded handshaking. For example, handshake-based peer switching would likely work well paired with random peer selection. As such, a possible line of research would be to conduct similar simulation

experiments that couple handshake-based peer switching with existing peer-selection strategies.

A second avenue for future work could explore whether handshake-based peer switching could outright replace probe-based peer selection, given that both approaches help the client determine whether a peer has service capacity is sufficient to download from (Li, 2002; Hsiao et al., 2011). However, considering that handshake-based peer-switching does not need to perform an actual download to gauge a peer's performance, a download that could in and of itself be hampered by low bandwidth, it is possible that this paper's handshake-based enhancement could not only replace peer probing, but outperform it as well.

Another area of research could investigate using the BitTorrent protocol's handshake message not only as a conduit for embedding a peer's estimated available bandwidth, but also as means of gauging a peer's network latency. By having the client measure the length of time between when it sends its handshake message to a peer and when the peer's handshake response is received, the client could effectively perform an RTT query similar to that performed by AEPS (Hsaio et al., 2011) or traceroute-based peer selection (Yang & Basu, 2006), without introducing a separate query that would add network overhead. This could make the handshake-based enhancement even more useful to those strategies that lack some form of peer localization.

Summary

Since its introduction over twenty years ago, peer-to-peer networking has become a popular methodology for sharing files, streaming videos, delivering operating system

updates, and facilitating crypto-currencies. In particular, peer-to-peer file sharing constitutes a significant percentage of the internet's overall bandwidth usage. Given this prevalence of peer-to-peer file sharing and the negative impact that poor performing peers can have on such usage, finding ways of improving these peer-to-peer downloads has become an important area of research for both network providers and their consumers.

Work recently conducted by Nicolas Hays and Gregory Simco advanced this field of scientific research, having explored combining historical knowledge regarding peer performance and localization with choke-based peer switching as a means of both gaining intelligence for improved peer selection without incurring additional on-the-fly network overhead and further limiting the negative impact of those poor-performing peers that were selected.

This paper's research examined such prior works as time-based and choke-based peer switching, and extended their shared premise of reducing the amount of time spent downloading from poor-performing peers in order to create a new switching strategy. This new approach was used to complement Hays & Simco's hybrid strategy, imbuing it with the benefits of past probe-based peer selection strategies, while retaining the original work's goal of avoiding additional network overhead.

While this handshake-based peer switching enhancement was successful in reducing overall completion times for BitTorrent downloads using Hays & Simco's strategy, there is always room for improvement. The use of historical knowledge for peer selection, as demonstrated in Hays & Simco's approach, proves to be a powerful

performance enhancer that, once it is given the chance to learn and develop from recent first-hand experience, can quickly compete with this research's capabilities. As such, while handshake-based peer switching can improve Hays & Simco's download performance, it shows the most promise when combined with those strategies that do not benefit from prior knowledge.

References

Bindal, R., Cao, P., Chan, W., Medved, J., Suwala, G., Bates, T., & Zhang, A. (2006). Improving traffic locality in BitTorrent via biased neighbor selection. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on* (pp. 66-66). IEEE.

BitTorrent and µTorrent Software Surpass 150 Million User Milestone; Announce New Consumer Electronics Partnerships. [Web press release] (2012, January 9). Retrieved February 13, 2018 from https://web.archive.org/web/20140326102305/http://www.bittorrent.com/intl/es/company/about/ces_2012_150m_users

Bittorrent Protocol Specification v1.0. [Wiki] (2017, February 1). Retrieved February 24, 2018 from https://wiki.theory.org/index.php/BitTorrentSpecification

Bolliger, J., Gross, T., & Hengartner, U. (1999, March). Bandwidth modelling for network-aware applications. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)* (Vol. 3, pp. 1300-1309). IEEE.

Bott, E. (2017, May 10). Windows 10 installed base hits 500 million. *ZDNet*. Retrieved February 15, 2018, from http://www.zdnet.com/article/windows-10-installed-base-hits-500-million/

Chiu, Y. M. (2010). On the performance of content delivery under competition in a stochastic unstructured peer-to-peer network. *IEEE Transactions on Parallel and Distributed Systems*, *21*(10), 1487-1500.

Chiu, Y. M., & Eun, D. Y. (2008). Minimizing file download time in stochastic peer-to-peer networks. *IEEE/ACM Transactions on Networking (TON)*, *16*(2), 253-266.

DHT Protocol. [Web documentation] (2017, May 1). Retrieved February 23, 2018 from http://www.bittorrent.org/beps/bep_0005.html

Erman, D., Ilie, D., & Popescu, A. (2005). Bittorrent session characteristics and models. In *3rd International Conference HET-NETs' 05*.

Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B. B., & Dagon, D. (2007). Peer-to-Peer Botnets: Overview and Case Study. *HotBots*, *7*, 1-1.

Hays, N., & Simco, G. (2017). Reducing the Download Time in Stochastic P2P Content Delivery Networks by Improving Initial Peer Selection. *Proceedings of the ISCA 30th International Conference on Computer Applications in Industry and Engineering (CAINE-2017).*

He, Q., Dong, Q., Zhao, B., Wang, Y., & Qiang, B. (2016). P2P Traffic Optimization based on Congestion Distance and DHT. *J. Internet Serv. Inf. Secur.*, *6*(2), 53-69.

Hsiao, T. H., Hsu, M. H., & Miao, Y. B. (2011). Adaptive and Efficient Peer Selection in Peer-to-Peer Streaming Networks. *2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS)*, 753-758.

Lawey, A. Q., El-Gorashi, T., & Elmirghani, J. M. (2012, December). Energy-efficient peer selection mechanism for BitTorrent content distribution. In *Global Communications Conference (GLOBECOM), 2012 IEEE* (pp. 1562-1567). IEEE.

Lehrfeld, M. R., & Simco, G. (2010, March). Choke-based switching algorithm in stochastic P2P networks to reduce file download duration. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the* (pp. 127-130). IEEE.

Li, J. (2008). On peer-to-peer (P2P) content delivery. *Peer-to-Peer Networking and Applications*, *1*(1), 45-63.

Li, K. (2012). Probing high-capacity peers to reduce download times in P2P file sharing systems with stochastic service capacities. *International Journal of Foundations of Computer Science*, *23*(06), 1341-1369.

Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, *7*(2), 72-93.

Magharei, N., Rejaie, R., Rimac, I., Hilt, V., & Hofmann, M. (2014). ISP-friendly live P2P streaming. *IEEE/ACM Transactions on Networking*, *22*(1), 244-256.

Newman, J. (2015, March 16). Microsoft may speed up Windows updates with peer-to-peer distribution. *PC World*. Retrieved February 15, 2018, from https://www.pcworld.com/article/2897275/microsoft-may-speed-up-windows-updates-with-peer-to-peer-distribution.html

Pacifici, V., Lehrieder, F., & Dán, G. (2016). Cache bandwidth allocation for P2P file-sharing systems to minimize inter-ISP traffic. *IEEE/ACM Transactions on Networking (TON)*, *24*(1), 437-448.

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). ACM.

Ren, S., Liu, Y., Zhou, X., Tang, H., Ci, S., & Wang, M. (2013). A novel peer selection mechanism in heterogeneous wireless peer-to-peer networks. *2013 19th IEEE International Conference on Networks (ICON)*, 1-7.

Schulze, H., & Mochalski, K. (2009). Internet Study 2008/2009. *Ipoque Report*, *37*, 351-362.

Sherman, A., Nieh, J., & Sten, C. (2009). FairTorrent: bringing fairness to peer-to-peer systems. *Proceedings of the 5th international conference on Emerging networking experiments and technologies,* 133-144. doi: 10.1145/1658939.1658955

Speed Test FAQ. [Wiki] (2010, November 17). Retrieved August 22, 2019 from https://wiki.vuze.com/w/Speed_Test_FAQ

Speedtest-Intelligence. [Web documentation] (n.d.). Retrieved August 20, 2019 from https://www.ookla.com/speedtest-intelligence

Stainov, R., Goleva, R., Genova, V., & Lazarov, S. (2013). Peer port implementation for real-time and near real-time applications in distributed overlay networks. In *9th Annual International Conference on Computer Science and Education in Computer Science* (Vol. 29, pp. 87-92).

The BitTorrent Protocol Specification. [Web documentation] (2017, February 4). Retrieved February 24, 2018 from http://www.bittorrent.org/beps/bep_0003.html

Traverso, S., Abeni, L., Birke, R., Kiraly, C., Leonardi, E., Lo Cigno, R., & Mellia, M. (2015). Neighborhood filtering strategies for overlay construction in P2P-TV systems: design and experimental comparison. *IEEE/ACM Transactions on Networking (TON)*, *23*(3), 741-754.

Varvello, M., & Steiner, M. (2011, May). Traffic localization for DHT-based BitTorrent networks. In *International Conference on Research in Networking* (pp. 40-53). Springer, Berlin, Heidelberg.

Wang, L., & Kangasharju, J. (2013, September). Measuring large-scale distributed systems: case of bittorrent mainline dht. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (pp. 1-10). IEEE.

Xie, H., Yang, Y. R., Krishnamurthy, A., Liu, Y. G., & Silberschatz, A. (2008). P4P: Provider portal for applications. *ACM SIGCOMM Computer Communication Review*, *38*(4), 351-362.

Ying, L., & Basu, A. (2006, December). Traceroute-based fast peer selection without offline database. In *Eighth IEEE International Symposium on Multimedia (ISM'06)* (pp. 609-614). IEEE.

Zhang, Y., Zhou, X., Tang, H., & Bai, F. (2011, March). Peer selection in mobile P2P systems over 3G cellular networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on* (pp. 467-470). IEEE.