2017

Reducing the Download Time in Stochastic P2P Content Delivery Networks by Improving Peer Selection

Nicholas Hays
Nova Southeastern University, fordtruck19@yahoo.com

Reducing the Download Time in Stochastic P2P Content Delivery Networks by Improving Peer Selection

by

Nicholas J. Hays

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

College of Engineering and Computing
Nova Southeastern University

2017

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

_____ _____

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

_____ _____

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

_____ _____

Sumitra Mukherjee, Ph.D. Date
Dissertation Committee Member

Approved:

_____ _____

Yong X. Tao, Ph.D., P.E., FASME Date
Dean, College of Engineering and Computing

College of Engineering and Computing
Nova Southeastern University

2017

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

Reducing the Download Time in Stochastic P2P Content Delivery Networks by Improving Peer Selection

by
Nicholas J. Hays
May 2017

Peer-to-peer (P2P) applications have become a popular method for obtaining digital content. Recent research has shown that the amount of time spent downloading from a poor performing peer effects the total download duration. Current peer selection strategies attempt to limit the amount of time spent downloading from a poor performing peer, but they do not use both advanced knowledge and service capacity after the connection has been made to aid in peer selection. Advanced knowledge has traditionally been obtained from methods that add additional overhead to the P2P network, such as polling peers for service capacity information, using round trip time techniques to calculate the distance between peers, and by using tracker peers. This work investigated the creation of a new download strategy that replaced the random selection of peers with a method that selects server peers based on historic service capacity and ISP in order to further reduce the amount of time needed to complete a download session.

The strategy developed in this investigation extended prior works and used advanced knowledge that contained historic service capacity and ISP information. This information was used to make peer selection decisions. After the connection had been made the requested file was downloaded for a predetermined time period and the service capacity was logged. At the end of the predetermined time period the strategy replaced only the worst performing peers.

The results of this new historic based peer selection strategy have shown that there are benefits in using advanced knowledge to select peers and only replacing the worst performing peers. This new approach showed an average download duration improvement of 16.6% in the single client simulation and an average cross ISP traffic reduction of 55.17% when ISPs were participating in cross ISP throttling. In the multiple clients simulation the new approach showed an average download duration improvement of 53.31% and an average cross ISP traffic reduction of 88.83% when ISPs were participating in cross ISP throttling. This new approach also significantly improved the consistency of the download duration between download sessions allowing for the more accurate prediction of download times.

Acknowledgements

The completion of this dissertation would not have been possible without the support of many people.

First, I would like to thank my loving wife Beth. Without her love and support the completion of this journey would not have been possible. Her enthusiasm and support helped to keep me on track.

Next, I would like to thank Dr. Bixler. The lessons you taught me while we discussed my undergraduate research were the foundation on which my graduate research was built.

Dr. Simco, my dissertation advisor. Your drive for quality and the advancement of knowledge is inspiring. Thank you for all the time you spent reviewing my work and all of the comments you gave me. I have learned so much from you.

Dr. Mitropoulos and Dr. Mukherjee, thank you for serving on my dissertation committee and for the opportunity to expand my knowledge base in your classes.

Lastly, I would like to thank Northrop Grumman for the financial support I received for my doctoral studies.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Background

Peer-to-Peer (P2P) networking is widely used by applications for file sharing, instant messaging, video sharing, and streaming. Unlike the more traditional client server model, P2P networks consist of peers that communicate directly with one another, share resources, and can act as both a client and a server simultaneously (Ferragut, & Paganini, 2016, He, Dong, Zhao, Wang, & Qiang, 2016). Since P2P networks are inherently scalable they have the ability to overcome the bottleneck problem found in the centralized client server model, theoretically provide faster download times, and P2P networks provide a low cost alternative since they do not require the overhead of new servers and network equipment (Ying & Basu, 2006; Chiu & Eun, 2010; Ferragut, & Paganini, 2016).

File download time is one of the most important performance metrics in a P2P content delivery network since the downloading of the file generally consumes the majority of the time required to obtain the requested resource (Chiu & Eun, 2010; Li, 2012, 2014; Zuo & Iamnitchi, 2016). The actual file download time when compared to the theoretical download time of the P2P network allows the efficiency of the peer selection strategy used to be calculated (Chiu & Eun, 2008; Ferragut & Paganini, 2016). In addition the file download time can be used to compare and evaluate different peer selection strategies within the same P2P network (Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). While it is believed that the physical bandwidth of the downloading peers

is the only limiting factor with regards to download performance (Chiu & Eun, 2010), there are other factors that have been shown to have an effect on the download time. High latency, long distance, multiple hop connections with peers, and cross ISP traffic can all have significant negative effects, showing the importance of peer selection (Xie, Yang, Krishnamurthy, Liu, & Silberschatz, 2008; Pacifici, Lehrieder, & Dán, 2016). Since P2P applications typically use random peer selection (Yang, Zhou, Chen, Fu, & Chiu, 2015) from a list of candidates, there is the possibility of making an initial peer selection that is plagued with download performance damaging problems such as low upload bandwidth, high latency, unnecessarily long distance, unnecessarily high hops, and located within an ISP that is participating in P2P traffic throttling when a better performing peer could be available (Ying & Basu, 2006; Xie, et al., 2008; Pacifici, et al., 2016).

This research developed a new peer selection strategy for selecting server peers in a P2P content delivery network that further reduced the download time by making an informed decision on which peers to select. The research described in this paper demonstrates that using prior knowledge about the server peer's service capacity, in conjunction with its Internet service provider (ISP), for initial selection is feasible, and this advanced knowledge can be used to reduce the average download time for individual client peers within the network when compared to prior works. The remainder of this chapter is organized as follows. First the problem this research addressed is defined. Then the specific goal of reducing the average download time for the individual client peer is laid out. This chapter concludes with a discussion on the relevance and significance the

problem and goal have on the current state of P2P networks, and the barriers and issues that this research needed to overcome.

Problem Statement

P2P network traffic has overtaken all other forms of network traffic on the Internet as the dominant form of network traffic. P2P network applications such as BitTorrent, Kazza, Gnutella, and uTorrent provide users with the ability to share digital content over P2P networks, and these applications are consuming a large portion of the network's bandwidth (Gummadi, et al., 2003; Brienza, et al., 2016; He, et al., 2016; Zuo & Iamnitchi, 2016). By some estimates P2P traffic accounts for as much as 80% of the total Internet's bandwidth usage, and this traffic is a significant source of cross ISP traffic (Chiu & Eun, 2008; Chandran & Sajeev, 2015; Brienza, et al., 2016; He, et al., 2016; Pacifici, et al., 2016). P2P applications operate at the application layer and generally do not have the underlying network topology information available when conducting peer selection, which can lead to the network being less efficient (Magharei, Rejaie, Rimac, Hilt, & Hofmann, 2014).

When P2P network applications do not use network resources efficiently they can have a negative impact on the efficiency and performance of the Internet (Liu, Wang, Lin, & Cheng, 2008; Brienza, et al., 2016). This inefficient use of network resources can cause the creation of bottlenecks within the network by saturating paths in the network with P2P traffic, thus reducing the available bandwidth, which results in the reduced performance of other applications (Liu, et al., 2008; Ijaz, Saleem, & Welzl, 2013). The inefficient use of network resources also place additional strain on ISPs with the use of unnecessary cross ISP traffic (Liu, et al., 2008; Chandran & Sajeev, 2015; Pacifici, et al.,

2016), and extra strain is also placed on the network by introducing overhead with the use of additional network traffic caused by polling peers and the use of tracker peers (Wilkins, 2013; Brienza, et al., 2016). In addition the poor usage of available network resources, such as not downloading from the best performing peers available, can cause the client download times to vary significantly between download sessions (Chiu & Eun, 2008; Wilkins, 2013). This variation in the download times cause the performance of the P2P application to be inconsistent, which results in an inability to accurately predict the time necessary to download the requested file, and a decreased user's Quality of Experience (QoE) (Gummadi, et al., 2003; Chiu & Eun, 2008; Wilkins, 2013).

Since downloading peers in P2P networks also contribute their upload bandwidth (Ferragut & Paganini, 2016) the network should be able to completely saturate the client's available bandwidth; however, according to the work presented by Chiu & Eun (2008, 2010) the actual performance is significantly less than what is required to saturate the client's bandwidth. This is caused by the peer's service capacity fluctuating over time (Jain & Dovrolis, 2005). This fluctuation can cause the performance of peers to both degrade and improve (Chiu & Eun, 2008). Time spent retrieving a requested resource from a poor performing or degraded peer instead of an available higher performing peer adds to the overall retrieval time (Grummadi, et al., 2003; Chiu & Eun, 2008; Wilkins & Simco, 2013).

P2P applications that are operating on a heterogenic network, and selecting peers with the common random based download strategy have the increased possibility of becoming stuck downloading from a poor performing peer (Chiu & Eun, 2008). The random based strategy works by splitting the requested resource into predetermined byte

size pieces and randomly switches server peers at the end of each downloaded piece. There has been research into increasing the performance of the random based download strategy. Chiu & Eun (2008) reported that using a download switching strategy based on time connected rather than content size downloaded reduced the average download time by 40%. Research conducted by Lehrfeld & Simco (2010) improved the random based switching strategy by introducing a choke to the Chiu & Eun (2008) periodic switching strategy that further reduced the time spent with a poor performing server peer. Building on the works of Chiu & Eun (2008; 2010) and Lehrfeld & Simco (2010), Wilkins & Simco (2013) further reduced the average client peer's download time by using recent peer service capacity history to make an informed decision on which of the peers to continue downloading from and which to replace.

These prior works did not attempt to use advanced knowledge about the peers to aid in peer selection, and they did not attempt to predict possible bottlenecks such as ISP throttling. Instead these strategies continue to switch peers in an attempt to limit the negative effects of poor performing peers. Both Lehrfeld & Simco (2010) and Wilkins & Simco (2013) indicated that additional research in download time reduction is needed. Lehrfeld (2009) proposed that a hybrid algorithm approach could further reduce the file download time, and Wilkins (2013) noted that having advanced knowledge about the performance of the peers in the network and selecting top performing peers will provide greater performance than randomly selecting peers.

Biased based peer selection strategies use advanced knowledge of the peers in order to increase the possibility of selecting a good peer. Some of these methods consist of reducing file download time by minimizing cross ISP traffic (Bindal, et al., 2006; Steiner

& Varvello, 2011; Fernando & Keppetiyagama, 2013; He, et al., 2016; Pacifici, et al., 2016), by using economic based selection (Adler, et al., 2005), and by using network end-to-end measurement based techniques (Ying & Basu, 2006; Ijaz, et al., 2013; Traverso, et al., 2015). These and similar methods that use advanced knowledge of peers attempt to predict the networks performance; however, unlike the random switching strategies described they do not take the stochastic nature of the network into account, nor do they react to the changing of service capacities in real-time (Chiu & Eun, 2008; Wilkins, 2013). Since these methods do not monitor the current service capacities after the connection has been made these strategies do not have the ability to limit the amount of time spent with a poor performing peer (Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). These methods typically require additional overhead, when compared to the random based strategies, by needing large amounts of data stored on the client, and introducing additional network traffic by polling each of the peers for the information needed to make an informed peer selection (Chiu & Eun, 2008; Hsiao, Hsu, & Miao, 2011; Wilkins, 2013).

Neither the random based switching strategies nor the biased based strategies use both advanced knowledge for peer selection and the monitoring of the service capacities after the connection is made in order to identify and limit time spent connected to a poor performing peer (Chiu & Eun, 2008; Hsiao, et al., 2011; Wilkins, 2013; Pacifici, et al., 2016). Random based switching strategies focus on the switching of server peers after an initial selection is made, thus allowing for the possibility of time spent downloading from a poor performing peer before the client peer has the opportunity to switch to a potentially better performing server peer. Biased based selection strategies focus on

advanced knowledge to make the peer selection, but they do not monitor the current service capacities and are unable to switch to a new peer in the event that a poor performing peer was selected or the currently selected peer's performance degrades. Both the random based strategies and the biased based strategies attempt to minimize the effects of poor performing peers on the client's download time; however, neither of these strategy types attempt to minimize the negative effects of poor performing peers by using techniques both before a connection is made and during the file transfer.

Dissertation Goal

This work developed a new peer selection strategy that further reduced the average P2P download time for the individual client and the application load on the network. This was accomplished by creating a new hybrid download strategy that combined techniques to limit time spent with a poor performing peer both before a connection has been made and during the file transfer. These techniques were: the use of advanced knowledge for peer selection, monitoring the performance of the server peer after the connection has been made, and only replacing the worst performing peers. This newly developed strategy used locally stored data to initialize a list of potentially good performing peers in order to help the client peer make an informed decision on peer selection. This data can be obtained from sources such as the Federal Communications Commission (2017), OOKLA (2017), the National Broadband Map (2017), and the MaxMind (2017) GeoLite database. This data is available for download from their respected sources and contains information about average bandwidth, self tested bandwidth download and upload speeds, latency, location, and average bandwidth by geographic location. This information has been collected outside of the P2P network, and since it is stored locally

on the client there is no need to poll the other peers in order to gain the needed performance information.

Once the initial selection has been made, the hybrid strategy monitors the performance of the connected peers, updating the peer list with the current peer specific service capacities. At the end of each predetermined time interval the worst performing peers are replaced. As the list matures the strategy relies more on the recent history of the peers and less on the initial population of the list.

The prior works conducted on random switching techniques have shown that limiting the amount of time spent with a poor performing peer reduces the average download time for the individual client (Chiu & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). This research determined that using available advanced knowledge to improve peer selection and only replacing the worst performing peer further limited the amount of time spent downloading from a poor performing peer. The effects of this new peer selection strategy were measured in a simulated environment and compared to prior reported work. The goal of this research was met when this new strategy showed a decrease in the average download duration for the individual client when compared to the works conducted by Lehrfeld & Simco (2010) and Wilkins & Simco (2013).

Relevance and Significance

P2P file sharing traffic has become the dominate traffic type being transmitted on the Internet (Fernando & Keppetiyagama, 2013; Ferragut & Paganini, 2016; He, et al., 2016). It is estimated that 65 - 70% of the traffic traveling on the Internet backbone, traveling in the last mile 50 - 65% of download and 75 - 90% of upload traffic can be attributed to

P2P applications, and this traffic is continuing to grow (Li, 2008; Liem, et al., 2016). With such a majority of P2P file requests coming from individual users the performance optimization for the individual peer is an important issue (Li, 2014), and the main problem for these individual peers is peer selection (Bernstein, Feng, Levine, & Zilberstein, 2003; Li, 2014). While a peer's average service capacity is primarily governed by network topological parameters (Chiu & Eun, 2008; Chougule & Deshmukh, 2011), download times can range from a few minutes to several hours since service capacity is not constant throughout the entire download session (Chiu & Eun, 2008, 2010; Li, 2014). This fluctuation is in part due to the number of connected peers, the server peer's resource utilization, network congestion, and ISP throttling (Bindal, et al., 2006; Chougule & Deshmukh, 2011; Hsiao, et al., 2011; Li, 2012, 2014).

The new peer selection strategy developed in this research decreased the individual client's average download duration by improving peer selection within the simulated P2P network. Improved peer selection increases the efficient use of available network resources, decreases the negative effects a low performing peer has on the download duration, and reduces the energy consumed by the P2P network (Chiu & Eun, 2008; Liu, et al., 2008; Brienza, et al., 2016). Improved peer selection also increases the consistency between download sessions (Chiu & Eun, 2008; Wilkins & Simco, 2013). A more consistent download duration allows for the more accurate prediction of the amount of time and resource allocation needed to complete the requested file transfer, which in turn, increases the end user's QoE (Chiu & Eun, 2008; Fiorese, Simoes, & Boavida, 2013; Wilkins, 2013; Li, 2014). In addition these improvements also allow the P2P network to

better serve heavy requests for a scarce file by quickly replicating the file throughout the P2P network (Yang & De Veciana, 2004).

Barriers and Issues

Since the downloading of a file takes the most significant amount of time during a file transfer in a P2P contend delivery network numerous researchers have investigated identifying good performing partners (Li, 2012). Prior works in random based peer selection have successfully reduced the download duration in stochastic P2P networks when compared to the traditional byte based switching method, but these methods focused entirely on the amount of time spent with poor performing peers after the connection had been made (Chiu & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). They do not attempt to use advanced information that could aid in improved peer selection. The main barriers to using advanced knowledge, as reported in the literature, is the additional overhead that is required to gather and use this information. This overhead traditionally consists of needing large amounts of data stored on the client, introducing additional network traffic by polling each of the peers for the information needed to make an informed peer selection, and the use of additional peers as trackers to gain and store this information (Chiu & Eun, 2008; Hsiao, et al., 2011; Wilkins, 2013).

The ability to overcome these overhead barriers has only recently become available. Information about client and server performance is now being collected by multiple sources outside of the P2P network. The Federal Communications Commission (2017), OOKLA (2017), the National Broadband Map (2017), and the MaxMind (2017) GeoLite

database contain historic information about the peers that can be used to gain some insight on the potential performance level of a peer prior to making a connection.

Rather than storing the IP address and attributes for every address that has similar attributes, these addresses were listed in Classless inter-domain routing (CIDR) notation (Fuller & Li, 2006). The use of CIDR allowed for a single entry to represent a range of IP addresses. For example, the CIDR notation of 77.103.61.128/25 represents 128 addresses ranging from 77.103.61.128 to 77.103.61.255 (Fuller & Li, 2006). By storing each entry as a 32 bit IP address, an 8 bit CIDR Prefix, a 16 bit service capacity, and a 32 bit ISP identifier 1MB of storage can contain over 95,000 entries. This database was stored locally, eliminating the need for additional network traffic created by polling peers, and the use of tracker peers to store this information.

Real world testing is not feasible. The Internet is a dynamic network (Cheng, Hutchinson, Ito, 2008; Hsiao, et al., 2011; Scandizzo & Imperiali, 2014), which makes it very difficult to conduct scientific tests where the environment needs to be consistent so that the testing and evaluation of different algorithms can be performed in the same environment. Having a consistent, predictable, and scalable environment allows for the comparison of different P2P download strategies. A simulation environment was needed, and the simulation environment proposed by Chiu & Eun (2008) was the baseline used. This environment was modified to simulate a network that spans a geographic area and the throttling of service capacity when crossing an ISP boundary.

Summary

This research was based on the work presented by Wilkins & Simco (2013). Wilkins & Simco (2013) presented a smart peer replacement strategy that extended the random

periodic switching strategy presented by Chiu & Eun (2008). This research created a new peer replacement strategy that extended the smart peer replacement strategy by combining the use of advanced knowledge for initial peer selection, the monitoring of the current peer's service capacity, and only replacing the worst performing peers. The remainder of this report includes the following: Chapter 2 contains a comprehensive review of the literature focusing on prior works that contain techniques used to minimize file download time in a P2P network; Chapter 3 contains a detailed description of the methodology used in this research in order to create, verify, and evaluate the new peer selection strategy; Chapter 4 contains the results of the experiments performed in the simulated environments and a comparison between the work presented and prior works; finally Chapter 5 contains the conclusions of this research, the implications this work has on the field, and recommendations for future works.

Chapter 2

Review of the Literature

P2P applications have become a popular method to move digital files over the Internet (Ijaz, et al., 2013). It is estimated that the amount of P2P file sharing traffic accounts for between 40% and 80% of the total Internet's bandwidth usage (Chiu & Eun, 2008; Schulze, 2009; Lehrfeld, 2009; Pacifici, et al., 2016). Since the peers being used to obtain the requested resource have a significant impact on the efficiency, data rate, utilization, network performance, and user experience, peer selection has become an important issue in P2P networks (Ren, et al., 2013).

There have been some interesting investigations into different techniques that are used for peer selection with regards to P2P networks. This review of literature did not attempt to cover all of the work conducted, but instead this chapter attempts to cover some of the accomplishments made as they relate to the problem of reducing download time in stochastic P2P content delivery networks. The remainder of this chapter discusses the effects of heterogeneity and service capacity fluctuation on download times, and it gives an overview of some of the download strategies used in P2P networks for peer selections that were relevant to this research. The download strategies presented do not discuss the querying of the network to find peers that contain the requested resource, it is assumed that such querying has already been performed, and a list of potential server peers is available.

Heterogeneity and Service Capacity Fluctuation

P2P networks consist of multiple peers, and each peer has the possibility of having a different average service capacity. This heterogeneity is caused by differing physical connection speeds, geographic location, and number of hops between peers (Chiu & Eun, 2008, 2010; Kaune, et al., 2009; Chougule & Deshmukh, 2011). In addition to differing average service capacity, each peer also exhibits service capacity fluctuation due to peer resource allocation and temporary network congestion (Chiu & Eun, 2008). The level of heterogeneity and service capacity fluctuation has been shown to have impacts on P2P file download times (Chiu & Eun, 2008, 2010; Chougule & Deshmukh, 2011; Li, 2012, 2014, 2015).

Chiu & Eun (2008) gave the following examples that illustrate the possible impacts of heterogeneity and service capacity fluctuation. Consider a P2P network consisting of two source peers and one client peer. Both of the source peers have a service capacity of 150 Kbps. Given a 1 MB file size and no prior knowledge of the source peer's service capacity the expected service capacity from the network is (150 + 150)/2 = 150 Kbps. It would take 53.3 seconds to complete the file download. Now consider the same P2P network but with the source peers having service capacities of c_1 = 100Kbps and c_2 = 150 Kbps. The expected service capacity from this network is (100 + 150)/2 = 125 Kbps. It would take 64 seconds to complete the file download. However, the above calculation does not take into account the actual time spent receiving the portions of the file from each of the server peers, and since the client peer does not know what the service capacities are in advance it selects from the list of peers with equal probability in order to minimize the possibility of making a poor selection. This means that the actual download

time is ½(1 MB/100 Kbps) + ½(1 MB/150 Kbps) = 66.7 seconds. This shows that the level of heterogeneity in the P2P network causes the download time to increase (Chiu & Eun, 2008).

Using the same P2P network with heterogenic peers, assuming advanced peer service capacity knowledge, and now having the goal of reducing the file download time, the client peer would choose peer c_2 as the server peer since it has the higher service capacity of 150 Kbps. Now assuming that the service capacity of this peer is not constant, but instead fluctuating between 50 and 250 Kbps with equal probability and the process $C_2(t)$ is strongly correlated over time, it will take (1 MB/50 Kbps + 1 MB/250 Kbps)/2 = 96 seconds on average to complete the file transfer. The result is significantly longer than would be expected given that the average service capacity is 150 Kbps. This example shows that service capacity fluctuations can have a significant effect on the download performance, and both service capacity fluctuation and the level of heterogeneity need to be taken into account in P2P download strategies in order to reduce average file download times (Chiu & Eun, 2008).

Random Selection

The random peer selection strategy accounts for the most widely used P2P file download peer selection strategy (Sherman, Neih, & Stein, 2009). The random selection strategy works by allowing a client peer to randomly select a server peer from a list of peers that contain the desired resource. This type of selection strategy has the advantage of being robust, have low overhead, and this strategy is relatively easy to implement (Chiu & Eun, 2008; Traverso, et al., 2015). In addition this strategy can be effective when used with a network that consists of a high level of homogenous peers (Hsiao, et al.,

2011; Hirave, Surve, & Malgaonkar, 2013). Unlike the method described in this research, the random peer selection strategy does not contain any advanced knowledge of the selected peer's performance potential, which increases the likelihood that a poor performing peer may be selected, and even with the various switching techniques available unnecessary time will be spent connected to this peer.

Chiu & Eun (2008) looked at the random strategy from the perspective that the service capacity is not constant, but instead a stochastic process. In their research Chiu & Eun (2008) examined three different random download strategies: random chunk based switching, parallel downloading, and time based switching. The random chunk based switching strategy begins by dividing the file into equal sized chunks. The client peer then selects a server peer randomly from a list of peers and retrieves the requested file chunk. After the file chunk is retrieved the client peer selects a new server randomly and retrieves the next file chunk. This is repeated until all of the file chunks are received. By randomly selecting a new source peer for each file chunk request, the random chunk based switching strategy attempts to prevent a poorly performing peer from having to great of a negative impact on the total download time (Chiu & Eun, 2008).

In the parallel downloading strategy the client connects to n number of server peers simultaneously and retrieves the requested file F in evenly divided chunks based on the number of server peers connected, e.g. Number of chunks = F/n. This strategy requires that all of the file chunks complete before the download session is completed. In this strategy a poorly performing server peer will cause the total file download time to increase since the download time is the amount of time it takes to download from the

lowest performing peer, e.g. $\max(t_1, t_2, \ldots t_k)$ where t_k is the time it took to download chunk k (Chiu & Eun, 2008).

Time based switching focused on the amount of time spent downloading from a peer, rather than the more common completion of a transfer block, in order to reduce the average file transfer time. Like other random download strategies this strategy begins by randomly connecting to a server peer from a given list of peers. The strategy then downloads as much of the file as possible in the predetermined time period. At the end of each time block the strategy connects randomly to a new peer and continues the retrieval of the file. This continues until the entire file has been received. Chiu & Eun (2008) found that when the client peer switched to a randomly chosen peer at the end of each predetermined time slot the impacts of a poorly performing peer were minimized when compared to switching the server peer at the end of a byte block transfer. This is due to a poorly performing peer taking additional time to complete the transfer of a byte block, where as in time base switching the time spent with a poorly performing peer was fixed (Chiu & Eun, 2008).

In order to model the server peer's service capacity fluctuation Chiu & Eun (2008) used a stationary first-order autoregressive process (AR-1). Each time a new server peer was selected this process was used to generate a new service capacity. This service capacity was then used for that server peer until a new service capacity was requested. This model of service capacity fluctuation was re-created, and it was used to produce the needed service capacity fluctuation in the simulation environment for the re-creation of prior works and the evaluation of the work described in this research.

While the time based switching approach helped to alleviate the possible negative effects of staying with a poor performing peer for the entirety of transferring a block of data, it did not take into account the fluctuations of the peer's service capacity during the predetermined time period. Lehrfeld & Simco (2010) introduced a preemptive choke to address this problem. They placed the choke at the client level, which allowed the client peer to abandon the connection to the server peer in the event that the performance became poor. The choking algorithm works by calculating a new chokepoint value at the start of every connection. The client peer keeps a record of all peers that have been connected to during the current session, maintaining a running average of the network capacity combined with the average overlay capacity in order to deduce a realistic chokepoint threshold. The algorithm used to calculate the chokepoint is shown below where N is the number of contacted server peers and c_i is the service capacity for each server i = 1, 2, … N.

$$CP = \frac{1}{N}\sum_{i=0}^{N} c_i$$

The algorithm used this value to determine if the connected server peers were of an acceptable level and if they are not or their service capacity fell below the choke threshold during the transfer the client peer would terminate the connection before the end of the predetermined time period and connect to a new randomly selected peer. This work built upon the research conducted by Chiu & Eun (2008). Lehrfeld & Simco (2010) used the same AR-1 simulation environment to model the fluctuations in the server peer's service capacity as used by Chiu & Eun (2008). This choking method showed reduced download times in both single client and multiple clients scenarios over the chunk base and the time based switching algorithms by further shortening the time spent with poor

performing source peers (Lehrfeld & Simco, 2010). This work showed that the historic average service capacity can be used as an effective seeding mechanism, and these values can be used to help make a decision on whether the peer is currently performing at an acceptable level; however, unlike the method created in this research the Lehrfeld & Simco (2010) choking method did not use historic information to help make peer selections.

Chiu & Eun (2010) extended their previous work by further modifying the random peer selection strategy to be more optimized for a stochastic network with multiple peers downloading in parallel. In this type of environment they reaffirmed the need to be able to make a good peer selection so that time and bandwidth is not used connected to a poor performing peer. Chiu & Eun (2010) used a random walk to connect to peers since the current performance of each peer is not known until a connection is made. In this work the peer was allowed to connect to a random number of peers rather than just having one connection open for the entirety of the download. The algorithm connects to a randomly selected peer, polls this peer for the needed performance data, then remains connected, downloading for a short period of time (Chiu & Eun, 2010).

The data obtained from the random walk is then logged for future use, and once data exists for all of the peers, the algorithm is then able to make connection decisions based on the logged data. If the peer finishes downloading the file prior to the connection of all peers in the network the download session is ended. Chiu & Eun (2010) also showed that the common belief that always opening more connections will allow a download to complete in a shorter amount of time is not always true. In fact they showed that

increasing the amount of connections can saturate the network and cause the download time to increase (Chiu & Eun, 2010).

Chiu & Eun (2010) used the NS-2 simulations to compare the performance of the peer selection strategies tested. Since they made the assumption that the main bottleneck was the access link of the peers only those connections were limited, thus allowing the remainder of the network modeled to be left un-throttled. In order to model the stochastic fluctuation found in a network Chiu & Eun saturated the network at 90% capacity with non-related messages on an average of every 10 minutes (Chiu & Eun, 2010). The logging and use of historical service capacity in the Chiu & Eun (2010) work showed that current live data is not required to make an informed decision on which server peers could offer the best performance.

Wilkins & Simco (2013) further extended the works of Chiu & Eun (2008, 2010) and Lehrfeld & Simco (2010) by modifying the random peer selection strategy with the use of a smart peer replacement component and a smart peer replacement component with choke. The smart peer replacement strategy begins by opening between four and six connections to multiple server peers. These initial connections are made based on a locally stored performance list of peers that is initially seeded with randomly selected peers and random values that represented the peer's current service capacity. The performance list is then sorted and the top performing half of the list selected, while the remaining bottom half is replaced with new randomly selected peers. The algorithm then downloads a portion of the file from each of the server peers in the list for a predetermined time period. During each time period the server peer's service capacity is updated, in order to simulate service capacity fluctuation. This value is then stored in the

performance list as the server's new historic service capacity. At the end of a fixed number of predetermined time periods the performance list is sorted and the bottom performing half is then replaced with new randomly selected peers. The algorithm continues this behavior until the file transfer is completed (Wilkins & Simco, 2013).

The smart peer replacement with choke operates with the same behavior as without the choke, but it incorporated the choking method described by Lehrfeld & Simco (2010). In the event that the current server peer's performance drops below the set choke threshold this choking method allows the client peer to sever the connection before the predetermined time period expires and replace it with a new randomly selected peer (Wilkins & Simco, 2013).

Wilkins & Simco's (2013) implementations were tested and verified with the use of the same AR-1 simulation environment used in the works performed by Chiu & Eun (2008) and Lehrfeld & Simco (2010). Their work evaluated the smart peer replacement with and without choke, and the overall results showed an average reduction of 19.87% when compared to Chiu & Eun's (2008) work, and an average reduction of 8.69% when compared to the work performed by Lehrfeld & Simco (2010). The research conducted by Wilkins & Simco (2013) is directly relevant to the research conducted in this report. Like the work presented by Chiu & Eun (2010) the Wilkins & Simco (2013) download strategy compiled a performance list of peers based on historic service capacities, but the Wilkin & Simco (2013) work showed that a list of historic service capacities does not have to include data for every peer in the network before it is useful to make peer selection decisions. This affirmed the hypothesis that initializing the performance list

with historic data from a subset of the network instead of seeding the list with random values would result in reduced average download times.

All of the above described variations on the random download strategies rely on the purely random selection of initial server peers. None of these methods attempt to gather or use advanced knowledge of the potential server peer prior to the connection. The literature reported that some of the common reasons for not using advanced knowledge is the increased burden of additional polling messages on the network, the cost of storing this information locally, and that this data is historic and may not accurately represent the current service capacity. The literature has also reported that as improvements have been made on the random selection strategy the level of locally stored data and the use of historic data has increased (Chiu & Eun, 2010; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). In addition to the random selection strategy there are a number of additional strategies that attempt to use advanced knowledge about the peers to make an informed decision on peer selection. The remainder of this review of literature covers the advanced knowledge selection approaches that are relevant to the research described in this paper.

Advanced Knowledge Peer Selection

Biased based peer selection strategies attempt to use some form of advanced knowledge such as performance, capacity, cost, or other information in order to gain an insight on the other peers in the P2P network. This type of selection strategy then uses this advanced knowledge to make a determination on which subset of peers from a list of candidate peers are potentially good performing prior to initiating a file transfer

connection. In other words the biased based peer selection strategy attempts to build an overlay where the client peer's neighbors are good performing peers (Xie, et al., 2008).

The information needed to make a biased decision such as the network topology, the other peer's service capacity, network congestion, and other average download time limiting factors are not immediately known (Chiu & Eun, 2008; Wilkins, 2013). This type of strategy has traditionally had the problems of increased overhead and cost when compared to the random selection strategy since this advanced information must be gathered, stored, and evaluated. If each peer sends a probing message to every peer in the P2P network, stores the obtained data, and then does a lookup for peer selection the cost reaches $O(n^2)$ (Ijaz, et al., 2013). In addition this type of strategy does not react to the real-time service capacity fluctuations that are found in P2P networks (Chiu & Eun, 2008; Hsiao, et al., 2011; Wilkins, 2013). Similar to the selection strategies discussed below that use advanced knowledge of peers, the peer strategy developed in this research used advanced knowledge about the peer such as ISP and average service capacity in order to increase the likelihood that a good performing peer is initially selected; however, the peer selection strategy described in this research did not add additional network traffic to the P2P network by polling peers for information, nor did it use tracker peers.

ISP Based Selection

The use of P2P network overlays has introduced a significant challenge for ISPs and researchers. Since the P2P network architecture is built at the application layer there is often little to no consideration of the underlying physical network, resulting in an increase of expensive cross ISP traffic (Steiner, & Varvello, 2011; Fernando & Keppetiyagama, 2013). This cross ISP traffic is expensive for the ISP since they may

have to pay for traffic that crosses the ISP boundaries and these links between ISPs are assumed to be bandwidth bottlenecks (Akella, Seshan, & Shaikh, 2003; Bindal, et al., 2006; Varvello & Steiner, 2011; Pacifici, et al., 2016).

Since P2P network traffic constitutes a major portion of the available bandwidth, some ISPs have engaged in the practices of throttling and blocking P2P traffic, and as a result these practices have had a negative impact on P2P network average file download durations (Bindal, et al., 2006; Steiner, & Varvello, 2011; Fernando & Keppetiyagama, 2013; Pacifici, et al., 2016). Any proposed solution will need to reduce the cross ISP traffic without cooperation from the ISP since it is unlikely to happen due to security and privacy concerns (Varvello & Steiner, 2011; Ijaz, et al., 2013). Some researchers attribute the high cross ISP traffic to the random selection strategy since it does not take the peer's ISP into consideration when choosing server peers (Fernando & Keppetiyagama, 2013; Pacifici, et al., 2016).

In light of the ISP's throttling and blocking of cross ISP P2P traffic, Bindal et al (2006) conducted research to determine if BitTorrent (2016), a popular P2P protocol, would perform as well if the random strategy was modified to consider the ISP of the peers when making a peer selection. Bindal et al (2006) replaced the random strategy being used with a biased based strategy that selects the largest portion of the server peers from peers within the same ISP as the client peer. In addition the Bindal et al (2006) method also used a small set of peers from other ISPs. In order for this to have been feasible there had to be information that could identify which ISP each peer was a member of. Bindal et al (2006) offered two ways in which this could be achieved:

- Use tracker peers to return a list of peers that contain 35 – k peers that are members of the same ISP and k peers that are not. The tracker peers can use Internet topology maps or the ISP can publish their IP ranges to the tracker.
- Use P2P traffic shaping devices. These devices use deep packet inspection to identify and manipulate P2P traffic. When a new peer joins the network these devices intercept and manipulate the response from the tracker peer. This method can be implemented without making changes to the client peers or the tracker peers.

In the event there did not exist enough peers with the same ISP to make up a majority of the selected peers the downloading peer had to continue to contact the tracker peer to see if more peers with the same ISP had become available (Bindal, et al., 2006). In order to evaluate the biased based strategy Bindal et al (2006) used an event driven simulation of both a homogenous network and a heterogeneous network. Bindal et al (2006) concluded that replacing the random based strategy with a biased based strategy that strives to keep most of the P2P traffic within the ISP boundaries operates at near optimal levels and provided a method to bypass bottlenecks on the Internet.

The work performed by Bindal et al (2006) was not explicitly targeted at reducing the average download time of the individual client peer. The work instead aimed at reducing the amount of cross ISP traffic and improving the overall download time of the P2P network. Bindal et al (2006) showed that ISP throttling does effect peer download time, and this ISP throttling needed to be accounted for in the peer selection strategy. The authors' methods for identifying good peers consisted of using tracker peers and adding additional messages to the P2P network by continuing to poll the trackers. The download

strategy described in this research used a strategy similar to this work by initially choosing a predetermined number of server peers with the same ISP as the client, and the remainder of the server peers initially selected were peers with different ISPs; however, this strategy did not use tracker peers or traffic shaping devices to identify same ISP server peers nor did it require ISP cooperation or add overhead by introducing additional polling messages.

Another method for discovering the ISP of peers that has been used by researchers is by using the MaxMind (2017) GeoLite database to map the server peer's IP address to the server peer's ISP (Steiner & Varvello, 2011; Fiorese, et al., 2013). Similar to the P2P traffic shaping device method proposed by Bindal et al (2006) the method used by Steiner & Varvello (2011) to reduce the cross ISP traffic worked by intercepting all of the messages from peers during the resource availability querying and answering the request with peers that are located on the same ISP as the requesting client peer. In this work Steiner & Varvello (2011) describe a proof of concept prototype.

In order to determine what ISP a peer belonged to the authors queried the Maxmind (2017) GeoLite database with the requesting peer's IP address. Once the ISP is known the process then formed a peer list and returned it to the requesting peer. In the event that there is not enough same ISP peers to complete the list external peers were used (Steiner & Varvello, 2011). Like the work performed by Bindal et al (2006) the work performed by Steiner & Varvello (2011) did not have the objective to reduce the average download time of the individual peer, instead this work aimed to reduce cross ISP traffic. The Steiner & Varvello (2011) work also had the added overhead of needing tracker peers to

intercept the traffic. In addition this work also added to the overhead of the network traffic by polling and pinging all of the peers in the network (Steiner & Varvello, 2011).

The work performed by Steiner & Varvello (2011) demonstrated that a database of information that was not obtained from the polling of current peers in the network can be used to make peer selection decisions. The ability to store and access a database that can map an IP address to useful information locally gives the individual client the ability to query and use this information when deciding which of the available peers to connect to without the use of additional tracker peers and without adding additional network traffic to the P2P network.

Proximity Strategy

The location of peers has also been used to make an informed peer selection (Kaune, et al., 2009; Fiorese, et al., 2013). The proximity strategy focuses on selecting peers that are near the client peer from a list of available peers (Kaune, et al., 2009). In order to calculate distance and connection reliability between the client and potential server peers researchers have used end-to-end measurement methods such as round trip time (RTT), jitter, and number of hops (Ying & Basu, 2006; Kaune, et al., 2009; Fiorese, et al., 2013; Ijaz, et al., 2013).

Traceroute is an application that allows the user to detect routing problems, determine number of hops, calculate latency, characterize paths, and discover the underlying network topology (Mao, Rexford, Wang, & Katz, 2003). Ying & Basu (2006) used metrics obtained from traceroute in an attempt to make good peer selections. When a new peer entered the network a traceroute was performed by the tracker peer to obtain information on the new peer. This information was then stored by the tracker peer. In this

work the peers reported to the tracker in 1 minute intervals to inform the tracker that the peer is still connected (Ying & Basu, 2006).

With this information the tracker peer can make an informed decision on which subset of available peers to send in the query response message back to the requesting client. Unlike the method described in this paper, Ying & Basu's (2006) work required the use of a tracker peer in order to store the traceroute information and make it accessible to the client peer. This method also requires additional messaging overhead to obtain the needed information and to maintain a list of online peers (Ying & Basu, 2006). Since this method did not take the path of each server peer into consideration, it could create local bottlenecks that reduce performance of the peers in the event that multiple peers that use the same pathway are selected for connection (Ijaz, et al., 2013).

Ijaz et al (2013) developed a method to help with the local bottleneck problem associated with proximity based selection. In this method the selection was made based on the Fewest Common Hops (FCH). The client peer began by using traceroute to obtain information on all of the available candidate peers, stored the path topology, and then compared the results in order to make a decision on which of the peers to connect to. By using information obtained from traceroute this method enabled client peers to select peers that are nearest to the client, but also have maximum path disjointness in order to help minimize the possibility of data being sent through a common link simultaneously (Ijaz, et al., 2013).

While these methods are not directly applicable to the research conducted in this report, they do show additional methods of obtaining and using advanced knowledge in an attempt to make good peer selections. Similar to the ISP based selection strategies,

these methods also show that non service capacity information can be a useful metric when determining which peer to connect to. Unlike the new method described in this work, the proximity methods described relied on the additional overhead of using tracker peers, and polling peers. The new method created did not have this additional overhead; instead the new method stored the peer information locally on the individual clients and relied on this information rather than more current information that could be obtained by continuous polling. In addition these methods did not take into account the service capacity of the server peer, which this new method did.

Relevance of Advanced Knowledge Methods

The advanced knowledge peer selection strategies described above attempted to improve on the random strategy by using some form of advanced knowledge prior to making a peer selection. The literature revealed that non service capacity information such as a peer's ISP and location in the network can be used to help make good peer selection (Bindal, et al., 2006; Ying & Basu, 2006). Many of the strategies described required the overhead of using tracker peers, and the overhead of polling the peers within the P2P network to gain and use the required information for peer selection (Bindal, et al., 2006; Ying & Basu, 2006; Steiner & Varvello, 2011). Others have shown that information stored at the client level can be used to make peer selections (Ijaz, et al., 2013). These advanced knowledge peer selection methods did not take the server peer's service capacity or service capacity fluctuation into consideration. A nearby peer on the same ISP as the client does not guarantee that the service capacity and download performance will be better than a peer located further away using a different ISP.

The literature has indicated that the peer's service capacity, service capacity fluctuation, and ISP should be taken into account in order to select good performing peers (Bindal, et al., 2006; Chui & Eun, 2008). In addition the literature also indicated that the selection of peers that share a common link can create local bottlenecks (Ijaz, et al., 2013). The new method described in this research used advanced knowledge, but it did not poll peers. Instead it used pre-seeded information that was stored locally to make an informed decision when selecting peers. This data was updated with current service capacity information after the peer connection had been made, relying more heavily on the current service capacity information as the file transfer progressed.

Summary

The literature has shown several different techniques that are being used to select peers in a P2P content delivery network. These different strategies each have advantages and disadvantages. The random strategies discussed above focused on the current service capacity performance and attempted to limit the time spend with poor performing peers in order to decrease the average download time, but these strategies did not take any advanced information into consideration when making a peer selection (Chiu & Eun, 2008, 2010; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). Other strategies used advanced knowledge, but they required the use of increased network traffic and trackers to gain and use this information (Bindal, et al., 2006; Steiner & Varvello, 2011; Ijaz, et al., 2013). These strategies did not monitor the current service capacity performance, and they did not attempt to limit the time spent with a poor performing peer. The new strategy described in this research built upon the work conducted by Wilkins & Simco (2013) by combining the advantages of the random and biased strategies in order to create a peer

selection strategy that used advanced knowledge that was obtained without adding network traffic to the P2P network and without the use of trackers. This new method also monitored the current service capacity for poor performing peers, and finally it limited the time spent with the poor performing peers.

Chapter 3

Methodology

Introduction

The goal of this research was to further reduce the average download time for the individual client in a P2P content delivery network. This research accomplished this goal with the development of a new strategy that removed the random selection of peers in the peer selection algorithms reported in prior works and replaced it with a new biased strategy that used historic information about the peer's service capacity and ISP. This information was stored locally on the client peer in order to eliminate the need of tracker peers and the need of polling other peers for the information. If this information would have been obtained by polling every peer in the network it would have required n messages to be sent and received for every peer in the P2P network resulting in $O(n^2)$ messages being introduced into the P2P network as additional overhead where n is the number of peers in the network. By storing this information locally on the client this polling was not necessary (Ijaz, et al., 2013). This information was used to make an informed decision on which of the available peers could potentially be good performing peers. After the peers were selected and a connection was made the service capacity of the peers were monitored and the locally stored peer information was updated with the peer's current service capacity throughout the file transfer. Finally at the end of a predetermined time period the lowest performing peers were replaced with potentially better performing peers.

Once the described peer selection strategy was developed it was evaluated by comparing the new strategy's performance with the performance of the smart peer replacement strategy developed by Wilkins & Simco (2013) and the smart peer replacement strategy that incorporated the choking strategy developed by Lehrfeld & Simco (2010). As reported in the literature review these strategies used a variation of the random peer selection strategy, and these strategies built upon the successes of the work that came before them. Each of the research iterations has shown a further reduction on the average download time. The smart peer replacement and smart peer replacement with choke strategies' performance was verified in a modified simulation environment that was based on the work presented by Chiu & Eun (2008). The remainder of this chapter discusses the research methods that were used in this research to include the processing of the advanced information, the simulation environment, peer selection strategies from prior works that were used to validate the simulation, the new peer selection strategy developed, and the experiments that were performed with the new peer selection strategy.

Preprocessing Data

The data used in this research was obtained from OOKLA (2017) and the MaxMind (2017) GeoLite database. OOKLA (2017) collects data from users voluntarily testing their upload and download service capacities. According to OOKLA (2017) the Speedtest Intelligence data is the most comprehensive collection of global network performance results. At the time of writing OOKLA (2017) reports that over 9.3 billion user tests have been conducted. The data obtained from OOKLA (2017) contained information from users around the world and was accessible from comma separated value (CSV) files that

contain over 20 different metrics for each data sample. An example of a CSV file with one record is shown in Figure 1 (OOKLA, 2017).

Header
test_id,device_id,test_date,client_ip_address,download_kbps,upload_kbps,latency,server_name,
server_country,server_country_code,server_latitude,server_longitude,server_sponsor_name,client_country,
client_country_code,client_region_name,client_region_code,client_city,client_latitude,client_longitude,
miles_between,connection_type,isp_name,is_isp,carrier_name,manufacturer,device_name,
hardware_version,firmware_version,location_type
Data
52755322,4683086,6/1/2016 0:00,109.188.xxx.xxx,7498,7390,82,Vladimir,Russian Federation,RU,
56.1333,40.4167,Rostelecom,Russia,RU,Vladimir Oblast,VLA,Vladimir,56.121,40.3745,1.83595,11,PJSC
MegaFon,1,Yota,NOKIA,RM-974_1162,4.1.0.0,02040.00021.15053.36002,1

Figure 1 – Example OOKLA Raw Data

The metrics contained in this data that were of interest to this research are the upload service capacity, and the ISP. A subset of this data was created and consisted of the ISP and the ISP's upload service capacity. The algorithm to calculate the ISP's upload service capacity is shown below where N is the number of data samples for the ISP and x_i is the upload service capacity for each of the data samples i = 1, 2, … N.

$$SC = \frac{1}{N}\sum_{i=1}^{N} x_i$$

The MaxMind (2017) GeoLite database allows users to determine the ISP associated with an IP address. According to MaxMind (2017) the ISP to IP address mapping is 95% accurate within the United States and between 50% and 80% outside the United States depending on the country. They also note that a higher amount of Internet users within a country correspond to a higher level of accuracy (MaxMind, 2017). This data is updated twice a month and is accessible through an online application program interface (API), and this data is also downloadable in binary and CSV format. This research used the

downloadable CSV format. An example of the raw format of a MaxMind CSV file is shown in Figure 2 (MaxMind, 2017).

Header
IP Address/CIDR Prefix,ISP Name,Organization Name,Autonomous System Number,Autonomous System Organization
Data
192.168.0.0/24,"ISPName","OrgName",25,”AutoSystemOrg”

Figure 2 – Example MaxMind Raw Data

Of the attributes included in this data the IP address, CIDR prefix, and ISP were of interest to this research. This data was downloaded and all attributes other than the attributes of interest were removed. The ISP service capacity calculated from the OOKLA (2017) data was mapped to the corresponding ISP. The resulting data was examined for IP address overlap and any that was found was condensed using CIDR notation (Fuller & Li, 2006) by updating the CIDR prefix and the additional records removed. The resulting dataset contained the 32 bit IP address, 8 bit CIDR prefix, 32 bit ISP unique identifier, and the 16 bit average service capacity. Figure 3 shows the data structure that was used to organize each record in the dataset. This dataset was stored locally on the client and was used by the new biased peer selection strategy. This dataset was also used to populate the simulation environment.

Preprocessed Data Structure

```
1.  struct AdvancedInfoNode contains
2.      unsigned int ipAddress
3.      char prefix
4.      unsigned int isp
5.      unsigned short serviceCapacity
6.  end
```

Figure 3 – Preprocessed Data Structure

Simulation Environment

The prior works of Chiu & Eun (2008), Lehrfeld & Simco (2010), and Wilkins & Simco (2013) described a simulation environment that was used to test the various algorithms in their works. This P2P simulation environment used an AR-1 process to simulate the stochastic nature of the server peer's service capacity. This allowed for the simulation of service capacity fluctuation for each of the server peers in the simulated P2P network. Each time a connection was made to a server peer or an update to the service capacity was needed the AR-1 process was called and a new service capacity was returned.

$$C(t+1) = p * C(t) + \epsilon(t) + \alpha$$

The above formula describes the AR-1 process where p is the correlation coefficient, $C(t)$ is the last value returned, $\epsilon(t)$ is a sequence of independent and identically distributed (i.i.d.) random variables with a zero mean, and α is a constant that is varied such that $\mathbb{E}\{C(t)\} = \mu = \alpha/(1-p)$ (Chiu & Eun, 2008). An example of an AR-1 process sample plot where the average service capacity is set to 100 Kbps is shown in Figure 4.

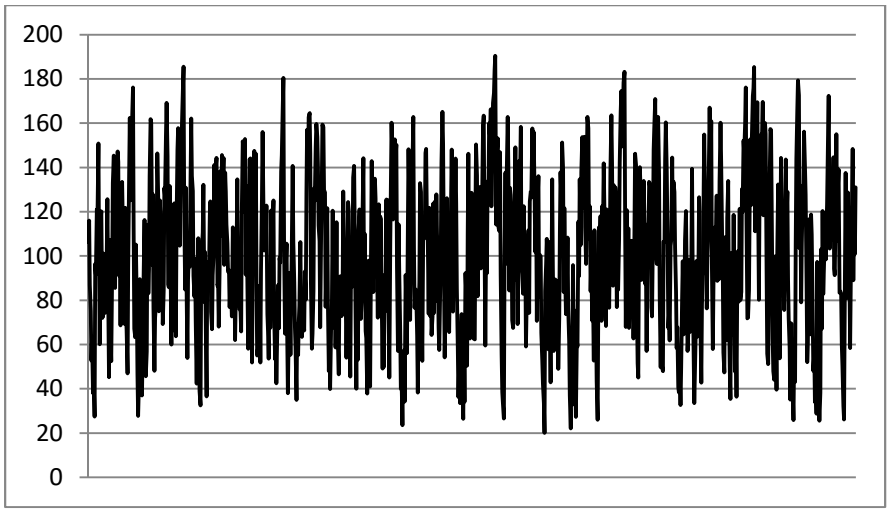

Figure 4 – AR-1 Stochastic Process Plot

Single Client Simulation

The single client simulation environment in the prior works used a file size of 150 MB with the average service capacity of the network set at 200 Kbps (Chiu & Eun, 2008; Lehrfeld, 2009; Wilkins, 2013). Four peers were used to simulate the server peers, all of which were assumed to have the file available for download. The peers' average capacities were altered for different simulation runs in order to modify the overall heterogeneity (δ) of the network. $\delta = \sqrt{\mathrm{Var}\{\vec{c}\}}/\mathrm{A}(\vec{c})$, where $\mathrm{A}(\vec{c}) = (c_1 + c_2 + \ldots + c_n)/n$. Table 1 shows the range of capacity heterogeneity of source peers that were chosen for the single client simulations in the works performed by Chiu & Eun (2008) and Lehrfeld (2009). The work conducted by Wilkins (2013) did not use the specific server peers shown, instead this distribution of server peer capacity was used as a template for creating the needed server peers while maintaining the same distribution of service capacity, and the same level of capacity heterogeneity (δ).

Table 1 – Average capacity of four source peers during the simulation runs

	Simulation Runs							
Source Peers	1	2	3	4	5	6	7	8
C_1	185	170	140	110	80	50	35	20
C_2	195	190	180	170	160	150	145	140
C_3	205	210	220	230	240	250	255	260
C_4	215	230	260	290	320	350	365	380
δ	0.05	0.11	0.22	0.33	0.45	0.56	0.61	0.67

Similar to the simulation environment used by Wilkins (2013), the research described in this paper also used the values shown in Table 1 for the purposes of validating the AR-1 simulation environment by the re-creation of the works performed by Chui & Eun (2008) and Wilkins & Simco (2013). When re-creating the work performed by Wilkins (2013) and conducting the single client experiments on the new biased based selection strategy Table 1 was used as a template for creating the needed server peers. The actual service capacity values used during the described experiments were derived from the preprocessed dataset. A similar distribution of service capacity and level of capacity heterogeneity (δ) shown in Table 1 was used.

Multiple Clients Simulation

The multiple clients simulation used in prior works added a source to client ratio to represent multiple client peers competing for the bandwidth of the server peers (Chiu & Eun, 2008; Lehrfeld, 2009; Wilkins, 2013). These prior works used the same AR-1 process as in the single client download simulation to simulate 100 peers with 4 different groups of service capacities. These groups consisted of 1 Mbps, 500 Kbps, 100 Kbps and 50 Kbps. The amount of peers in each group was distributed in a manner that would provide a heterogeneity value of 0.99. The distribution from 1 Mbps to 50 Kbps was 10,

5, 65, and 20 with an overall network service capacity of 200 Kbps (Chiu & Eun, 2008; Lehrfeld, 2009; Wilkins, 2013).

$$\mathbb{E}\{C\} = 1Mbps * .10 + 500Kbps * .05 + 100Kbps * .65 + 50Kbps * .20 = 200Kbps$$

The available capacity was divided evenly between the connected client peers, providing a method of indicating the amount of congestion or competition in the simulated P2P network. Since the server peer capacity was evenly divided the capacity of individual source peers was calculated by dividing the server peer's available service capacity by the source to client ratio (Chiu & Eun, 2008). For example if the server peer's available service capacity is 100 Kbps and the source to client ratio is 4, meaning there are 400 clients in the network, then the per client connection service capacity would be 25 Kbps. The amount of client peers were increased from 100 to 600 in 100 peer increments while the server peers remain fixed at 100 (Chiu & Eun, 2008; Lehrfeld, 2009; Wilkins, 2013).

Similar to the single client simulation used, the research described in this work used the values described in prior works (Chiu & Eun, 2008; Lehrfeld, 2009; Wilkins, 2013) for the purpose of validating the multiple clients simulation environment and re-creating the relevant prior works. This research used service capacity information that was obtained from the preprocessed data to populate the needed peers for the multiple client experiments conducted with the new biased based selection strategy while maintaining a heterogeneity value of 0.99.

ISP Throttling Simulation

As reported in the literature the practice of cross ISP throttling has the ability to make peers that would otherwise be good performing peers into poor performing peers (Pacifici, et al., 2016). The impact of ISP throttling on a given P2P connection is based on a number of factors to include: the client peer's ISP, server peer's ISP, server peer's upload capacity, client peer's download capacity, network path taken, downloading during peak demand hours, and protocol used (Bindal, et al., 2006; Dischinger, Mislove, Haeberlen, & Gummadi, 2008; Pacifici, et al., 2016). For example, consider a P2P network that consists of two server peers, c_1 and c_2, and one client peer. The server peers c_1 and c_2 have upload capacities of 1 Mbps and 500 Kbps, and these peers have ISPs of i_1 and i_2 respectively. The client has an ISP of i_2. Both ISPs are participating in cross ISP throttling with a bandwidth cap of 50 Kbps. If the client connects to the larger service capacity server c_1 the resulting service capacity will be the min(1 Mbps, 50 Kbps) = 50 Kbps, and if the client connects to c_2 the service capacity will be 500 Kbps. The simulation environment used in the prior works (Chiu & Eun, 2008; Lehrfeld, 2009; Wilkins, 2013) did not simulate cross ISP throttling. Under the prior works simulation the service capacity for c_1 would be 1Mbps, which does not accurately represent the example scenario above. The literature has indicated that this cross ISP throttling needs to be taken into consideration in order to select good performing peers (Bindal, et al., 2006; Steiner & Varvello, 2011; Fernando & Keppetiyagama, 2013; He, et al., 2016; Pacifici, et al., 2016).

When using either the preprocessed information or the statically assigned values found in prior works (Chiu & Eun, 2008; Lehrfeld, 2009; Wilkins, 2013) to populate the

simulation environment, the environment did not simulate the throttling of server peer's service capacity. In order to create a more accurate simulation environment the simulation environment used in this work was updated to simulate the throttling that takes place when P2P traffic cross the ISP boundaries (Bindal, et al., 2006; Magharei, et al., 2014).

If the ISP of the potential server peer was different than the requesting client peer and the server ISP was participating in throttling the service capacity was capped at the ISP's throttled service capacity. Since ISPs generally do not publicly disclose whether or not they are participating in ISP throttling, and the throttle service capacity being used is variable (Dischinger, et al., 2008) the ISP throttling simulation used random values for bandwidth caps. Whether or not the ISP was participating in cross ISP throttling was determined by the experiment being performed, experiments were conducted with both cross ISP throttling on and off. If the ISP was participating in throttling the throttled service capacity was randomly assigned to the different ISPs within the simulated P2P network and the throttled service capacity range used was 10 Kbps to 100 Kbps. For example, consider a P2P network that has a client operating from ISP i_1 and a server peer operating from ISP i_2. The server peer has a current service capacity of 200 Kbps, SC = 200 Kbps. ISP i_2 is practicing P2P traffic throttling and has a P2P bandwidth cap of 50 Kbps, BC = 50 Kbps. The returned result for the service capacity of the server peer would be 50 Kbps since the returned service capacity is the minimum value between SC and BC, e.g. min(SC,BC) = 50 Kbps. If the ISPs for the client and the server peers are operating within the same ISP or if the ISP is not practicing P2P traffic throttling the

returned service capacity is the non-capped service capacity of 200 Kbps. This algorithm is shown in Figure 5.

```
ISP Throttled Service Capacity
1.  Input: clientISP, serverPeer
2.  Output: serviceCapacity
3.  IF (clientISP == serverPeer.ISP or ISP not throttling)
4.      serviceCapacity = AR1(serverPeer)
5.  ELSE
6.      bandwidthCap = getBC(serverISP)
7.      serviceCapacity = min(bandwidthCap, AR1(serverPeer))
8.  END IF
9.  return serviceCapacity
```

Figure 5 – ISP Throttled Service Capacity

Confirmation of Prior Work

In order to validate the above simulation environment the algorithms described in the work performed by Chiu & Eun (2008), the smart peer replacement strategy described by Wilkins & Simco (2013), and the smart peer replacement strategy using the choking algorithm that was described by Lehrfeld & Simco (2010) were re-created and confirmed. The remainder of this section describes the single client without competition and the multiple clients with competition environment and the algorithms that were used to validate the simulation environment. Once the simulation environment was validated against the prior work the simulation environment was held static and the experiments with the new biased peer selection strategy described in this work were performed. All of the algorithms described in this section are variations of the random peer selection strategy. None of these algorithms attempted to use advanced knowledge about the potential server peer's service capacity, nor did they attempt to make selections based on possible ISP throttling.

Single Client without Competition

The work performed by Chiu & Eun (2008) implemented the permanent connection, chunk-based switching, periodic switching, and parallel downloading strategies. These four peer selection strategies were compared using the AR-1 single client simulation environment described above without the ISP throttling algorithm. The simulation environment used a file size of 150 MB with an average service capacity of the network set at 200 Kbps. The chunk-based switching strategy divided the file into 20 sections, each 7.5 MB in size. The periodic switching used a predetermined time slot of 5 minutes since that is the time it would take to download a 7.5 MB file using a service capacity of 200 Kbps (Chiu & Eun, 2008). The results of the Chiu & Eun (2008) single client experiments showed that a periodic switching algorithm based on the randomly switching to a new server peer at the end of a predetermined time period reduced the client download time when compared to chunk-based switching, permanent connection, and parallel chunk-based as the degree of heterogeneity approached 1.

The work performed by Wilkins (2013) implemented and compared the following four peer selection strategies: parallel periodic switching, choke based switching, smart peer replacement, and smart peer replacement with choke. The single client without competition experiments were conducted using a simulation environment similar to that of Chiu & Eun (2008) and Lehrfeld (2009). Instead of using the exact number of peers and service capacities as prior works, Wilkins (2013) used the information in Table 1 as a template to create as many peers as needed to conduct the single client without competition experiments. The levels of heterogeneity, file size, and average service capacity for the P2P network were left the same as in the prior works of Chiu & Eun

(2008) and Lehrfeld (2009). The results of the Wilkins (2013) single client experiments showed that keeping track of recent server peer history and only replacing the worst performing half of the connected servers at the end of each time interval further reduced the average client download time when compared to the parallel periodic switching and the choke based switching strategies.

The algorithms described in this section were modified to use the simulation environment described above by using the ISP throttling algorithm shown in Figure 5. In addition the Chiu & Eun (2008) periodic switching strategy and chunk based strategy were modified to use the 1 minute micro capacity adjustment described in the work performed by Lehrfeld (2009). This modification was done to accurately mimic the multiple client implementation of the periodic strategy and to allow for a more dynamic network capacity simulation (Lehrfeld, 2009).

Permanent Connection

The permanent connection download strategy consists of the client peer randomly selecting a server peer and maintaining that connection throughout the entire download session regardless of ISP or service capacity performance. The server peer's network capacity does not change during the download and the micro adjustment was not added to this strategy since this strategy was only used to validate the simulation environment. The permanent random connection algorithm is shown in Figure 6.

```
Permanent Random Connection
1.  Input: heterogeneity
2.  Output: downloadTime
3.  downloadTime = 0
4.  peer = randomly selected peer
5.  isp = getISP(selfIP)
6.  //Call the ISP Throttle Service Capacity
7.  bandwidth = ispTSC(isp, peer)
8.  downloadTime = fileSize / bandwidth
9.  return (downloadTime / 60)
```

Figure 6 – Permanent Random Connection

Chunk-Based Switching

The chunk-based switching download strategy consists of the client peer breaking the file to be downloaded into even fixed length chunks. Before each of the file chunks are downloaded the client peer randomly selects a new server peer to download from. This method attempts to avoid downloading from the same peer for the entire duration of the file download session in order to avoid downloading the entire file from a poorly performing peer (Chui & Eun, 2008). The chunk based switching algorithm is shown in Figure 7.

```
Chunk-based Switching
1.  Input: heterogeneity
2.  Output: downloadTime
3.  downloadTime = 0
4.  isp = getISP(selfIP)
5.  FOR i = 0 to 19 DO // File is divided into 20 chunks
6.          peer = randomly selected peer
7.      FOR j = 0 to 4 DO //loop for micro adjustments
8.          //Call the ISP Throttle Service Capacity
9.          bandwidth = ispTSC(isp, peer)
10.         downloadTime += (fileSize / (20 * 5)) / bandwidth
11.     END LOOP
12. END LOOP
13. return (downloadTime / 60)
```

Figure 7 – Chunk-Based Switching

Periodic Switching

The periodic switching download strategy consists of switching source peers periodically based on a predetermined time interval rather than the completion of some portion of the download (Chiu & Eun, 2008). Before a time interval begins the client peer randomly selects a new server peer. The client then downloads as much of the file as possible from this selected server peer for the duration of the predetermined time interval. Before a new time interval begins the client peer randomly selects a new server peer. This method helps to minimize the amount of time spent downloading from a poor performing server peer since the time interval is fixed rather than taking longer to download a fixed sized (Chiu & Eun, 2008). The periodic switching algorithm is shown in Figure 8.

```
    Periodic Switching
1.  Input: heterogeneity
2.  Output: downloadTime
3.  downloadTime = 0
4.  isp = getISP(selfIP)
5.  totalDownloaded = 0
6.  downloaded = 0
7.  WHILE fileSize > totalDownloaded DO
8.      peer = randomly selected peer
9.      //Call the ISP Throttle Service Capacity
10.     bandwidth = ispTSC(isp, peer)
11.     FOR 0 to 4 DO //loop for micro adjustments
12.         bandwidth = ispTSC(isp, peer)
13.         downloaded = bandwidth * 60
14.         IF ((fileSize - totalDownloaded) < downloaded)
15.             downloadTime += (fileSize - totalDownloaded) / bandwidth
16.             totalDownloaded = fileSize
17.            Break Download Complete
18.         ELSE
19.             downloadTime += 60
20.             totalDownloaded += downloaded
21.         END IF
22.     END LOOP
23. END LOOP
24. return (downloadTime / 60)
```

Figure 8 – Periodic Switching

Parallel Downloading

The parallel downloading strategy consists of dividing the file across all available server peers. Each server peer is connected to and an even portion of the file is downloaded. The total time to download the file depends on the worst performing peer since that peer will take the longest to finish; however, since the portion of the file is smaller this method can offer lower download times than a single chunk-based switching when the degree of heterogeneity is low (Chiu & Eun, 2008). The micro service capacity

adjustment was not added to this strategy since it was only used for simulation validation. The parallel chunk based switching algorithm is shown in Figure 9.

```
    Parallel Chunk-based Switching
1.  Input: heterogeneity
2.  Output: downloadTime
3.  downloadTime = 0
4.  isp = getISP(selfIP)
5.  //Call the ISP Throttle Service Capacity
6.  tempBandwidth = ispTSC(isp, peer[0])
7.  FOR i = 0 to 3 DO
8.      bandwidth = ispTSC(isp, peer[i + 1])
9.      IF (tempBandwidth < bandwidth)
10.         bandwidth = tempBandwidth
11.     END IF
12. downloadTime = (fileSize / 4) / bandwidth
13. return downloadTime / 60
```

Figure 9 – Parallel Chunk-Based Switching

Smart Peer Replacement

The smart peer replacement strategy built upon the periodic switching strategy described by Chiu & Eun (2008). This strategy uses recent server peer service capacity history to further reduce the individual client peer's average download time. This strategy randomly populates a list of server peers. Each server peer in this list is assigned a random value to represent its service capacity. This strategy then begins downloading the requested file by sorting the server list by the service capacity and replacing the worst performing half with new randomly selected peers. The strategy then downloads from each of the servers for 5 minutes, updating the service capacity for the server peer in the locally maintained list at each 1 minute intervals. This behavior is continued until the entire file has been received (Wilkins & Simco, 2013). The smart peer replacement algorithm is shown in Figure 10.

```
    Smart Peer Replacement
 1. Input: heterogeneity
 2. Output: downloadTime
 3. downloadTime = 0
 4. isp = getISP(selfIP)
 5. totalDownloaded = 0
 6. downloaded = 0
 7. serverList[numOfConnections] = random servers and capacities
 8. WHILE fileSize > totalDownloaded DO
 9.     sort(serverList) //Sort server list by service capacity
11.     //Replace worst performing half of serverList
10.     FOR 1 to serverList.size / 2 DO
11.         serverList[size - i] = randomly selected peer
12.         //Call the ISP Throttle Service Capacity
13.         serverList[size - i].serviceCapacity = ispTSC(isp, peer)
14.     END LOOP
15.     FOR EACH peer in serverList DO
16.         FOR 0 to 4 DO  //loop for micro adjustments
17.             bandwidth = ispTSC(isp, serverList[i])
18.             serverList[i].serviceCapacity = bandwidth //Update serverList
19.             downloaded = bandwidth * 60
20.             IF ((fileSize - totalDownloaded) < downloaded)
21.                 downloadTime += (fileSize - totalDownloaded) / bandwidth
22.                 totalDownloaded = fileSize
23.                 Break Download Complete
24.             ELSE
25.                 downloadTime += 60
26.                 totalDownloaded += downloaded
27.             END IF
28.         END LOOP
29.     END LOOP
30. END LOOP
31. return (downloadTime / 60)
```

Figure 10 – Smart Peer Replacement

Smart Peer Replacement with Choke

The smart peer replacement with choke was built upon the smart peer replacement algorithm described in Figure 10 with the addition of the choking method described in the

work performed by Lehrfeld & Simco (2010). This strategy behaves in a similar manner as the non choke strategy with the following modification. At the end of each 1 minute micro adjustment the strategy checks the service capacity of the server peer to ensure that it is above the set threshold. If it is the choke method is updated with the server peer's current service capacity. If the strategy is not above the threshold the strategy severs the connection with the server peer and replaces it with a new randomly selected peer (Wilkins & Simco, 2013). The smart peer replacement algorithm with choke is shown in Figure 11.

```
    Smart Peer Replacement with Choke
 1. Input: heterogeneity
 2. Output: downloadTime
 3. downloadTime = 0
 4. isp = getISP(selfIP)
 5. totalDownloaded = 0
 6. downloaded = 0
 7. serverList[numOfConnections] = random servers and capacities
 8. WHILE fileSize > totalDownloaded DO
 9.     sort(serverList) //Sort server list by service capacity
10.     //Replace worst performing half of serverList
11.     FOR 1 to serverList.size / 2 DO
12.         serverList[size - i] = randomly selected peer
13.         //Call the ISP Throttle Service Capacity
14.         serverList[size - i].serviceCapacity = ispTSC(isp, peer)
15.     END LOOP
16.     FOR EACH peer in serverList DO
17.         FOR 0 to 4 DO //loop for micro adjustments
18.             bandwidth = ispTSC(isp, serverList[i])
19.             serverList[i].serviceCapacity = bandwidth //Update serverList
20.             downloaded = bandwidth * 60
21.             IF ((fileSize - totalDownloaded) < downloaded)
22.                 downloadTime += (fileSize - totalDownloaded) / bandwidth
23.                 totalDownloaded = fileSize
24.                 Break Download Complete
25.             ELSE
26.                 downloadTime += 60
27.                 totalDownloaded += downloaded
28.                 IF (Peer is below capacity threshold)
29.                     Randomly choose new peer and update serverList
30.                 ELSE
31.                     Update the capacity threshold with additional data
32.                 END IF
33.             END IF
34.         END LOOP
35.     END LOOP
36. END LOOP
37. return (downloadTime / 60)
```

Figure 11 – Smart Peer Replacement with Choke

Multiple Clients with Competition

Chiu & Eun (2008) used the multi-client with competition environment described above to compare the performance of the chunk based and periodic switching strategies. Three strategies were simulated: a parallel version of the chunk based strategy with varying chunk sizes, a chunk based strategy with a single connection and a chunk size of 7.5MB, and a periodic switching strategy with a single connection. The file chunk size of 7.5MB was used since it is the same size that was used in the single client experiments (Chiu & Eun, 2008). This was done for a fair comparison against the periodic switching algorithm. The chunk based switching strategy described in Figure 7 and the periodic switching strategy described in Figure 8 were adapted to re-create the experiments described by Chiu & Eun (2008). The results of the Chiu & Eun (2008) multiple client experiment showed that a periodic switching algorithm produced lower download times than the chunk-based algorithm when competition was introduced into the network.

Wilkins (2013) conducted multiple client experiments on four strategies: parallel time, parallel time with choke, smart peer replacement, and smart peer replacement with choke. These strategies were compared using the same simulation environment described in this paper. In addition to the 150 MB file size that was used in the work conducted by Chiu & Eun (2008), Wilkins (2013) also used file sizes of 300 MB and 3 GB. The results of the Wilkins (2013) multiple client experiments showed that replacing the worst performing half of the server peers, rather than replacing the entire server list, at the end of the predetermined time slot reduced the download time for the individual client over the other strategies tested. The research discussed in this paper re-created the Chiu & Eun (2008) experiment using the chunk based strategies and the periodic switching strategy,

and it re-created the work performed by Wilkins (2013) in order to validate the multiple client simulation environment.

Experiments Completed

Once the simulation environment was created and the prior works re-created, the modified simulation described in this work was verified by re-implementing the single client without competition experiment and the multiple clients with competition experiments from the prior works performed by Chui & Eun (2008) and Wilkins & Simco (2013) and executed in the modified simulation environment. The results from the prior work re-implementation were written out to a CSV file, imported into Microsoft Excel, plotted into chart form, and compared with the reported results and confirmed the simulation environment produced similar results.

The work performed in this report evaluated the use of historic data that included average service capacity and ISP for the individual peers in the P2P network to make peer selection decisions. This research improved upon the smart peer replacement strategy by using historic data to initialize the server list rather than populating it with random values. This method also allowed the client to anticipate ISP P2P throttling and avoid bandwidth bottlenecks by keeping a running average of the ISP's service capacity. If the ISP is participating in P2P throttling the ISP level service capacity will be low, thus moving all of the peers that have the same ISP further down the performance list. The goal of this new strategy was to further reduce the individual client peer's average file download time in a P2P network when compared to the peer selection strategies presented by Lehrfeld & Simco (2010) and Wilkins & Simco (2013). The re-creation of the work performed by Wilkins & Simco (2013) was compared to the results of the newly

created strategy and this strategy showed significant improvements over the prior works. The experiments described in this section were performed with no ISPs participating in P2P throttling and with ISPs that were participating.

Historic Based Selection

The new biased peer selection strategy created was based on the smart peer selection strategy, shown in Figure 10, presented by Wilkins & Simco (2013) and further reduced the average download time of the individual client in a P2P network. Similar to the smart peer selection strategy the new historic based strategy monitors the performance of parallel downloads from several peers, logs the service capacity data, and uses it to make an informed decision on which of the currently connected peers are poor performers and need to be replaced (Wilkins & Simco, 2013). The smart peer selection strategy does not have any advanced information about the peers, it randomly initializes the performance list, and it selects new peers randomly. This selection strategy must make a connection and spend time downloading the resource in order to gain the service capacity information needed to determine if the server peer is a poor performer. The smart peer replacement strategy attempts to minimize the effects of being connected to a poor performing peer by limiting the amount of time spent with the server peer (Wilkins & Simco, 2013).

The literature has shown that both service capacity (Chiu & Eun, 2008, 2010; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013) and the peer's ISP (Bindal, et al., 2006; Steiner & Varvello, 2011; Fernando & Keppetiyagama, 2013; He, et al., 2016; Pacifici, et al., 2016) can be used to make a determination on which peers could be good performers. Unlike the prior methods described in the literature review (Chapter 2), the

historic based strategy used both service capacity and ISP information to initially determine which peers to connect to. This strategy used advanced knowledge from the preprocessed data described above. Each element in the preprocessed data contained an IP address, a CIDR prefix, the corresponding ISP, and the historic average service capacity for the ISP (Figure 3).

Each peer in the simulation environment was assigned an IP address. This IP address was used by the simulation environment to query data obtained from the preprocessing of the OOKLA (2017) files and the MaxMind (2017) GeoLite database described above. The preprocessed data used in this research covered all IPv4 addresses (2^{32}). The number of records in the data was dependent on how the addresses were divided among the ISPs in the MaxMind (2017) GeoLite data. The preprocessed data used in this research contained over 1.7 million records and required 18.6MB of disk space. The preprocessed data was loaded into a binary search tree (BST) resulting in a search time complexity of O(log n) for each potential server peer.

This data allows the historic based strategy to have access to peer's historic service capacity and ISP by querying the information with the potential server peer's IP address. The result of this query is a populated Peer Node structure. The Peer Node structure is shown in Figure 12.

	Peer Node
1.	**struct** PeerNode **contains**
2.	unsigned int ipAddress
3.	unsigned int isp
4.	unsigned short serviceCapacity
5.	**end**

Figure 12 – Peer Node Structure

This information is then used to initially rank the potential servers by their historic service capacity. The potential server list is separated into two different lists: peers with the same ISP as the client, and peers that have different ISPs than the client. If the ISP does not already have a service capacity from the preprocessed data then a random value with the range of 1 to the maximum known service capacity is assigned. The potential of these peers are unknown prior to connection. They could be good performing peers or they could be poor performing peers, and without a value for the service capacity they will be moved to the bottom of the list when it is sorted, thus decreasing the chance of being used as a server peer. Once the server has been connected the current service capacity will be captured and the peer's actual service capacity will be known. The two lists are then sorted by service capacity. The predetermined numbers of top performing peers from each list are then combined into a list of servers to initially connect to. The initialize peer list algorithm is shown in Figure 13.

```
    Initialize Peer List
 1. Input: clientISP, numOfConnections, numOfSameISP
 2. Output: List of server peers
 3. serverList[numOfConnections] = {0}
 4. FOR EACH peer in potential server list DO
 5.     //query historic data for peer
 6.     PeerNode tmpPeer = getPeerData(peer.IP)
 7.     //assign a random service capacity if one does not exist
 8.     IF (tmpPeer.serviceCapacity == 0)
 9.         tmpPeer.serviceCapacity = randomINT{1, maxSC}
10.     END IF
11.     IF (clientISP != tmpPeer.isp)
12.         nonSameISPList.add(tmpPeer)
13.     ELSE
14.         sameISPList.add(tmpPeer)
15.     END IF
16. END LOOP
17. sort(nonSameISPList) //sort lists by service capacity
18. sort(sameISPList)
19. FOR 0 to numOfSameISP - 1 DO
20.     serverList.add(sameISPList[i])
21. END LOOP
22. FOR 0 to (numOfConnections - numOfSameISP)  DO
23.     serverList.add(nonSameISPList[i])
24. END LOOP
25. return serverList
```

Figure 13 – Initialize Peer List

Once the server list has been initialized the strategy connects to each of the servers in the list. Once connected the strategy begins downloading the resource and monitoring the service capacity of each server. As the service capacities of the servers are updated with the AR-1 process, to simulate fluctuation, the strategy updates the potential server list with this new information. The specific peer's service capacity is then updated with the current service capacity and all other peers with the same ISP as the current server peer is updated with a running average for that ISP. In addition the preprocessed data stored

locally on the client is updated in non-volatile memory with the running average for the ISP so that the information is available during future download sessions. The update process is shown in Figure 14.

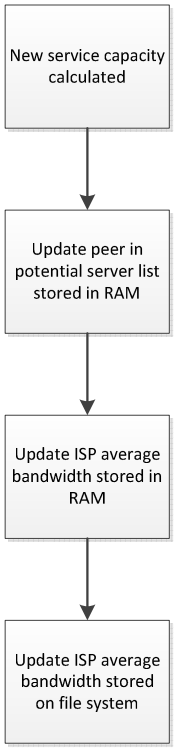

Figure 14 – Updating Service Capacity

Updating the average service capacity for all peers that share the same ISP is done to help identify ISPs that are participating in P2P throttling. By updating the service capacity of peers with the same ISP potential poor performing peers are moved down in the sorted list, thus having less of a chance of being selected. Similar to prior works (Chiu & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013) the historic based strategy limits the amount of time spent with poor performing peers by using time connected rather than bytes received in order to determine when to replace peers. The

time interval used was 5 minutes with a micro adjustment to the server's service capacity at 1 minute interval in order to make a fair comparison with prior works.

```
Replace Worst Performing Peers
1.  Input: clientISP, numOfConnections
2.  Output: List of server peers
3.  sort(potentialServers)
4.  serverList = {0}
5.  //Replace worst performing peers of serverList
6.  FOR 0 to numOfConnections DO
7.      serverList.add(potentialServers [i])
8.       //Call the ISP Throttle Service Capacity
9.       serverList[size - 1].serviceCapacity = ispTSC(clientISP, potentialServers [i])
10. END LOOP
11. return serverList
```

Figure 15 – Replace Worst Performing Peers

In order to determine which server peers are in need of replacement this strategy begins by sorting the updated potential server list. The strategy then clears the current server list and repopulates this list with the top performing servers based on the server peer's service capacity. The replace worst performing peers algorithm is shown in Figure 15. Once the current server list has been repopulated the strategy continues downloading the requested resource. This strategy continues this behavior until the resource has been fully downloaded. The historic based strategy is shown in Figure 16.

```
    Historic Based Strategy
 1. Input: heterogeneity
 2. Output: downloadTime
 3. downloadTime = 0
 4. isp = getISP(selfIP)
 5. totalDownloaded = 0
 6. downloaded = 0
 7. serverList[numOfConnections] =
 8.      initilizeList(isp,numOfConnections,numSameISP)
 9. WHILE fileSize > totalDownloaded DO
10.     FOR EACH peer in serverList DO
11.         FOR 0 to 4 DO //loop for micro adjustments
12.             bandwidth = ispTSC(isp, serverList[i])
13.             //Update Server List Service Capacity and ISP Average
14.             serverList[i].serviceCapacity = bandwidth
15.             updateAllISP(serverList[i].isp, bandwidth)
16.             downloaded = bandwidth * 60
17.             IF ((fileSize - totalDownloaded) < downloaded)
18.                 downloadTime += (fileSize - totalDownloaded) / bandwidth
19.                 totalDownloaded = fileSize
20.                 Break Download Complete
21.             ELSE
22.                 downloadTime += 60
23.                 totalDownloaded += downloaded
24.             END IF
25.         END LOOP
26.         //Replace worst performing peers
27.         serverList = replacePeers(isp, numOfConnections)
28.     END LOOP
29. END LOOP
30. return (downloadTime / 60)
```

Figure 16 – Historic Based Strategy

Single Client without Competition

The single client experiments were performed using the single client simulation environment described above. This research evaluated four different strategies: the smart peer replacement, the smart peer replacement with choke, the historic based, and the

historic based with choke. The historic based with choke strategy consisted of the historic based strategy described in Figure 16 with the addition of the choking algorithm described by Lehrfeld & Simco (2010).

The source peer list service capacity template used is shown in Table 2. Similar to the prior works performed by Wilkins (2013) the following values were used as a template for additional nodes required for the parallel downloading experiments. These values were obtained from the preprocessed data and chosen to closely match the values used in prior works for a fair comparison.

Table 2 – Average capacity of four source peers during single client experiments

	Simulation Runs							
Source Peers	1	2	3	4	5	6	7	8
C_1	185	167	145	113	82	46	31	18
C_2	196	196	180	167	166	149	149	145
C_3	199	216	221	229	238	248	252	270
C_4	216	226	266	292	328	338	361	389

Table 3 and 4 shows the IP address for each of the source peers that were used in the simulation. Table 3 contains the information for the first 4 simulation runs, and Table 4 contains the remaining 4 simulation runs. The individual peer's IP addresses were used to query the preprocessed data for the ISP and average service capacity information.

Table 3 – IP Address of four source peers during first 4 simulation runs

	Simulation Runs			
Source Peers	1	2	3	4
C_1	45.126.232.82	213.163.96.19	5.11.32.21	190.11.224.24
C_2	78.17.17.25	61.14.231.24	5.133.48.20	213.163.96.5
C_3	194.247.24.97	197.242.144.15	37.48.64.18	110.34.0.19
C_4	197.242.144.12	195.8.52.23	187.250.0.21	178.208.160.20

Table 4 – IP Address of four source peers during final 4 simulation runs

	Simulation Runs			
Source Peers	5	6	7	8
C_1	180.178.0.19	103.196.43.24	185.78.0.22	185.67.236.23
C_2	43.230.156.22	162.255.40.23	162.255.40.3	5.11.32.21
C_3	91.185.96.19	178.253.64.20	188.241.96.21	163.215.104.24
C_4	93.185.250.23	185.38.136.22	192.35.0.21	185.57.200.22

The IP Address used for the client peer was C_2(IpAddress) + 1, meaning that the client and server peer C_2 belonged to the same ISP.

Four separate single client without competition experiments were performed. The file size, number of parallel connection, and choke threshold remained the same as the prior work performed by Wilkins (2013). The parameters for the experiments performed can be seen in Table 5.

Table 5 – Single client without competition simulation parameters

		Simulation Parameters			
Experiment	File Size	Num of Connections	Choke Threshold	Num of Same ISP	ISP Throttling
1	150 MB	4	20%	2	Off
2	150 MB	6	20%	3	Off
3	150 MB	4	20%	2	On
4	150 MB	6	20%	3	On

Multiple Clients with Competition

The multiple client experiments were performed using the multiple clients with competition simulation environment described above. The same four strategies evaluated in the single client without competition environment were also evaluated in the multiple clients with competition environment. The simulation environment was configured to use the same values for the download ratio, and number of server peers as was used in prior

works (Chui & Eun, 2008; Lehrfeld, 2009; Wilkins, 2013). The potential source peer list consisted of 100 peers. These peer's average service capacities were obtained from the preprocessed data and were chosen to closely match the values of prior works. The average service capacities of the four groups used were 1 Mbps, 498 Kbps, 97 Kbps, and 57 Kbps. The number of peers in each group used was 10, 5, 65, and 20, respectively. Table 6 shows the IP address range that corresponds to each group. The client was assigned an IP address that was in the IP address range of 185.91.208.0/22.

Table 6 – Multiple clients IP Addresses

Simulation Parameters	
Group	IP Address
1 Mbps	108.171.128.0/21
498 Kbps	185.91.208.0/22
97 Kbps	37.34.96.0/22
57 Kbps	103.204.166.0/23

Six separate multiple clients with competition experiments were performed. The file size, number of parallel connection, and choke threshold remained the same as the prior work performed by Wilkins (2013). The parameters for the experiments performed are shown in Table 7.

Table 7 – Multiple clients with competition simulation parameters

	Simulation Parameters				
Experiment	File Size	Num of Connections	Choke Threshold	Num of Same ISP	ISP Throttling
1	150 MB	4	20%	2	Off
2	300 MB	6	20%	3	Off
3	3 GB	6	20%	3	Off
4	150 MB	4	20%	2	On
5	300 MB	6	20%	3	On
6	3 GB	6	20%	3	On

Experiment Environment

The algorithms and simulation environment described in this chapter were implemented using the C++ language. Microsoft Visual Studio 2012 was used as the development environment. The experiments described in this chapter were performed using an Asus G751J with an Intel i7-470HQ processor, 32GB of RAM, and running the Windows 10 operating system. The simulation environment's pseudo random number generator was seeded with a known value so that the same simulation environment could be created as many times as needed during this research. Each download strategy was performed 32 times in order to be consistent with prior works (Wilkins, 2013). The download duration and normalized standard deviation of each of the strategies were written out to a CSV file and imported into Microsoft Excel. This data was reviewed and used to create charts for the evaluation and comparison of these strategies. These charts are included in Chapter 4.

Summary

This research re-created and validated the prior work conducted by Chiu & Eun (2008) and Wilkins & Simco (2013). The results of the reproduced and validated work performed by Wilkins & Simco (2013) were used to compare the historic based peer selection strategy against. The simulation environment presented in the work performed by Chiu & Eun (2008) and used in the work performed by Wilkins & Simco (2013) was modified to use data obtained from OOKLA (2017) and MaxMind (2017) to populate the simulated server peers with an IP address and average service capacity. The results of the experiments described above are reported in Chapter 4.

The new historic based peer selection strategy described was based on the Wilkins & Simco (2013) smart peer replacement strategy. This new strategy used historic data that was collected outside of the P2P network to increase the likelihood that a good performing peer is selected. The data was stored locally on the client and did not require additional network traffic to be introduced into the P2P network when querying the data for information on the potential server peers. This strategy captured and logged the actual performance of the connected server peers, and it used this information in addition to the historic data to make informed decisions on which server peers the strategy would remain connected to and which to replace.

Chapter 4

Results

Introduction

This research investigated the use of service capacity and ISP metrics to improve peer selection. As stated in Chapter 1 the goal of this research was to further reduce the average download time for an individual peer in a stochastic P2P network. In order to achieve this goal a new biased based download strategy that did not introduce additional network traffic into the P2P network was created. This historic based strategy used advanced data obtained outside of the P2P network and stored locally to seed a list of potential server peers in order to make an informed decision on the initial peer selection. This strategy then monitored the connection, updated the locally stored data, and used this information to determine which peers were in need of replacement.

Prior works have shown that limiting the time spent downloading from a poor performing peer can decrease the average download time (Chui & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). In these works the chunk based random selection strategy was extended by using time rather than bytes downloaded to determine when to switch peers, monitored the service capacity after the connection was made, and used this information to determine if a peer was performing at an acceptable level (Chui & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). Unlike the historic based strategy described in this research these prior works did not use any advanced information to increase the likelihood that a good peer was selected.

Since P2P applications generally do not have information about other peers prior to a connection being made the advanced knowledge used in this research had to be acquired from a third party that collected the relevant information outside of the P2P network and this information had to be preprocessed. A P2P simulator based on prior works (Chui & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013) was modified to use service capacities, ISPs, and IP addresses obtained from the preprocessed data was developed. In addition this simulation was also modified to simulate the throttling of service capacity when downloading a file that crosses ISP boundaries. The simulation used in prior works used specific values to represent the heterogeneity and the performance of each potential server peer. The simulation used in this work used values from the preprocessed data that closely match the values reported in prior works.

The effects of heterogeneity in a P2P network and several methods for decreasing the average download time in P2P content delivery networks were described in Chapter 2. These methods were separated into two main categories: random selection and advanced knowledge selection. Random selection methods select server nodes randomly with equal probability in an attempt to reduce the average download time by reducing the amount of time spent downloading from a poor performing peer. The variations of the random selection methods described did not use any advanced knowledge to initially select peers. Instead they relied only on data captured after the connection had been made in order to determine the performance level of the server peer. The advanced knowledge selection strategies try to select good performing server peers by using some form of advanced knowledge, such as ISP (Bindal, et al., 2006; Steiner, & Varvello, 2011) and proximity information (Ying & Basu, 2006; Ijaz, et al., 2013) that was obtained before

making an initial selection in order to increase the chances of selecting a good performing peer. These methods did not take into account the stochastic nature of the network, and they did not monitor the service capacity after the connection was made. Since these methods did not have recent historic information about the connected peers they were unable to change peers in the event a poor performing peer was selected or a good performing peer's performance level degraded.

Chapter 3 described the new hybrid peer selection strategy that was developed in this research to achieve the goal of reducing the individual client's average download time by further reducing the amount of time spent with a poor performing peer. This strategy combined the use of advanced knowledge for peer selection with the monitoring and logging of connected peers' service capacities. At the end of each predetermined time period the potential server peer list is sorted by service capacity and the top performing peers are selected to continue the download.

Chapter 3 also described the modified simulation environment used in this work, the methods used to preprocess the external data in order to create the advanced knowledge used, and the methods used to validate the simulation environment. The simulation used in this work employed a method to simulate cross ISP throttling by randomly assigning a bandwidth cap to each potential server peer. If the client peer's ISP differs from the server peer's ISP this method returns the minimum between the bandwidth cap and the server peer's service capacity; otherwise it returns the unmodified service capacity calculated from the AR-1 process. The experiments performed used the same parameters when possible as those performed in previous works with the addition of an ISP throttle parameter. The remainder of this chapter consists of the results from prior works used to

validate the simulation environment, the results of the single client with no competition, and the multiple clients with competition experiments performed in this investigation.

Simulation Validation

Prior to performing the experiments described in this work the simulation environment created was validated with re-created prior works. These strategies were executed in the modified simulation environment and the results compared against their reported results. The simulation environment consisted of both a single client environment and an environment representing multiple clients with service capacity values that were obtained from the preprocessed information as described in Chapter 3. The results shown in this section contain the average download duration in minutes of 32 runs for each of the tests conducted.

Single Client Environment

The validation of the simulation environment consisted of running the single client without competition experiment performed by Chiu & Eun (2008) and the single client without competition experiment performed by Wilkins & Simco (2013). This work implemented the chunk, parallel, permanent, and periodic strategies found in the work performed by Chiu & Eun (2008) and executed them in the modified simulation environment. The file size was 150 MB. These strategies used a single connection. The results from the Chui & Eun strategies were consistent with their reported results, showing that the periodic strategy performed consistent across all simulation runs while the other strategies' average download time increased as the level of heterogeneity was increased. The results of this validation are shown in Figure 17.

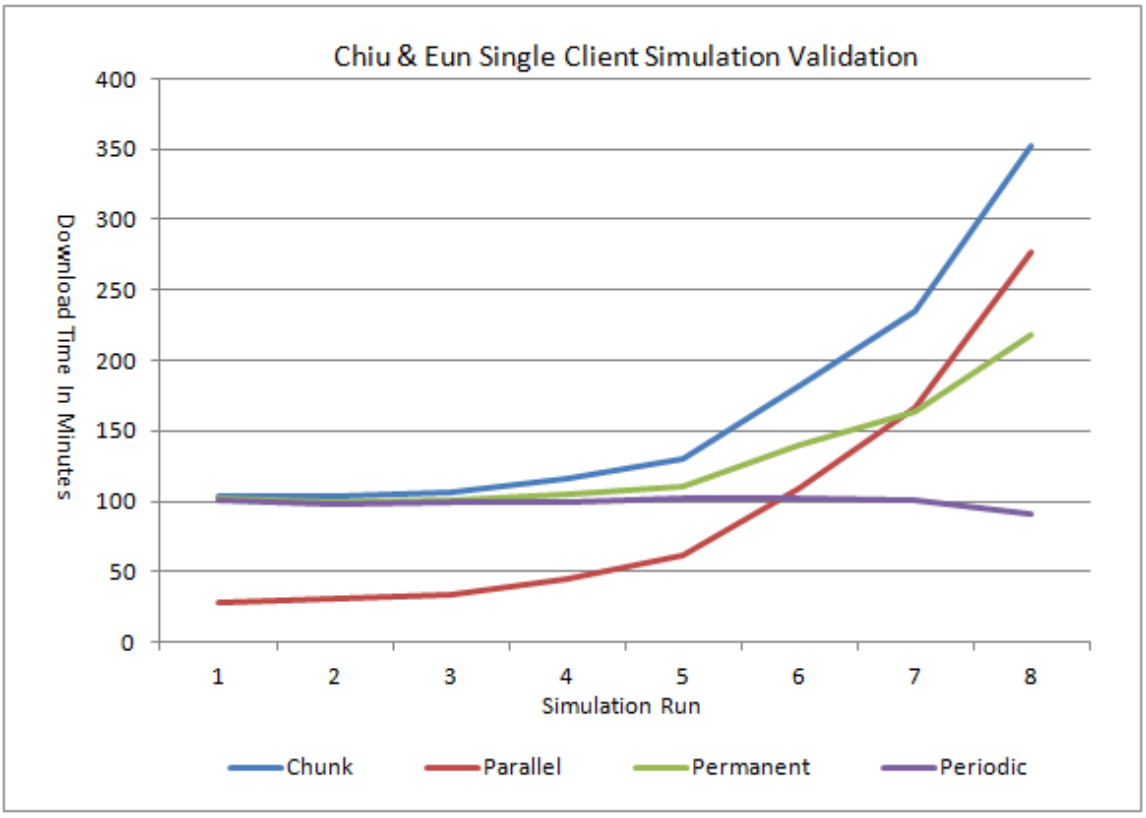

Figure 17 – Chiu & Eun Single Client Simulation Validation

This work also implemented the parallel periodic, parallel periodic with choke, smart peer replacement, and smart peer replacement with choke strategies found in the work performed by Wilkins & Simco (2013) and executed them in the modified simulation environment. The file size used was 150 MB, the choke threshold was 20%, and the number of connections used was 4. The results from the Wilkins & Simco strategies were consistent with their reported results, showing that the smart peer replacement and smart peer replacement with choke decreased the average download duration when compared with the parallel periodic and parallel periodic with choke. The results of this validation are shown in Figure 18.

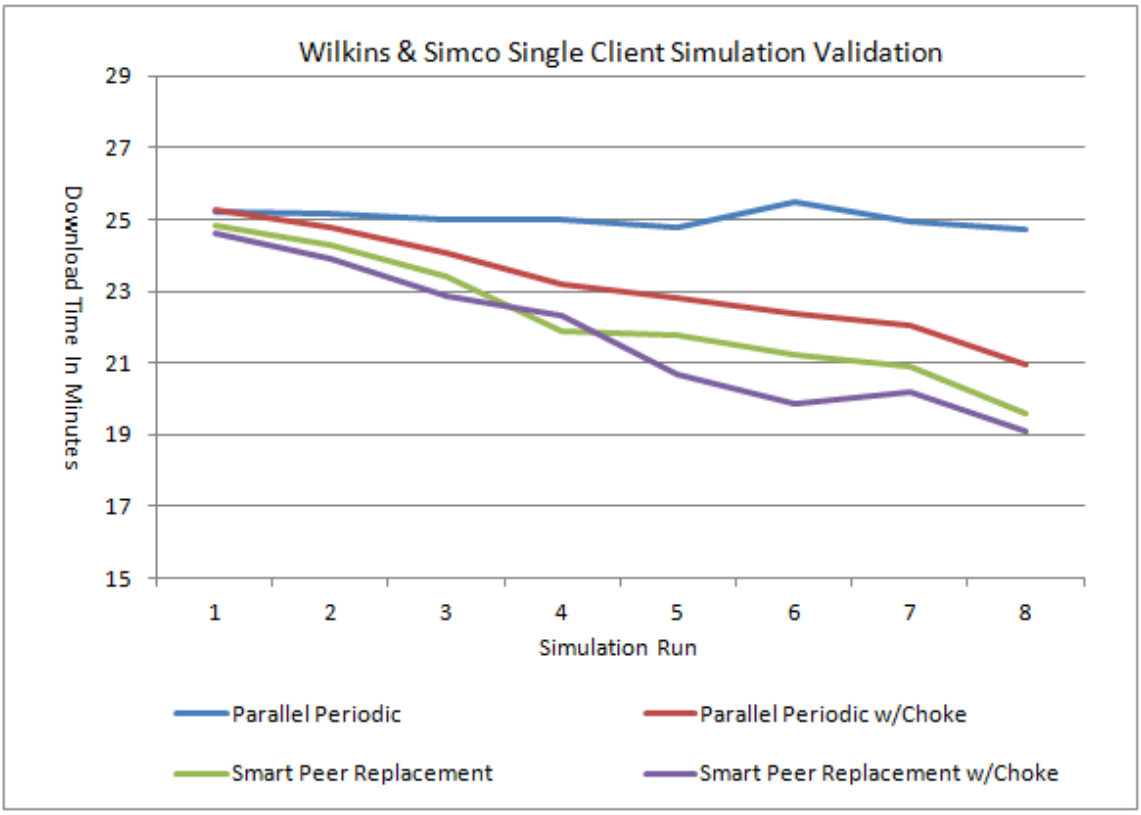

Figure 18 – Wilkins & Simco Single Client Simulation Validation

Multiple Clients Environment

The continued validation of the simulation environment consisted of running the multiple clients with competition experiment performed by Chiu & Eun (2008) and the multiple clients with competition experiment performed by Wilkins & Simco (2013). This work implemented the chunk and periodic strategies found in the work performed by Chiu & Eun (2008) and ran them in the modified simulation environment. The file size was 150 MB. These strategies used a single connection. The results from the Chui & Eun (2008) strategies were consistent with their reported results, showing that the periodic strategy consistently decreased the average download duration as the download ratio was increased when compared to the chunk based strategy. The results of this validation are shown in Figure 19.

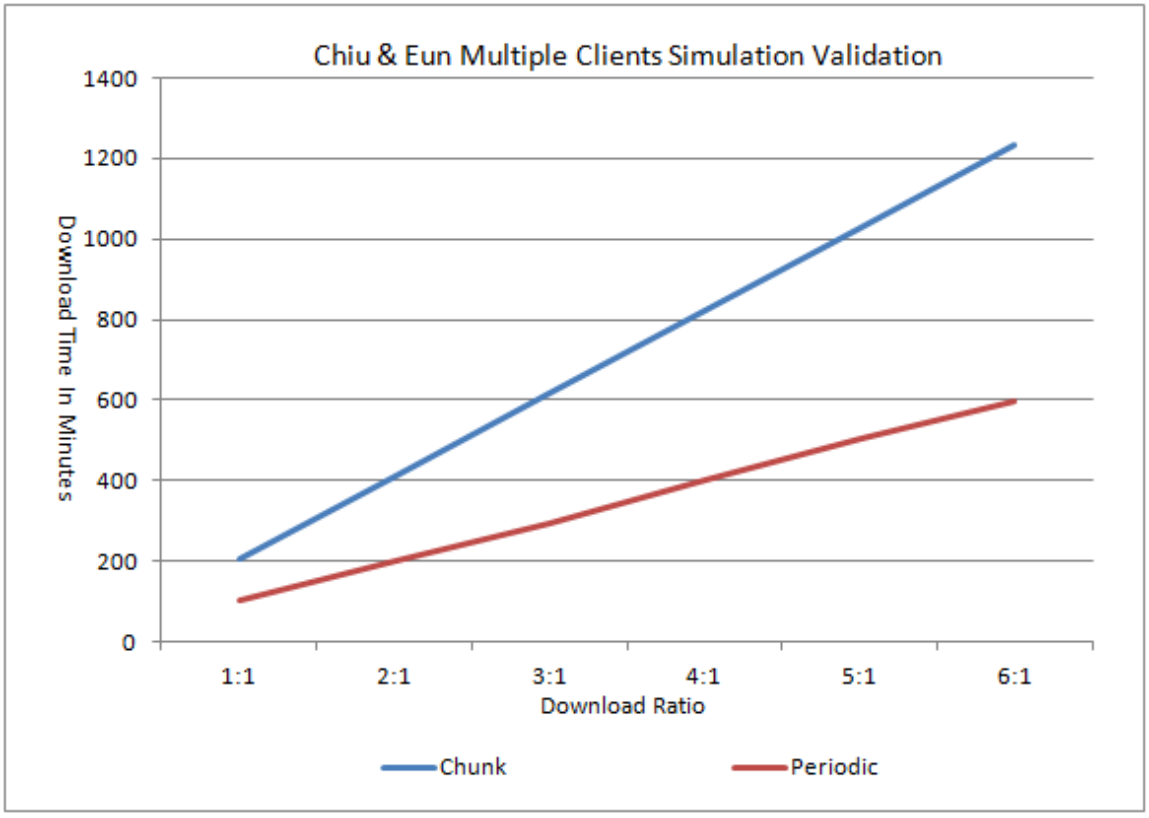

Figure 19 – Chiu & Eun Multiple Clients Simulation Validation

This work also implemented the parallel periodic, parallel periodic with choke, smart peer replacement, and smart peer replacement with choke strategies found in the work performed by Wilkins & Simco (2013) and ran them in the modified simulation environment. The file size was 150 MB, the choke threshold was 20%, and the number of connections was 4. The results from the Wilkins & Simco strategies were consistent with their reported results, showing that the smart peer replacement and smart peer replacement with choke improved upon the parallel periodic and parallel periodic with choke, further reducing the average download time across all of the download ratios tested. The results of this validation are shown in Figure 20.

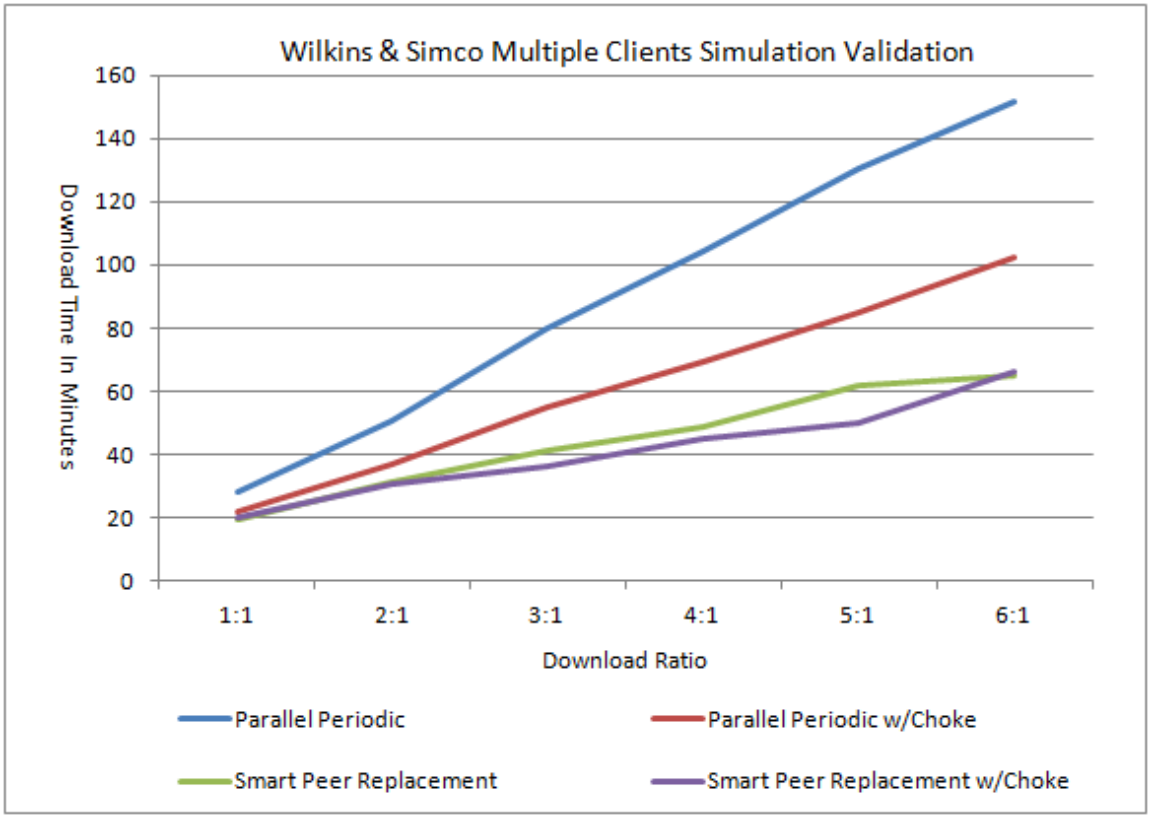

Figure 20 – Wilkins & Simco Multiple Clients Simulation Validation

Single Client Results

The single client without competition experiments performed in this research consisted of simulating the downloading of a file with four different download strategies as described in Chapter 3. The download strategies used from prior works were the Wilkins & Simco (2013) smart peer replacement and the smart peer replacement with choke that incorporated the Lehrfeld & Simco (2010) choke algorithm. The third strategy tested, described by this investigation, was the historic based strategy. Instead of randomly selecting a server peer during initial connection this strategy used advanced knowledge to look up the historic service capacity and ISP for the potential peer and selected the potentially best performing. After the connection was made the strategy monitored the connection and updated the locally stored service capacity. Finally instead

of replacing a predetermined subset of the server peers this strategy only replaced peers that were performing worse than the potential of the other peers in the server list. The forth strategy tested was the historic based with the inclusion of the Lehrfeld & Simco (2010) choke algorithm. This allowed the strategy to sever the connection to a peer in the event that its performance was below the choke threshold and replace it with the next best performing peer that was not currently connected. The single client experiment parameters used were shown in Table 5. These experiments consisted of downloading a 150 MB file with 4 connections and 6 connections, a 20% choke threshold, and ISP throttling on and off. The results of these experiments are shown in this section. All download duration values are an average of 32 runs.

Without ISP Throttling

Figure 21 shows the results of the single client without competition experiment with a file size of 150 MB, 4 parallel connections, a 20% choking threshold, and ISP throttling turned off. The results of the Wilkins & Simco (2013) strategies were consistent with their reported results. The results show that both the smart peer replacement and the smart peer replacement with choke strategies reduced the average download duration as the level of heterogeneity (Simulation Run) was increased. The smart peer replacement with choke strategy performed better overall than the smart peer replacement without choke strategy. The performance of both the historic based strategy and the historic based strategy with choke were very similar as the other two strategies when the heterogeneity of the network was low; however, as the heterogeneity was increased the average download duration decreased at a faster rate than that of the other two strategies. The historic based strategy showed an average 16.3% improvement over the Wilkins & Simco

(2013) smart peer replacement strategy, and the historic based with choke showed an average 14.27% improvement over the Wilkins & Simco (2013) smart peer replacement with choke strategy. The performance of the historic based and historic based with choke strategies were almost identical showing that the Lehrfeld & Simco (2010) algorithm had little effect on the average download duration when combined with the historic based strategy.

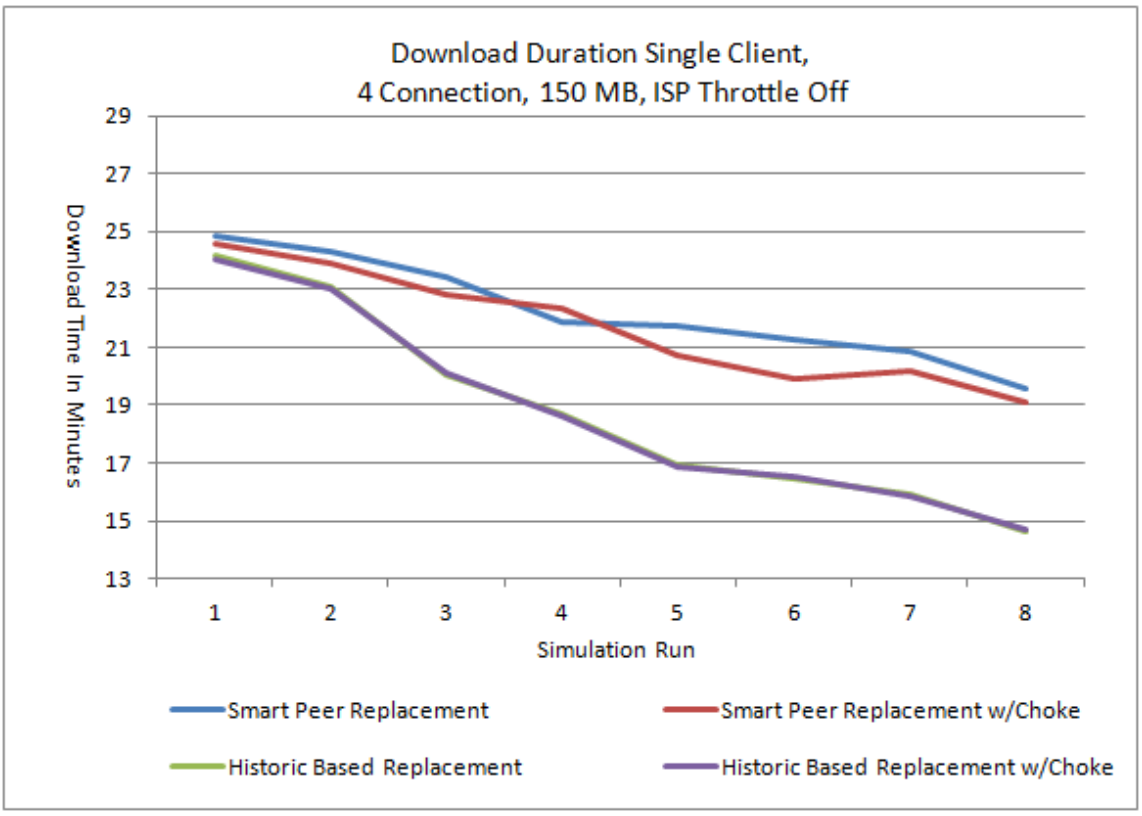

Figure 21 – Download Duration Single Client, 4 Connection, 150 MB, Throttle Off

Figure 22 displays the normalized standard deviation between each of the download sessions in this experiment. The lower the normalized standard deviation the more likely users are able to predict the amount of time it will take to download a file. The Wilkins & Simco (2013) strategies increased the normalized standard deviation as the heterogeneity

was increased. The strategies described in this work remained relatively flat, showing that the download duration was as predictable as or more predictable than prior works throughout the range of heterogeneity tested.

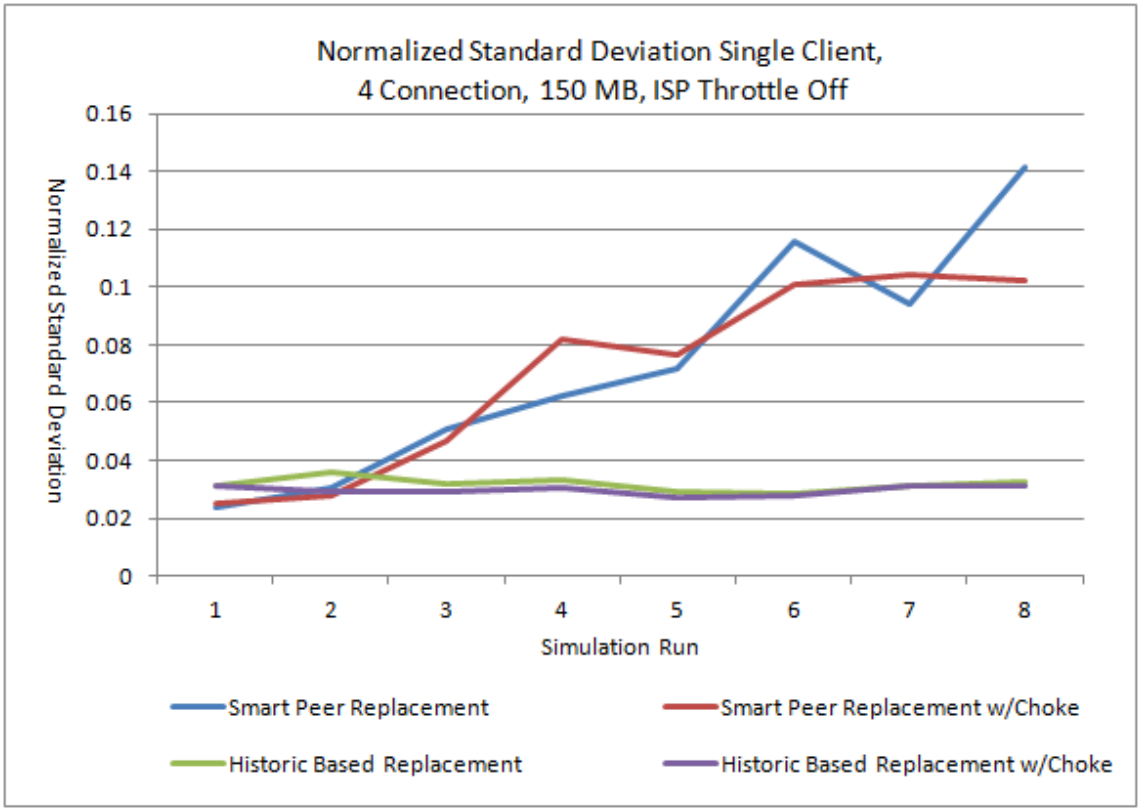

Figure 22 – Normalized Standard Deviation, 4 Connection, 150 MB, Throttle Off

The results of the single client six parallel connections were similar to the four connection experiment shown above. All four tested download strategies decreased the average download duration as the level of heterogeneity was increased. The historic based strategy showed an average 15.18% improvement over the smart peer replacement strategy, and the historic based strategy with choke showed an average 12.78% improvement over the smart peer replacement with choke strategy. These results are shown in Figure 23.

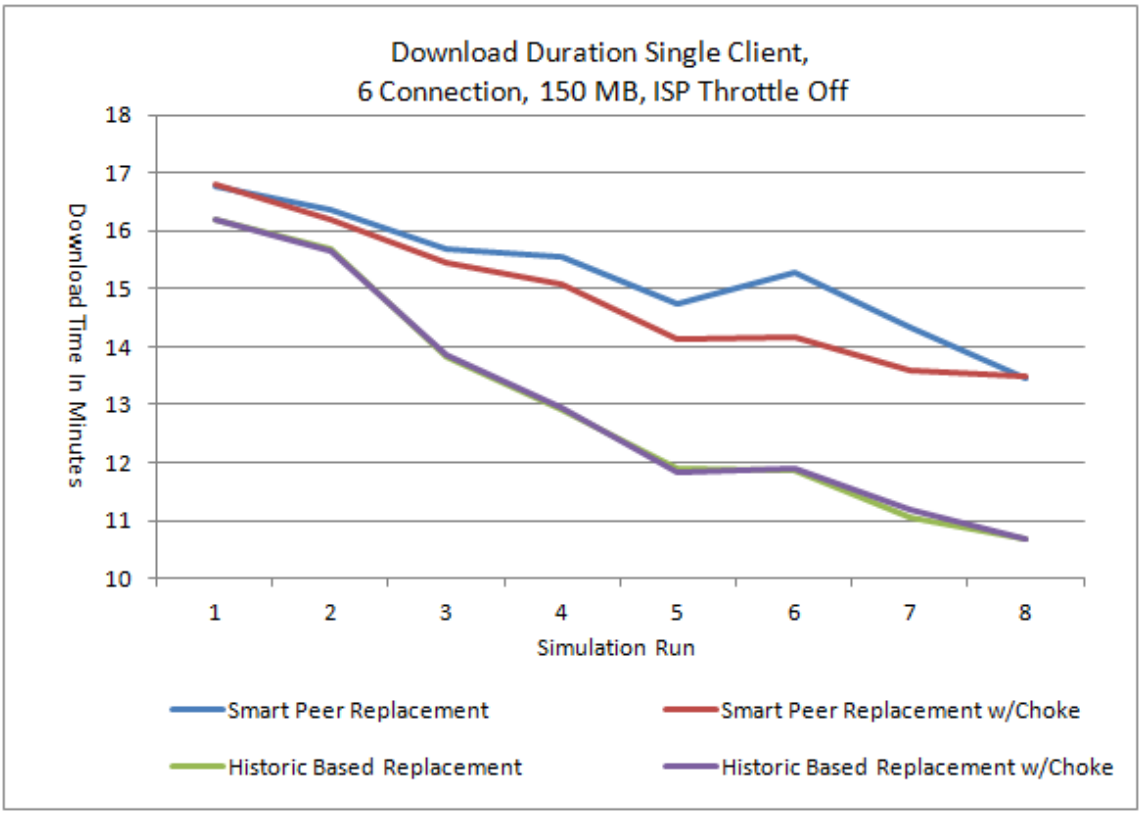

Figure 23 – Download Duration Single Client, 6 Connection, 150 MB, Throttle Off

The normalized standard deviation results of the single client six parallel connections experiment, Figure 24, are also similar to the four connection experiment shown above. As the heterogeneity increased the prior works (Wilkins & Simco, 2013) strategies steadily climbed, where as the historic based and historic based with choke remained relatively flat throughout the tested simulation runs.

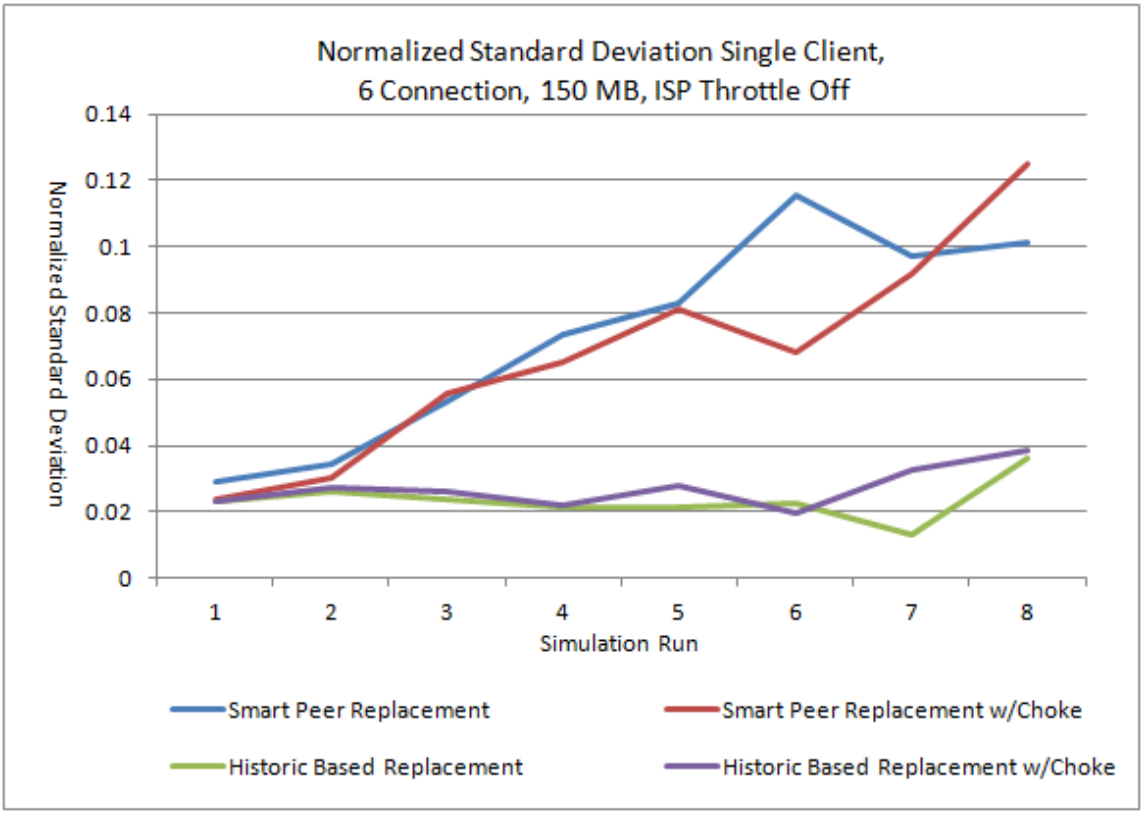

Figure 24 – Normalized Standard Deviation, 6 Connection, 150 MB, Throttle Off

With ISP Throttling

The same single client experiments shown above were also performed in a simulation environment where the ISPs were participating in ISP throttling. In order to simulate the ISP throttling each ISP was randomly assigned a bandwidth cap between 10 Kbps and 100 Kbps during the initialization of the simulation environment. When a new service capacity was requested the simulation environment returned the minimum between the bandwidth cap for the ISP that the server peer belonged to and the generated service capacity returned from the AR-1 process.

Figure 25 shows the results of the single client with ISP throttling experiment performed. This experiment used a file size of 150 MB, 4 parallel connections, a 20% choking threshold, and ISP throttling turned on. All four tested strategies' average

download duration increased as the simulation run increased. The Wilkins & Simco (2013) strategies both performed very close to one another. The strategies described in this work showed improvement over the other strategies across all simulation runs. The historic based strategy showed an average 18.1% improvement when compared to the Wilkins & Simco (2013) smart peer replacement strategy, and the historic based with choke showed an average improvement of 15.01% when compared to the Wilkins & Simco (2013) smart peer replacement with choke strategy.

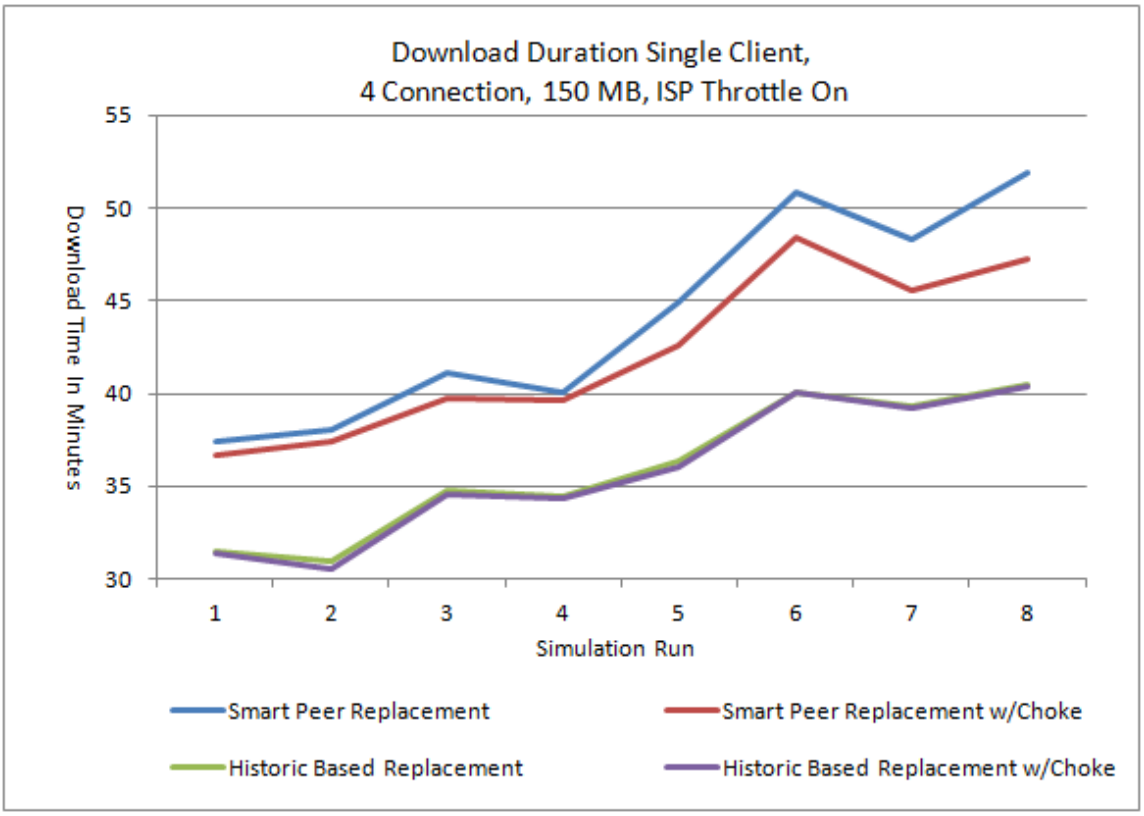

Figure 25 – Download Duration Single Client, 4 Connection, 150 MB, Throttle On

The normalized standard deviation results for each of the strategies tested showed that the strategies described in this work improved upon the Wilkins & Simco (2013) strategies by producing more consistent results. The results for the historic based strategy

had the lowest normalized standard deviation of the four tested strategies throughout the simulation runs. These results are shown in Figure 26.

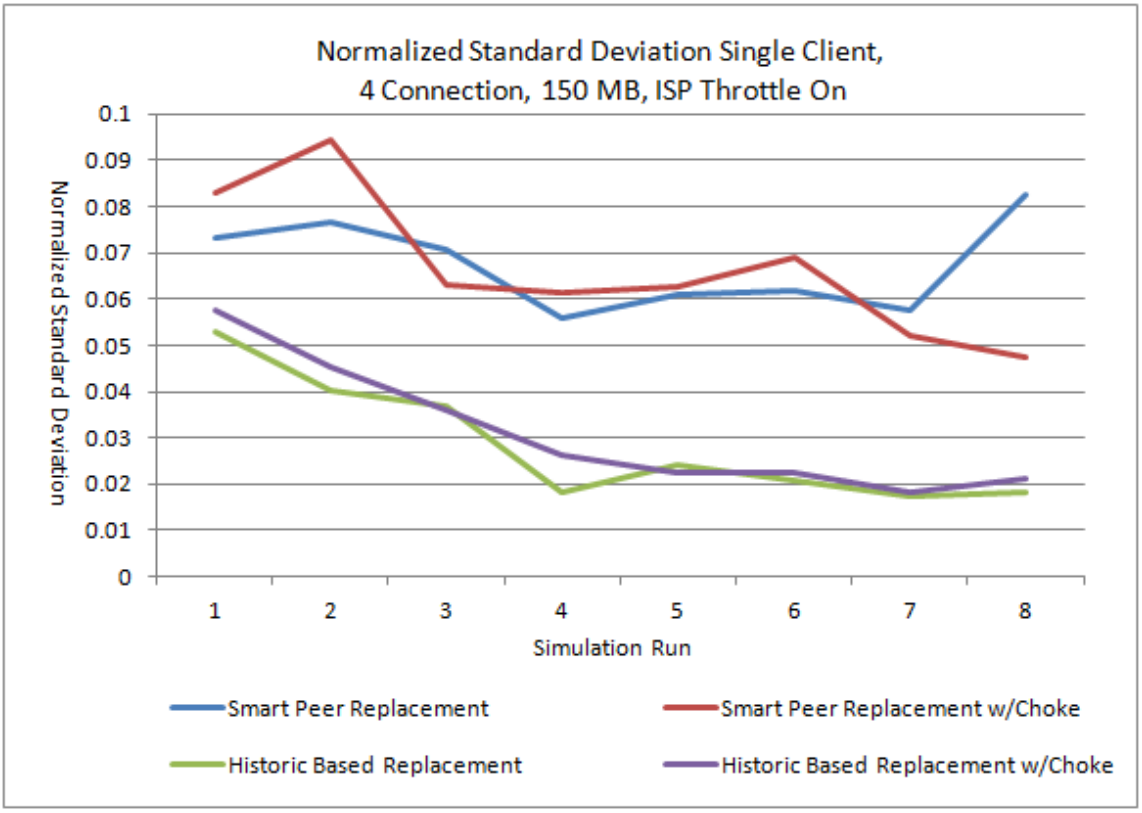

Figure 26 – Normalized Standard Deviation, 4 Connection, 150 MB, Throttle On

When the number of parallel download streams were increased the difference in the performance of the algorithms decreased. The overall result trend was similar to the four connection experiment described above. The four tested strategies increased in average download duration as the simulation run increased. The historic based strategy improved performance over the Wilkins & Simco (2013) smart peer replacement strategy by an average of 16.82%, and the historic based with choke showed an average 14.58% decrease in download duration when compared to the Wilkins & Simco (2013) smart peer replacement with choke strategy. These results are shown in Figure 27.

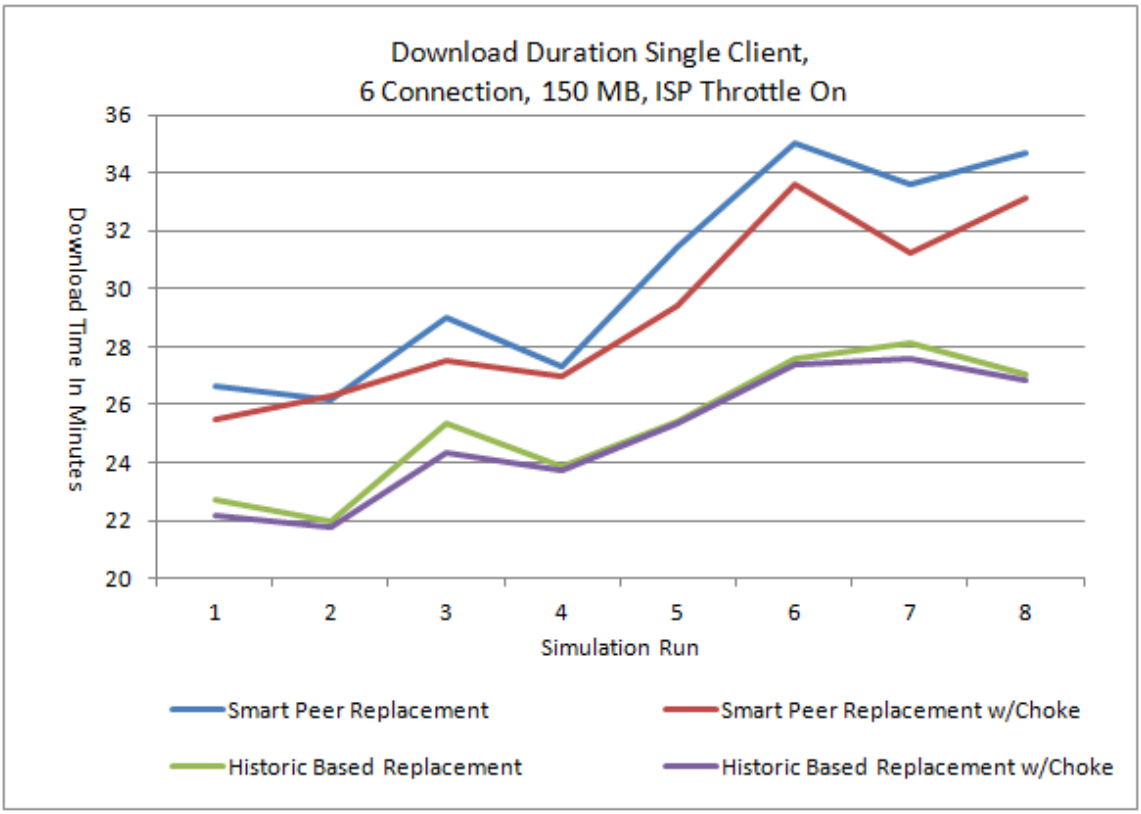

Figure 27 – Download Duration Single Client, 6 Connection, 150 MB, Throttle On

Figure 28 shows the normalized standard deviation for the single client six connections with ISP throttling experiment. The results of the historic based strategy had a lower normalized standard deviation than the other three tested strategies. When the level of heterogeneity was low the result are similar to the other algorithms tested, but as the level of heterogeneity was increased the normalized standard deviation decreased. Interestingly the historic based results were less erratic than the other three tested strategies.

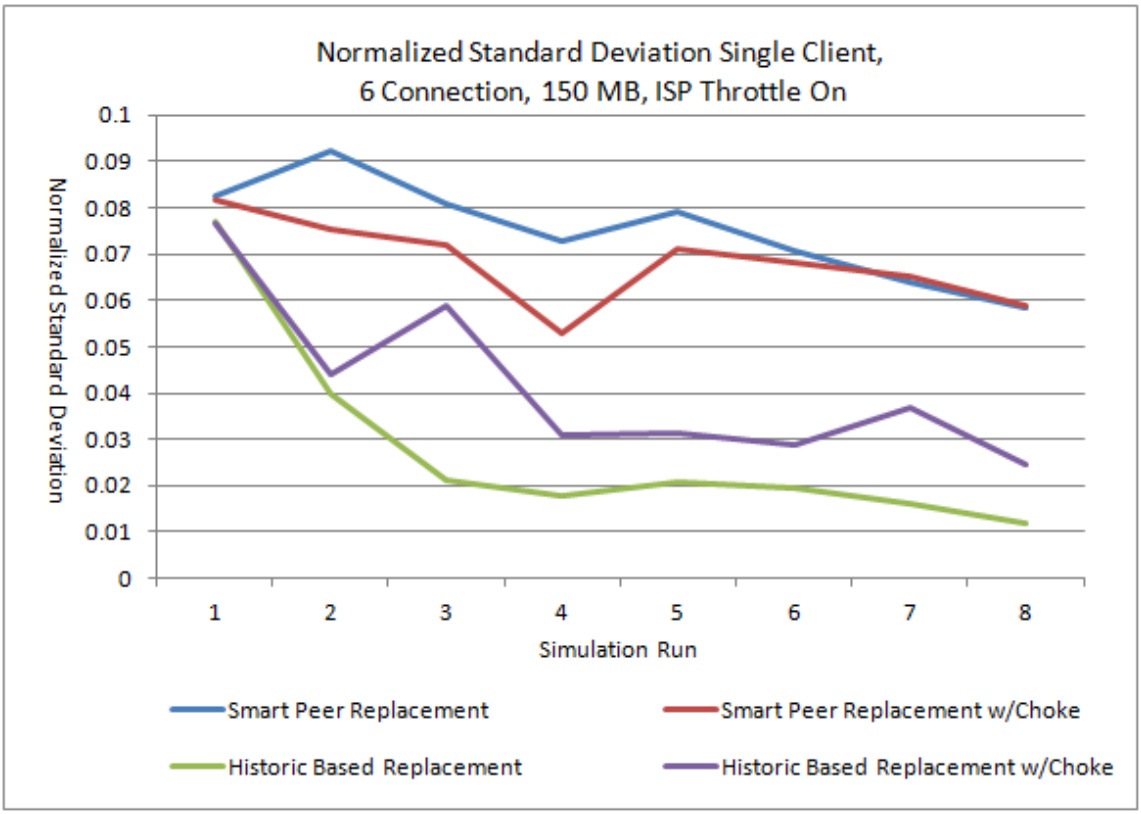

Figure 28 – Normalized Standard Deviation, 6 Connection, 150 MB, Throttle On

The single client without competition experiments performed showed that the historic based strategy described in this investigation decreased the average download duration for the individual client when ISPs are participating in ISP throttling and when no ISP throttling exists in the network. The addition of the Lehrfeld & Simco (2010) choking algorithm to the historic based strategy did not have significant effects on the download duration. In addition to download duration reduction the historic based strategy also showed a decrease in the normalized standard deviation when compared to prior works (Wilkins & Simco, 2013) across all experiments performed in the single client without competition simulation environment.

Multiple Clients Results

This section describes the results of the multiple clients with competition experiments performed in this research. These experiments used the multiple clients simulation environment described in Chapter 3. This simulation environment, based on prior works (Chiu & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013), consisted of 100 server peers and a variable amount of client peers ranging from 100 to 600 in 100 peer increments. This simulation environment used service capacity values that were obtained from the preprocessed information described in Chapter 3, and simulated ISPs participating in cross ISP throttling and no ISPs participating. The degree of heterogeneity of this simulated network was fixed throughout the downloader ratio range tested.

The multiple clients with competition experiments tested the same four download strategies that were tested in the single client without competition experiments. The parameters for this simulation environment were shown in Table 7 and consisted of downloading various file sizes with 4 connections and 6 connections, a 20% choke threshold, and ISP throttling on and off. The results of these experiments are shown in this section. All download duration values reported are an average of 32 runs.

Without ISP Throttling

Figure 29 shows the results of the multiple clients with competition experiment performed. This experiment had a file size of 150 MB, 4 parallel connections, a 20% choking threshold, and ISP throttling turned off. The results of the Wilkins & Simco (2013) strategies were consistent with their reported results. The results show that all four of the tested strategies' download duration increased as the level of competition

(download ratio) was increased. The smart peer replacement with choke strategy performed very similar to the smart peer replacement without choke, performing almost identical when competition was low and increasing as the download ratio was increased, then converging back when the download ratio was 6:1. The average download duration for the historic based and historic based with choke also increased as the level of competition was increased; however, these strategies showed a decreased average download time when compared to the other strategies. The historic based strategy showed an average 59.34% improvement over the Wilkins & Simco (2013) smart peer replacement strategy and the historic based with choke showed an average 56.18% improvement over the Wilkins & Simco (2013) smart peer replacement with choke strategy. These results show that the Lehrfeld & Simco (2010) algorithm did not significantly affect the average download duration when combined with the historic based strategy.

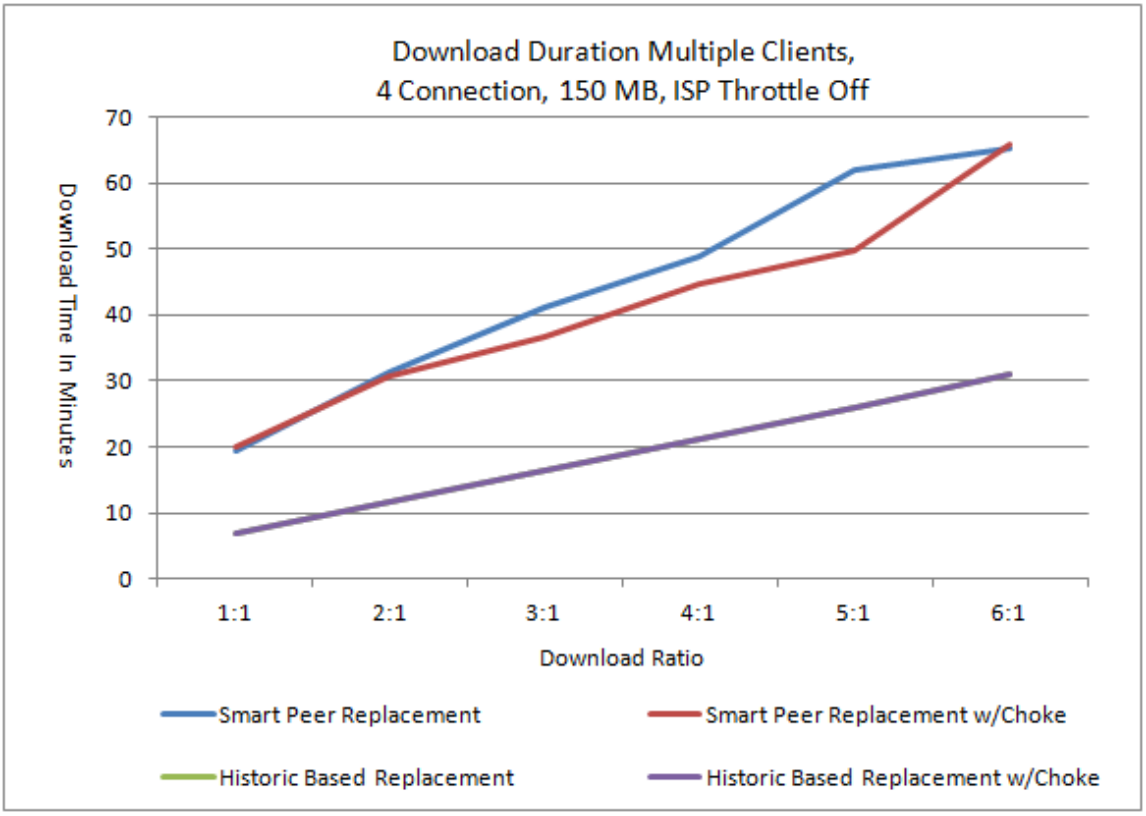

Figure 29 – Download Duration Multi-Client, 4 Connection, 150 MB, Throttle Off

The historic based with and without choke strategies' normalized standard deviation was more consistent and lower than the Wilkins & Simco (2013) strategies throughout all download ratios tested. Increasing the download ratio did not have a significant effect on the strategies described in this investigation. These results are shown in Figure 30.

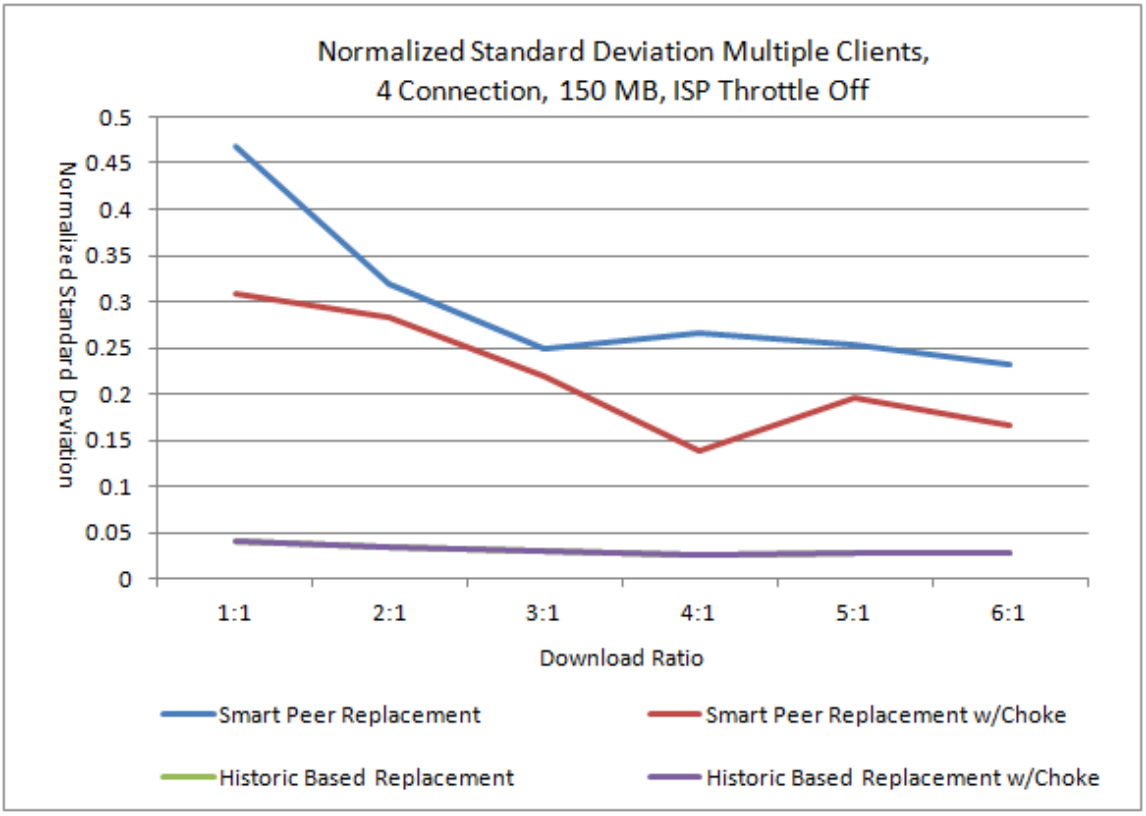

Figure 30 – Normalized Standard Deviation, 4 Connection, 150 MB, Throttle Off

When increasing the number of connections and the file size the results were very similar to the previous discussed results. All strategies' average download duration increased as the level of competition was increased. The historic based strategy showed an average 57.04% improvement over the Wilkins & Simco (2013) smart peer replacement strategy and the historic based with choke showed an average 53.62% improvement over the Wilkins & Simco (2013) smart peer replacement with choke strategy. These results are shown in Figure 31.

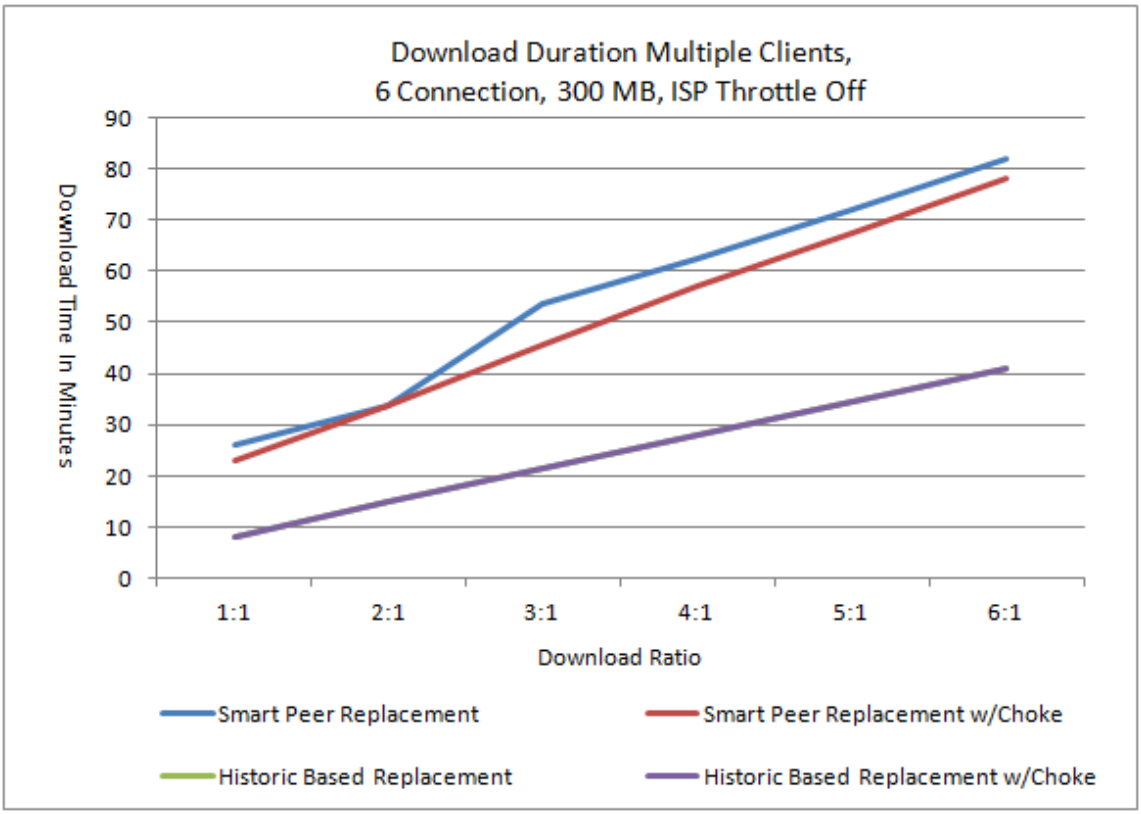

Figure 31 – Download Duration Multi-Client, 6 Connection, 300 MB, Throttle Off

The normalized standard deviation results for the 300 MB file size, six connection experiment was also similar to the 150 MB file size, previous experiment discussed. The historic based strategies produced a lower normalized standard deviation and performed more consistently than the smart peer replacement and the smart peer replacement with choke throughout all download ratios tested. These results are shown in Figure 32.

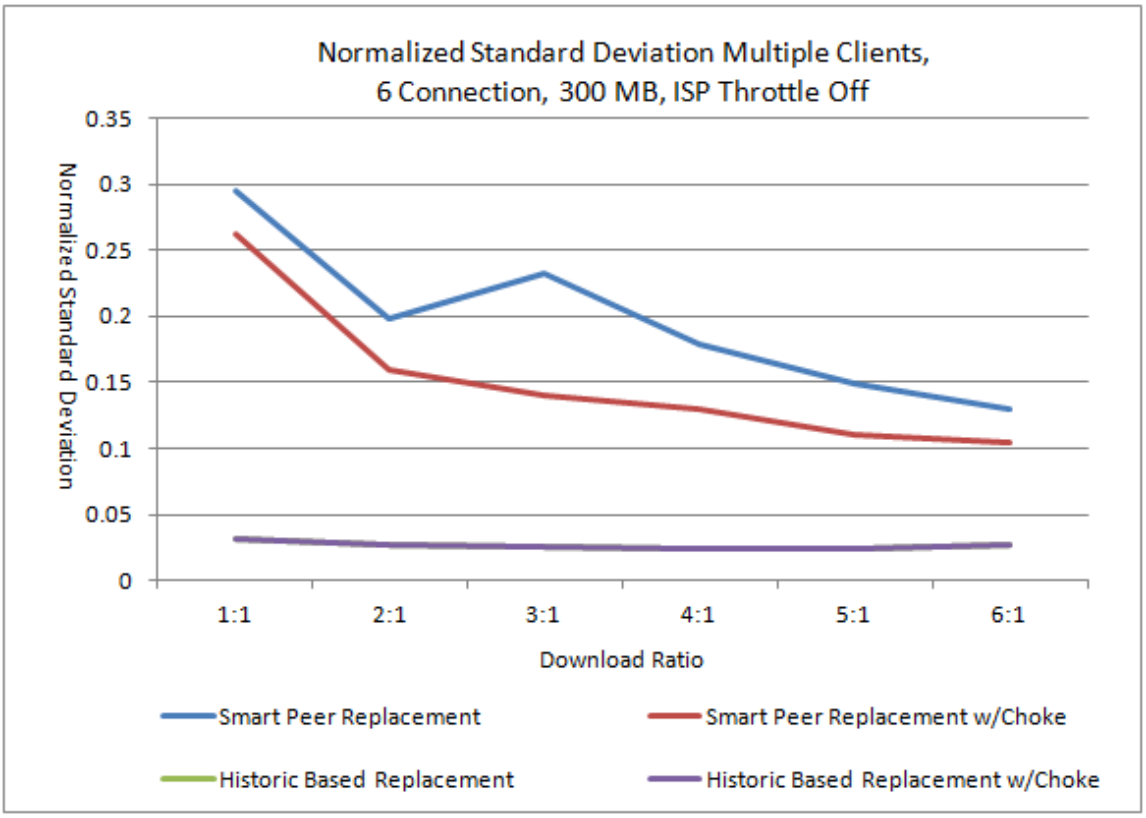

Figure 32 – Normalized Standard Deviation, 6 Connection, 300 MB, Throttle Off

Increasing the file size further continued to show the same trend as the other multiple clients with competition and no ISP throttling tests conducted. All four strategies' average download duration increased as the level of competition was increased. The strategies described in this investigation showed decreased average download duration when compared with the other two tested strategies. The historic based strategy showed an average 43.65% improvement over the Wilkins & Simco (2013) smart peer replacement strategy and the historic based with choke showed an average 40.03% improvement over the Wilkins & Simco (2013) smart peer replacement with choke strategy. These results are shown in Figure 33.

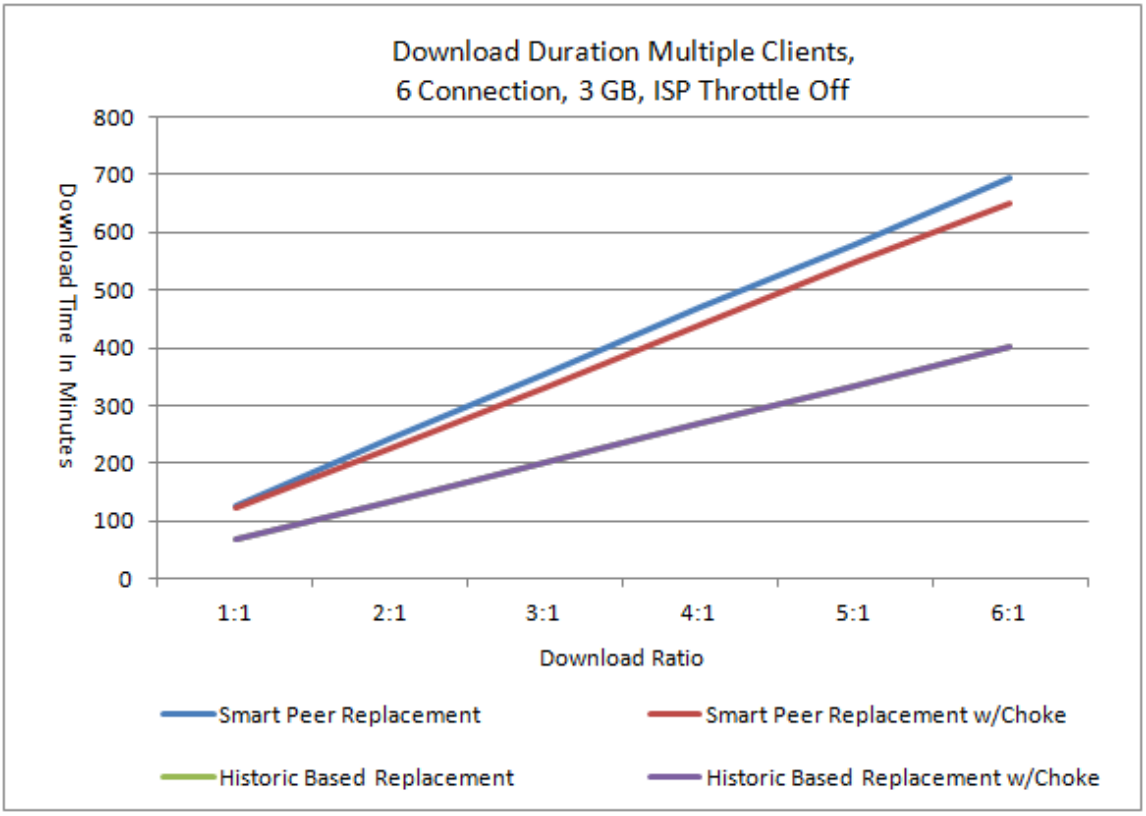

Figure 33 – Download Duration Multi-Client, 6 Connection, 3 GB, Throttle Off

The normalized standard deviation results, Figure 34, for the 3 GB file size, 6 connection experiment were also similar to the other multiple clients with competition experiments conducted. The historic based strategies produced a lower normalized standard deviation and performed more consistently throughout the range of download ratios tested than that of the Wilkins & Simco (2013) strategies.

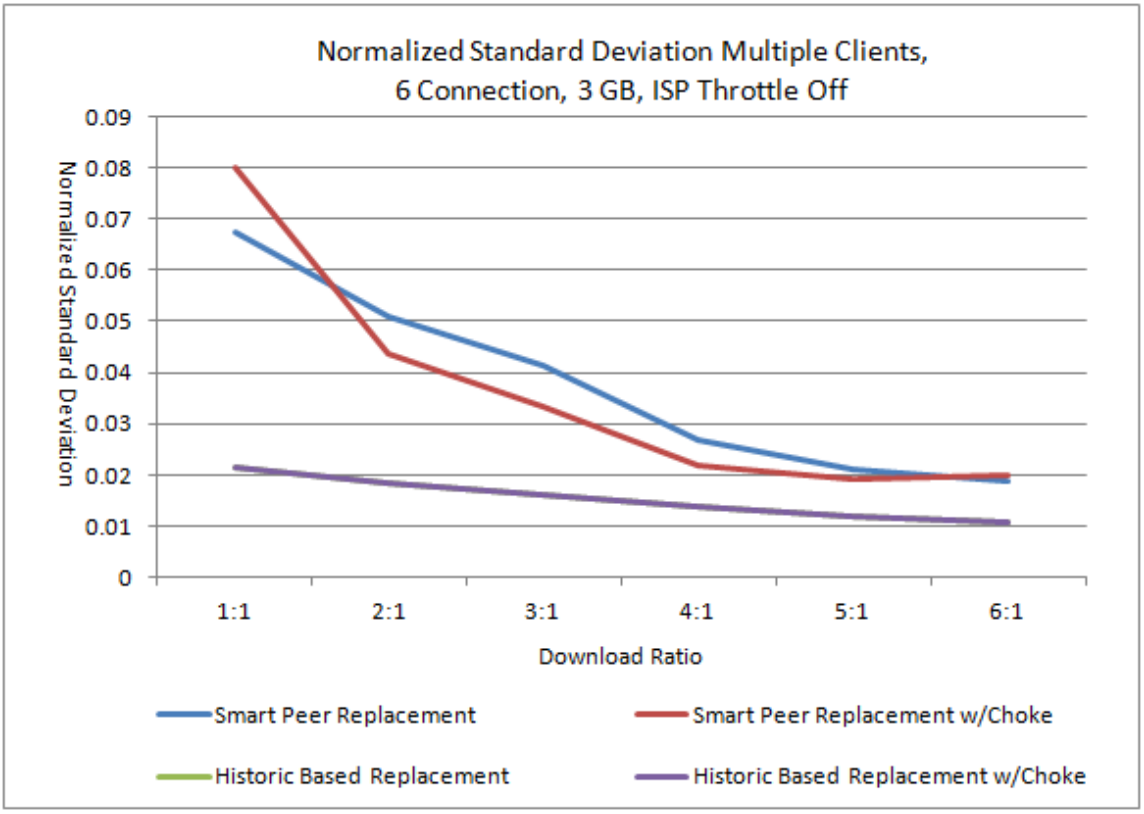

Figure 34 – Normalized Standard Deviation, 6 Connection, 3 GB, Throttle Off

With ISP Throttling

The same multiple clients with competition experiments shown above were also performed in a simulation environment where the ISPs were participating in ISP throttling. This simulation environment used the same ISP throttling algorithm as the single client with ISP throttling experiments performed in order to provide a reduced service capacity when the client peer's ISP did not match the server peer's ISP.

Figure 35 shows the results of the multiple clients with competition experiment conducted. This experiment had a file size of 150 MB, 4 parallel connections, a 20% choking threshold, and ISP throttling turned on. All four tested strategies' average download duration increased as the download ratio was increased. The Wilkins & Simco (2013) strategies both perform very close to one another. The strategies described in this

work showed improvement over the other strategies across all download ratios tested. The historic based strategy showed an average 65.57% improvement when compared to the Wilkins & Simco (2013) smart peer replacement strategy, and the historic based with choke showed an average improvement of 64.25% when compared to the Wilkins & Simco (2013) smart peer replacement with choke strategy. The addition of the Lehrfeld & Simco (2010) choke algorithm did not significantly affect the download duration of the historic based strategy.

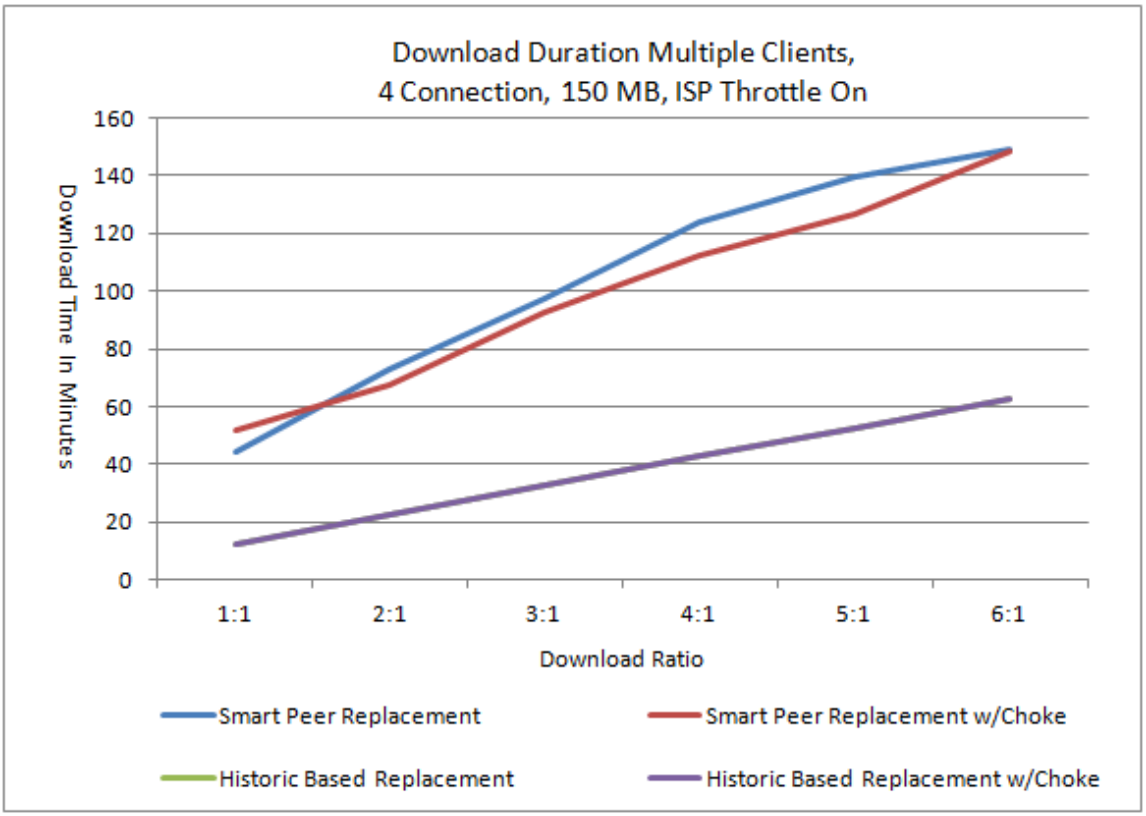

Figure 35 – Download Duration Multi-Client, 4 Connection, 150 MB, Throttle On

The normalized standard deviation results showed the strategies described in this work were lower and more consistent throughout the download ratio range tested than the Wilkins & Simco (2013) strategies. These results are shown in Figure 36.

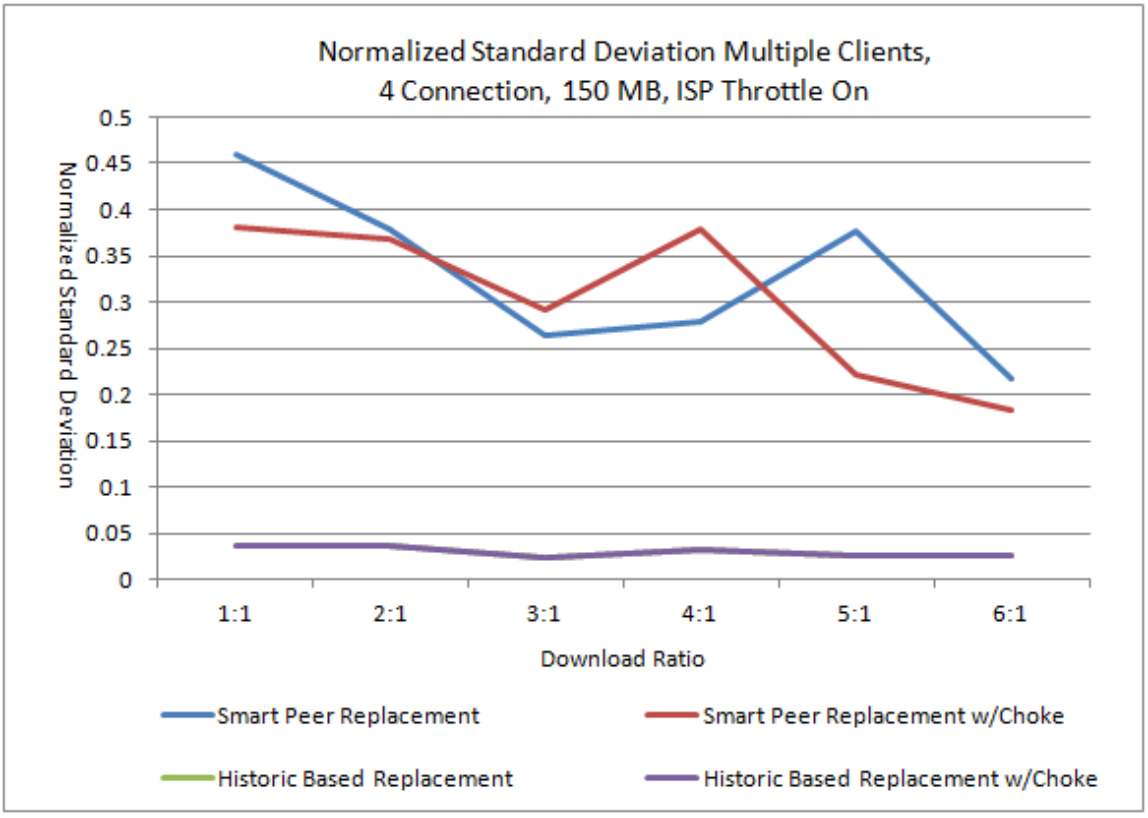

Figure 36 – Normalized Standard Deviation, 4 Connection, 150 MB, Throttle On

When the number of parallel download streams in the multiple clients with competition simulation and with ISPs participating in cross ISP throttling was increased the overall result trend continued to be similar to the other multiple client experiments described above. The four tested strategies increased in average download duration as the download ratio was increased. The historic based strategy improved performance over the Wilkins & Simco (2013) smart peer replacement strategy by an average of 56.28%, and the historic based with choke showed an average 55.7% decrease in download duration when compared to the Wilkins & Simco (2013) smart peer replacement with choke strategy. These results are shown in Figure 37.

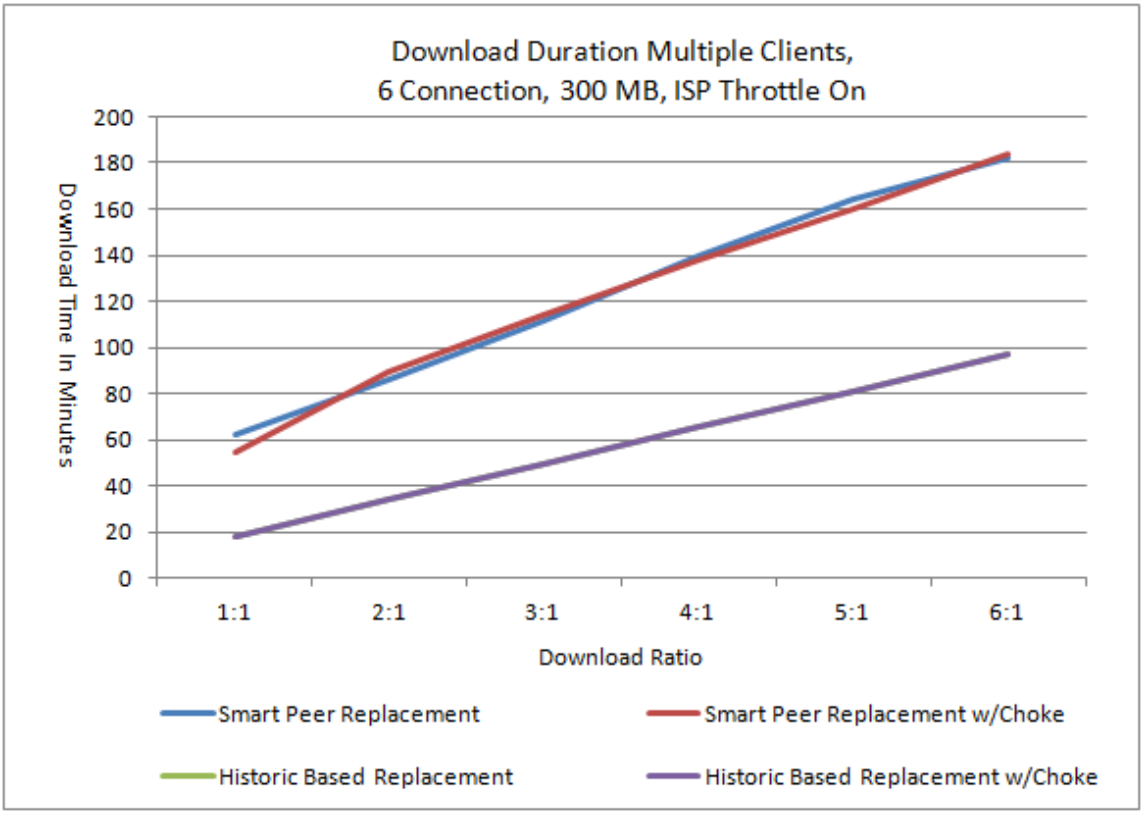

Figure 37 – Download Duration Multi-Client, 6 Connection, 300 MB, Throttle On

The normalized standard deviation results, Figure 38, show the Wilkins & Simco (2013) tested strategies normalized standard deviation decreased as the download ratio was increased while the historic based strategies remained relatively flat. The results for the historic based strategies were lower and more consistent throughout all download ratios tested.

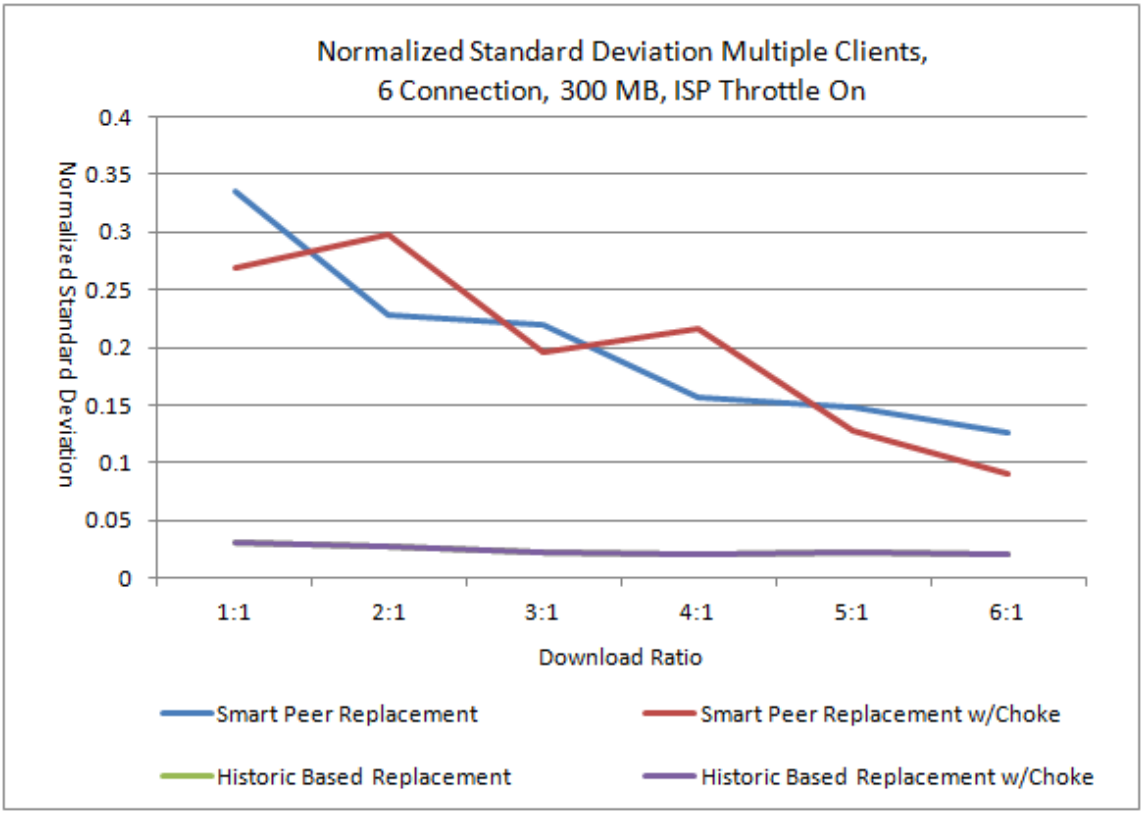

Figure 38 – Normalized Standard Deviation, 6 Connection, 300 MB, Throttle On

When the file size was increased to 3 GB with ISP throttling on the results continued to show the same trend as the other multiple clients with competition tests conducted. All four strategies' average download duration increased as the level of competition was increased. The strategies described in this investigation showed decreased average download duration when compared with the other two tested strategies. The historic based strategy showed an average 37.96% improvement over the Wilkins & Simco (2013) smart peer replacement strategy and the historic based with choke showed an average 36.84% improvement over the Wilkins & Simco (2013) smart peer replacement with choke strategy. These results are shown in Figure 39.

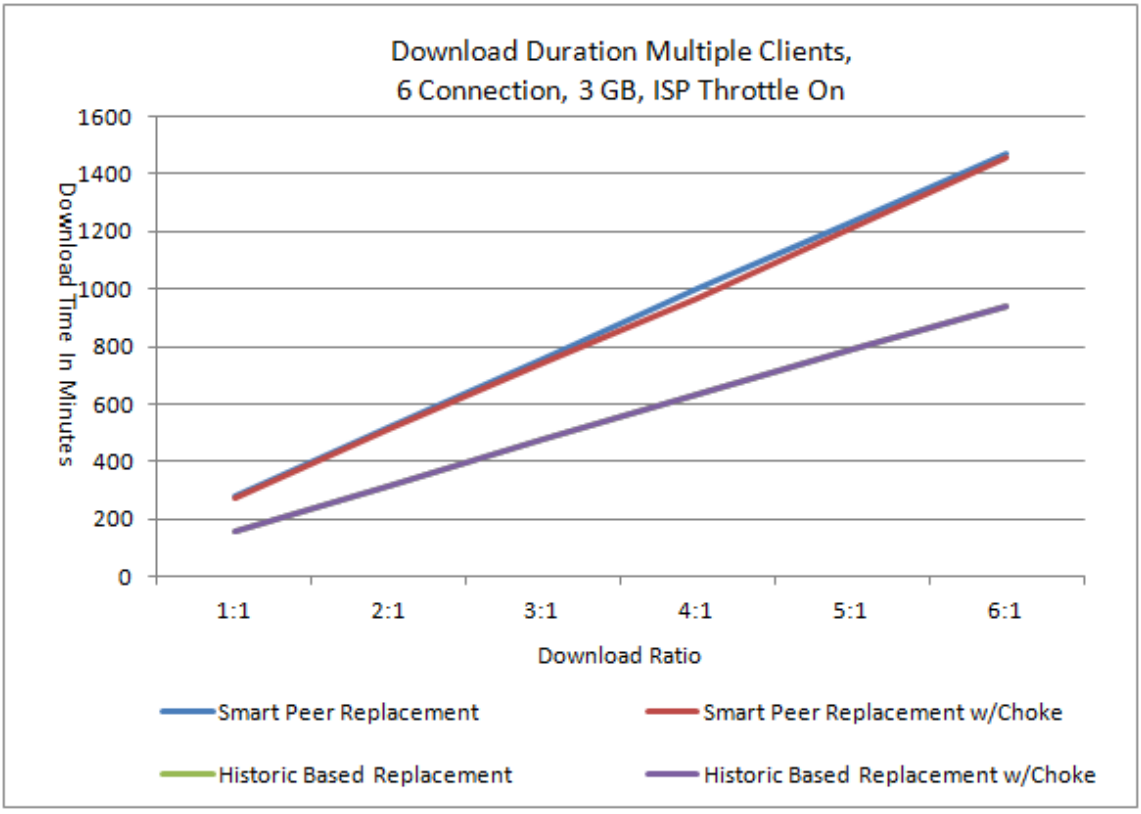

Figure 39 – Download Duration Multi-Client, 6 Connection, 3 GB, Throttle On

The normalized standard deviation results for the 3 GB file size, 6 connections, ISP throttling on experiment was similar to the other multiple clients with competition experiments conducted. The historic based strategies produced a lower normalized standard deviation and performed more consistently throughout the range of download ratios tested than the Wilkins & Simco (2013) strategies. These results are shown in Figure 40.

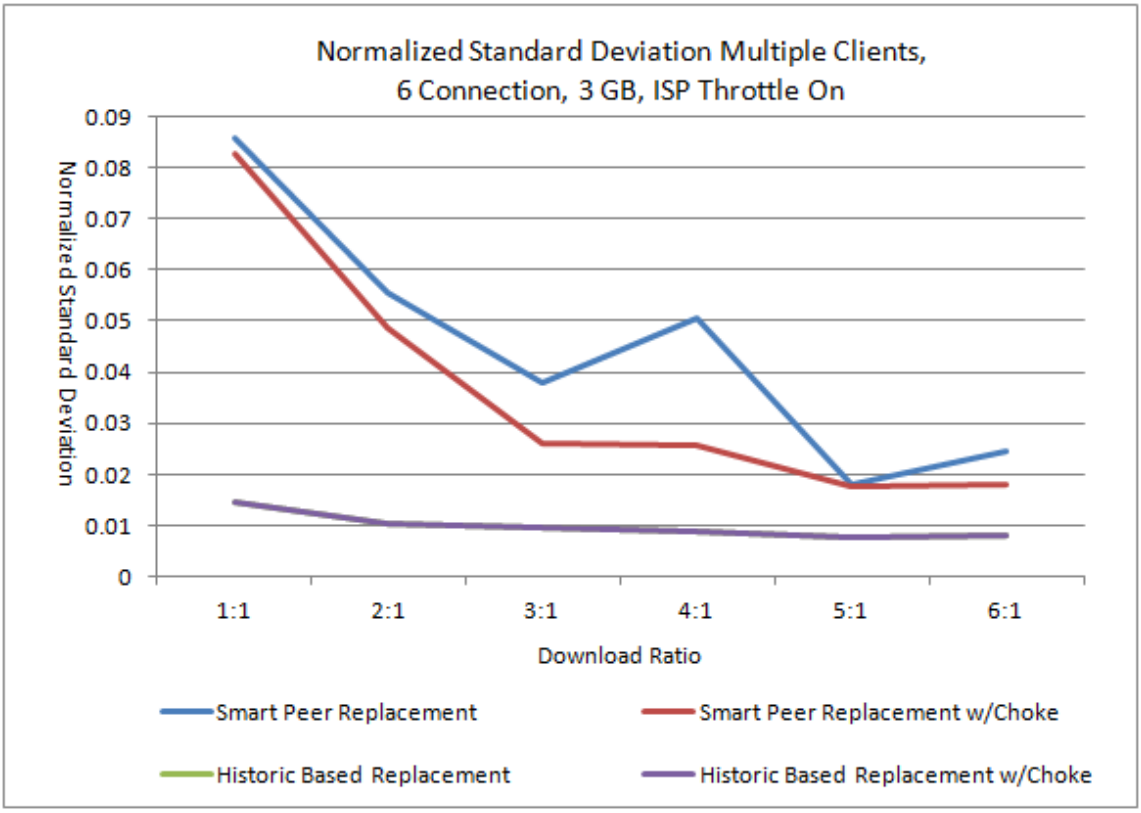

Figure 40 – Normalized Standard Deviation, 6 Connection, 3 GB, Throttle On

The multiple clients with competition experiments performed showed that the historic based strategy described in this investigation decreased the average download duration and the normalized standard deviation for the individual client when ISPs were participating in ISP throttling and when no ISP throttling existed in the network. Interestingly the addition of the Lehrfeld & Simco (2010) choking algorithm to the historic based strategy did not have significant effects on the download duration.

Summary

The results of the experiments performed in this investigation showed that modifying the random based strategy to use advanced knowledge for initial peer selection, monitoring the connection, logging the current service capacity, and only replacing the

worst performing peers reduced the individual client's average download duration in the simulated P2P network. The single client tests performed showed that the historic based strategy described in this research reduced the average download duration by an average 16.6% when compared to the Wilkins & Simco (2013) smart peer replacement strategy. The multiple clients with competition experiments performed showed that the historic based strategy improved the average download duration for the individual client by an average 53.31% over the Wilkins & Simco (2013) smart peer replacement strategy. Interestingly the percentage of improvement decreased as the file size was increased and as the level of competition was increased.

The historic based strategy results for both the single client without competition and the multiple clients with competition experiments performed showed that the normalized standard deviation was significantly reduced when compared to prior works (Wilkins & Simco, 2013) and remained relatively consistent throughout the download ratio range tested with the multiple clients with competition simulation. The results also showed that the addition of the Lehrfeld & Simco (2010) choke algorithm to the historic based strategy did not significantly change the recorded average download duration of the performed experiments.

In addition to download duration and normalized standard deviation the historic based strategies described in this investigation also reduced the average amount of cross ISP traffic during the download session when ISPs were participating in cross ISP throttling. In the single client with ISP throttling environment the historic based strategy reduced the cross ISP traffic by an average 55.17% when compared to the smart peer replacement strategy and the historic based with choke reduced the ISP traffic by an

average 57.12% when compared to the smart peer replacement with choke. In the multiple clients with ISP throttling environment the historic based strategy reduced the cross ISP traffic by an average 88.83% when compared to the smart peer replacement strategy and the historic based with choke reduced the ISP traffic by an average 88.73% when compared to the smart peer replacement with choke.

Chapter 5

Conclusions

Conclusions

The results of the research conducted during this investigation confirmed the hypothesis that the random based peer selection strategy can be improved with the use of advanced knowledge and only replacing the worst performing peers. The use of a small amount (< 20MB) of historic service capacity and ISP information allowed the new historic based peer selection strategy to make an informed decision on which of the available server peers could be good performers. The monitoring of the connection after a peer was selected allowed the strategy to capture the current performance of the server peer and log this information for future use. Finally the availability of both historic and recent service capacity information allowed the strategy to determine which peer's performance level was below the potential of another available peer and replace it. This historic based peer selection strategy improved both the average download duration and reduced the normalized standard deviation when compared to the Wilkins & Simco (2013) smart peer replacement strategy and the smart peer replacement strategy that incorporated the Lehrfeld & Simco (2010) choke algorithm. Since the advanced information used was obtained outside of the P2P network and stored locally the new strategy did not have the traditional disadvantage of adding additional traffic to the P2P network nor did it use additional tracker peers. The results of this peer selection strategy were compared with prior works (Wilkins & Simco, 2013) in a single client and a

multiple clients simulation environment that simulated ISPs participating in cross ISP throttling and without participation.

The combination of the Lehrfeld & Simco (2010) choke algorithm and the historic based strategy described in this research did not show significant effects on the results of the experiments performed in the single client simulation nor the multiple clients simulation. Interestingly the results did show that the percentage of improvement for the historic based strategy decreased as the level of competition was increased and as the file size downloaded was increased.

Limitations of the P2P Network Simulation

The simulation environment used in this research was based on prior works (Chiu & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013). It used an AR-1 process to simulate the server peer's service capacity fluctuation, and the simulation was extended in this research effort to also simulate cross ISP service capacity throttling. This environment was provided specific parameters, to configure the simulated P2P overlay, for each experiment conducted. This simulation environment did not accurately reflect real world P2P networks. The Internet is a dynamic network (Scandizzo & Imperiali, 2014); network nodes are constantly joining and leaving the network (Zuo & Iamnitchi, 2016). The simulated environment used in this work did not take this into account. The number of server peers and client peers remained static for the length of the experiment being conducted. The largest P2P overlay network simulated in this work consisted of 700 peers, 100 server peers and 600 client peers. Prior works have shown that real world P2P networks can be orders of magnitude larger, several hundred thousand peers (Grummadi, et al., 2003; Zuo & Iamnitchi, 2016). Finally the file sizes represented in this

simulation only represent medium to large files; it did not represent small files such as audio files which are typically under 10 MB in size and make up a significant portion of P2P traffic (Grummadi, et al, 2003; Lehrfeld, 2009; Wilkins, 2013).

Implications

Taking into account the large percentage of P2P traffic on the Internet this research has wide reaching implications to the field of P2P network based data file sharing. The ability to move data from one peer to another efficiently is crucial for the Internet to continue handling the increasing amount of P2P traffic. In addition the more efficient use of network resources allows for bandwidth to be freed quicker, reduced traffic congestion, rare file segments to be replicated quicker, and for a higher QoE for the users on the network.

The reduction of cross ISP traffic is highly beneficial to the ISPs since they are often charged for cross ISP traffic. In addition this reduction in cross ISP traffic would help reduce congestion at the ISP boundaries and allow for the continued growth of P2P traffic without the need for ISPs to participate in cross ISP throttling. This reduction in cross ISP traffic can be accomplished without additional cost or the participation of the individual ISPs.

The reduction in the normalized standard deviation between download sessions allows for both the network and client to better plan for the amount of network resources and time needed to complete a data transfer. Having a consistent rate of transfer also allows for the increased use of P2P networks for time sensitive transmissions such as live video streaming and voice over internet protocol (VoIP).

Contributions to the Field of Study

This research extended the random based strategy found in prior works (Lehrfeld & Simco, 2010; Wilkins & Simco, 2013) to a strategy that used a combination of historic service capacity and ISP information in order to improve peer selection. This modification showed reduced download duration and increased consistency between download sessions in the simulated P2P content delivery network. The benefits of the research conducted are not limited to P2P overlays that match the experiments performed. This work is applicable to overlays of all sizes and a full range of file sizes and streams. While this research effort specifically targeted P2P content delivery networks the new peer selection technique is applicable to any P2P network where the quality of peers are not the same, e.g. a heterogenic P2P overlay on a wide area network (WAN).

Advancement of Knowledge

This research advanced the knowledge of peer selection algorithms in a P2P overlay network. This work combined the use of advanced knowledge to aid in peer selection prior to the connection and the monitoring of the connected peer's performance after the connection to aid in good peer selection. The advanced information used in this work was obtained outside the P2P network and stored locally. This eliminated the need for additional network traffic to gain the needed information and it eliminated the need for additional peers to store the data. The monitoring and logging of the server peer's current service capacity allowed the new strategy to have the information necessary to determine if a connected peer was performing below the potential level of other peers and replace it. The historic based peer selection strategy extended the smart peer replacement strategy

presented by Wilkins & Simco (2013). This approach of using advanced information stored locally, monitoring the service capacity after connection, logging this information, and only replacing the worst performing peers demonstrated increased performance, reduced normalized standard deviation between download sessions, and reduced the amount of cross ISP traffic when compared to the prior works. In addition this research also showed that adding the Lehrfeld & Simco (2010) choke algorithm to the new strategy did not significantly change the results of the base strategy.

Recommendations for Future Work

The research reported in this work extended the body of knowledge in the area of peer selection strategies in P2P overlay networks. This work was built upon many prior contributions made by other researchers. While this investigation advanced the body of knowledge there is additional research to be conducted. It is likely that additional work will continue to improve on the work conducted in this investigation and continue to advance the body of knowledge.

Advanced Knowledge

This investigation used advanced knowledge that contained service capacity information for ISPs. In order to not introduce additional network traffic into the P2P network this information was stored locally on the client causing additional memory overhead when compared to prior works. It is likely that different data aggregation and the use of file compression techniques such as the Lempel-Ziv-Markov chain algorithm (LZMA, 2017) would result in less memory required for storage.

This investigation focused on only a few of the available metrics contained in the data obtained. It is likely that the other metrics available such as the age of the data and latency measured can be used to increase the likelihood that a good performing peer is selected thus further reducing the time spent with a poor performing peer (Fernando & Keppetiyagama, 2013; Wilkins, 2013).

Additionally this investigation used the same average service capacity across all same ISPs regardless of location, e.g. a query for an AT&T IP Address would return the same service capacity regardless of where in the world the physical machine is located. It is likely that using geographic location such as city, zip code, and country, in combination with the ISP would increase the likelihood of the stored service capacity accurately reflecting the true service capacity for the potential peer.

File Size

This research used prior works (Chui & Eun, 2008; Lehrfeld & Simco, 2010; Wilkins & Simco, 2013) as the base for the experiments performed. The file sizes used were 150 MB, 300 MB, and 3 GB. The experiments performed did not cover small files (< 10 MB) nor did they cover extremely large files (> 1TB). The historic based strategy results showed that the percentage of improvement over prior works decreased as the file size was increased. Does this trend continue, or is there a file size where the percentage of improvement levels out? If the improvement trend does not level out at what point does the prior work strategies and the historic based strategy converge? Is there a file size point, both small and large, where it makes sense to switch strategies during the download session? It is likely that a strategy consisting of multiple different peer selection strategies triggered by byte size transferred or some other metric would further

reduce the average download duration when tested with a wide range of file sizes, e.g. 1 MB to 1 TB.

Cross ISP Traffic

The newly created peer selection strategy was designed to further reduce the individual client's download duration. It was not specifically designed to reduce cross ISP traffic. It is likely that adjustments to the strategy would further reduce the cross ISP traffic produced when running this strategy. The historic based peer selection strategy initially tries to select half of its server peers from the same ISP as the client. After the initial connection is made there is no attempt to maintain same ISP servers. Prior work has shown that maintaining a level of same ISP peers throughout the download session can decrease the amount of cross ISP Traffic (Bindal, et al., 2006; Steiner & Varvello, 2011). Modifying the ISP connection management from only looking at initial connection to a method to one that attempts to maintaining a level of same ISP peers throughout the file transfer may further reduce the cross ISP traffic.

Real World Testing

A simulation provides a consistent, predictable, and repeatable environment to accurately compare different peer selection strategies. However, the performance recorded in the simulation environment does not necessarily translate to an active dynamic network. The only way to show how the strategy will perform in a real world active network is to implement the strategy in live applications and record the performance.

Other Applications

The techniques developed in this investigation are not limited to P2P content delivery networks. They are applicable to any situation where server nodes are selected from a list of potential nodes. Historic information has been shown to improve Cyber Foraging (Ou, Yang, & Zhang, 2006) and research has shown peer selection significantly affects P2P video streaming (Traverso, et al., 2015) and P2P VoIP (Marchetto, et al., 2011). It is likely applications such as Cyber Foraging, video streaming, and VoIP would benefit from the techniques discussed in this research.

Summary

Over recent years P2P networks have become a popular method for file sharing, instant messaging, video sharing, and streaming content. Since peers can act as both servers and clients simultaneously P2P networks are inherently scalable and able to overcome some of the traditional problems found in the client server model. P2P traffic currently represents a significant percentage of the total bandwidth being used on both local area networks (LAN) and the Internet. Because of their significant bandwidth usage P2P networks can negatively affect other applications on the network. Increasing their efficient usage of network resources has become important to both network users and network providers.

Recent research has shown that service capacity is not constant throughout a download session. Instead it fluctuates, and this fluctuation needs to be taken into consideration for download duration reduction. High latency, long distance, multiple hop connections with peers, cross ISP traffic, and network congestion can all have significant

negative effects on the level of service capacity. This service capacity fluctuation has been shown to cause previously good performing peers to become poor and vice versa.

There has been significant research focusing on reducing the average download duration for the individual client. The traditional approach has been to break the file up into equal chunks, randomly select a server peer from a list of available peers, download a chunk, and randomly select a new server peer. This process continues until the entire file has been downloaded. The random selection of a new peer after each chunk obtained attempts to minimize the negative effects of any one poor performing peer on the total download duration. While the byte based strategy increased performance over randomly connecting to a server peer and remaining connected for the entire download duration, recent research has shown that using time connected rather than content downloaded as a metric to determine when to select a new peer reduced the time spent with poor performing peers and reduced average download duration.

The random based peer selection strategy was extended by keeping track of the performance of the peers after the connection had been made and using this information to only replace the worst performing half. This method showed reduced time spent with poor performing peers and further reduced the time needed to complete a download session. This method showed that recent service capacity history can be used as a metric to determine a peer's potential performance. This method also showed that information for every peer in the P2P overlay is not necessary to be useful. Since the random peer selection strategy randomly selects peers it is possible to connect to a poor performing peer and spend time connected until the client can switch to a new randomly selected

peer. The random based techniques do not use advanced knowledge that could aid in good peer selection.

Other researchers have focused on the use of advanced knowledge to aid in peer selection. This advanced knowledge contains information that can be used to determine the potential of a server peer such as service capacity, location, ISP, and hops. Traditionally the use of advanced information increases overhead in the P2P network by polling peers for useful information and the use of additional tracker peers to store it. These biased based strategies examine the advanced knowledge, select a potentially good performing peer, connect to that peer, and download the requested file. These strategies do not monitor the service capacity after the connection has been made and are unable to react in the event a poor performing peer was selected or a good performing peer degrades. These methods have shown that having information about potential peers before a connection has been made can be used to make an informed decision on which peers might be good performers. They have also shown that non service capacity information such as the peer's ISP can be used as a metric to help determine a peer's quality.

The approach to peer selection described in this research built upon the prior works of both random strategies and biased based strategies by combining the advantages to form a new peer selection strategy. This research achieved the goal of creating a new peer selection strategy that further reduced the individual client's average download duration. The new peer selection strategy modified the random based peer selection strategy. These modifications included the use of advanced knowledge stored locally on the client to select peers, monitoring of the service capacity after a connection had been made,

logging this information for future use, and only replacing the worst performing peers. Reducing the average download time and producing more consistent download durations between download sessions enables the client and network to better plan for resource allocation and they reduce the demands P2P traffic has on the network.

A simulated P2P environment based on prior works was used to validate the new peer selection strategy since a stable static environment was needed and it was not feasible to set up a large P2P network in a controlled environment. After the simulation was created it was validated by the re-creation of prior works and the comparison of the results from the simulation environment to those published. This simulation environment simulated single client without competition and multiple clients with competition. The simulated server peer's service capacity was obtained from the same data used as advanced knowledge in the new peer selection strategy. In addition this simulation was modified to also simulate cross ISP throttling by randomly assigning a bandwidth cap to each ISP and when a new service capacity was requested the minimum value between the generated service capacity and the bandwidth cap was returned.

Four peer selection strategies were tested in the simulated environment: the smart peer replacement, the smart peer replacement with choke, the historic based, and the historic based with choke. These strategies where tested in both the single client without competition and the multiple clients with competition environments. The single client experiments tested the strategies against P2P networks consisting of eight different levels of heterogenic peers with cross ISP throttling both on and off. The results of the single client experiments show that the use of advanced knowledge in conjunction with monitoring the service capacity and only replacing the worst performing peers reduced

the average download duration and improved the consistency between the download durations. In addition this new historic based method also reduced the amount of cross ISP traffic when cross ISP throttling was simulated. Combining the choking algorithm with the historic based strategy did not significantly affect the observed performance.

The multiple clients with competition experiments tested the strategies against a P2P network with different levels of competition ranging from no competition to a six to one client to server ratio. Three different file sizes were used: 150 MB, 300 MB, and 3 GB. In addition these experiments were conducted with ISPs participating in cross ISP traffic throttling and no ISPs participating in throttling. The results of the multiple clients with competition experiments showed that the historic based strategy reduced both the average download duration and the normalized standard deviation between download sessions. These results also showed that the historic based strategy decreased the amount of cross ISP traffic when cross ISP throttling was simulated. The addition of the choking algorithm with the historic based strategy did not affect the observed results.

While this new approach further reduced the average download duration, improved the consistency between download sessions, and reduced the amount of cross ISP traffic for the individual client without the addition of polling traffic and the use of tracker peers there is still room for improvement. It is possible that future research will continue to improve upon the peer selection strategy by further reducing the time spent downloading from a poor performing peer. One possible avenue of improvement is by examining the information gathered outside of the P2P network for additional metrics that can be used to better predict the performance of potential server peers.

References

Adler, M., Kumar, R., Ross, K. W., Rubenstein, D., Suel, T., & Yao, D. D. (2005). Optimal peer selection for P2P downloading and streaming. *Proceedings of the IEEE INFOCOM,* 1538 - 1549.

Akella, A., Seshan, S., & Shaikh, A. (2003). An empirical evaluation of wide-area internet bottlenecks. *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement,* 101-114.

Bernstein, D. S., Feng, Z., Levine, B. N., & Zilberstein, S. (2003). Adaptive peer selection. *Peer-to-Peer Systems II,* 2735, 237-246.

Bindal, R., Cao, P., Chan, W., Medved, J., Suwala, G., Bates, T., & Zhang, A. (2006). Improving traffic locality in BitTorrent via biased neighbor selection. *Proceeding ICDCS '06 Proceedings on the 26th IEEE International Conference on Distributed Computing Systems,* 66-76.

BitTorrent (2016). Retrieved June 15, 2016, from www.bittorrent.com

Brienza, S., Cebeci, S. E., Masoumzadeh, S. S., Hlavacs, H., Özkasap, Ö., & Anastasi, G. (2016). A survey on energy efficiency in P2P systems: File distribution, content streaming, and epidemics. *ACM Computing Surveys (CSUR), 48*(3), 36.

Chandran, R. M., & Sajeev, G. P. (2015). Intelligent Pollution Controlling Mechanism for Peer to Peer Caches. *CIMSIM '15 Proceedings of the 2015 Seventh International Conference on Computational Intelligence, Modelling and Simulation,* 141-146.

Cheng, L., Hutchinson, N. C., & Ito, M. R. (2008). RealNet: A topology generator based on real internet topology. *22nd International Conference on Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008,* 526-532.

Chiu, Y.-M., & Eun, D. Y. (2008). Minimizing File Download Time in Stochastic Peer-to-Peer Networks. *IEEE/ACM Transactions on Networking,* 16(2).

Chiu, Y.-M., & Eun, D. Y. (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.

Chougule, A., & Deshmukh, S. (2011). Variable Chunk Based Parallel Switching To Minimizing File Download Time in P2P Network. *IJCSI International Journal of Computer Science Issues, 8*(4).

Dischinger, M., Mislove, A., Haeberlen, A., & Gummadi, K. P. (2008). Detecting bittorrent blocking. *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement,* 3-8.

Federal Communications Commission. (2017). Raw Data – Measuring Broadband America 2014. Retrieved January 1, 2017, from https://www.fcc.gov/general/raw-data-measuring-broadband-america-2014#block-menu-block-4

Fernando, T., & Keppetiyagama, C. (2013). ISP friendly peer selection in bittorrent. *2013 International Conference on Advances in ICT for Emerging Regions (ICTer),* 160-167.

Ferragut, A., & Paganini, F. (2016). Fluid models of population and download progress in P2P networks. *IEEE Transactions on Control of Network Systems, 3*(1), 34-45.

Fiorese, A., Simoes, P., & Boavida, F. (2013). Approach for service search and peer selection in P2P service overlays. *2013 International Conference on Information Networking (ICOIN),* 303-308.

Fuller, V., & Li, T. (2006). Classless inter-domain routing (CIDR): The Internet address assignment and aggregation plan. *IETF RFC 4632*.

Gummadi, K. P., Dunn, R. J., Saroiu, S., Gribble, S. D., Levy, H. M., & Zahorjan, J. (2003). Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. *SIGOPS Oper. Syst. Rev., 37(5)*, 314-329. Doi: 10.1145/1165389.945475

He, Q., Dong, Q., Zhao, B., Wang, Y., & Qiang, B. (2016). P2P Traffic Optimization based on Congestion Distance and DHT. *Journal of Internet Services and Information Security (JISIS), 6*(2), 53-69.

Hirave, T., Surve, S., & Malgaonkar, S. (2013). Selecting efficient peers in P2P networks for parallel task computing. *2013 International Conference on Advances in Technology and Engineering (ICATE),* 1-5.

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.

Ijaz, H., Saleem, S., & Welzl, M. (2013). Fewest common hops (FCH): an improved peer selection approach for P2P applications. *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing,* 449-453.

Jain, M., & Dovrolis, C. (2005). End-to-end estimation of the available bandwidth variation range. *SIGMETRICS Perform. Eval. Rev., 33*(1), 265-276. doi: 10.1145/1071690.1064242

Kaune, S., Pussep, K., Leng, C., Kovacevic, A., Tyson, G., & Steinmetz, R. (2009). Modelling the internet delay space based on geographical locations. *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing,* 301-310.

Lehrfeld, M. (2009). Peer selection Algorithm in Stochastic Content Delivery Networks to Reduce File Download Time. Doctor of Philosophy, Nova Southeastern University, Fort Lauderdale, FL.

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

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.

Li, K. (2014). On the expected file download time of the random time-based switching algorithm in P2P networks. *Peer-to-Peer Networking and Applications, 7*(2), 147-158.

Li, K. (2015). Analysis of file download time in peer-to-peer networks with stochastic and time-varying service capacities. *Future Generation Computer Systems, 42,* 36-43.

Liem, A. T., Hwang, I. S., Nikoukar, A., Yang, C. Z., Ab-Rahman, M. S., & Lu, C. H. (2016). P2P live-streaming application-aware architecture for QoS enhancement in the EPON. *IEEE Systems Journal, 99*, 1-11.

Liu, Y., Wang, H., Lin, Y., & Cheng, S. (2008). Modeling and Quantifying the Impact of P2P File Sharing Traffic on Traditional Internet Traffic. *Proceedings of the International Conference on Advanced Information Networking and Applications Workshops,* 1428-1433.

LZMA SDK (Software Development Kit) (2017). Retrieved April 19, 2017, from http://www.7-zip.org/sdk.html

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.

Mao, Z. M., Rexford, J., Wang, J., & Katz, R. H. (2003). Towards an accurate AS-level traceroute tool. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications,* 365-378.

Marchetto, G., Ciminiera, L., Manzillo, M. P., Risso, F., & Torrero, L. (2011). Locating Equivalent Servants over P2P Networks. *IEEE Transactions on Network and service Management, 8*(1)*,* 65-78.

MaxMind. (2017). GeoLite ISP Database. https://www.maxmind.com/en/geoip2-isp-database#features

National Broadband Map. (2017). Analize. Retrieved January 3, 2017, from http://www.broadbandmap.gov/analyze

OOKLA. (2017). Speedtest Intelligence from Ookla | Internet performance database. Retrieved January 6, 2017, from http://www.ookla.com/speedtest-intelligence

Ou, S., Yang, K., & Zhang, Q. (2006). An Efficient runtime offloading approach for pervasive services. *IEEE Wireless Communications and Networking Conference, 4*, 2229-2234.

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*, *24*(1), 437-448.

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.

Scandizzo, P. and Imperiali, A. (2014) Internet as a Growing and Dynamic Network: An Economic View. *Communications and Network, 6*, 69-75. doi: 10.4236/cn.2014.62009.

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

Steiner, M., & Varvello, M. (2011). Peer-to-peer traffic localization as a service. *Proceedings of the IEEE International Conference on Computer Communications (Demo), Shanghai, China*.

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). Traffic localization for DHT-based BitTorrent networks. *International Conference on Research in Networking,* 40-53.

Wilkins, R. (2013). Download Time Reduction Using Recent Performance-Biased Peer Replacement In Stochastic P2P Content Delivery Networks. Doctor of Philosophy, Nova Southeastern University, Fort Lauderdale, FL.

Wilkins, R., & Simco, G. (2013). Download Time Reduction Using Recent Performance-Biased Peer Replacement In Stochastic P2P Content Delivery Networks. *2013 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNet),* 86-91.

Xie, H., Yang, Y. R., Krishnamurthy, A., Liu, Y. G., & Silberschatz, A. (2008). P4P: Provider Portal for Applications. *Proceedings of the ACM SIGCOMM 2008 conference on Data Communication,* 351-362.

Yang, C., Zhou, Y., Chen, L., Fu, T. Z., & Chiu, D. M. (2015). Turbocharged video distribution via P2P. *IEEE Transactions on Circuits and Systems for Video Technology, 25*(2), 287-299.

Yang, X., & De Veciana, G. (2004). Service capacity of peer to peer networks. *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, 4*, 2242-2252. doi: 10.1109/INFCOM.200401354647

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

Zuo, X., & Iamnitchi, A. (2016). A Survey of Socially Aware Peer-to-Peer Systems. *ACM Computing Surveys (CSUR), 49*(1), 9.