ABSTRACT

SELVANAYAGAM, ALPHONSE HANSEL A. Swarm Over Swarm. (Under the direction of Dr. Injong Rhee).

43% to 70% of the Internet traffic is attributed to P2P file sharing. P2P protocols require clients to upload and download the same amount of data. However, traditional ISP billing models do not account for upload traffic. ISPs, therefore, lose a significant amount of their revenue for traffic outgoing from their egress routers with other ISPs. To curb cross-ISP traffic, ISPs even throttle cross-ISP P2P traffic resulting in lawsuits against them. Various application-level protocols for P2P have been proposed and implemented. But none of the existing approaches effectively reduce redundant data traffic coming into an ISP due to uncoordinated file piece requests among the peers in the same ISPs or ASes (called local peers). We propose a new technique, called Swarm Over Swarm (SOS) that implements efficient, scalable coordination among local peers. SOS allows local peers to make an informed decision regarding file piece requests so that different peers tend to download different pieces. This effectively reduces multiple downloads of the same file pieces by local peers. SOS is implemented on top of Vuze, a Java based BitTorrent (BT) client. Experiments were conducted on Planetlab as well as in real live BT swarms. The results show that on average, SOS reduces cross-ISP traffic by 17% on Planetlab and between 7%-48% in live swarms.

Swarm Over Swarm

by
Alphonse Hansel A. Selvanayagam

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2011

APPROVED BY:

Dr. Rudra Dutta	Dr. Khaled Harfoush

Dr. Injong Rhee
Chair of Advisory Committee

DEDICATION

To my Mom, Dad, Brother & Sister.

BIOGRAPHY

Alphonse Hansel A. Selvanayagam was born in Chennai, India. He graduated from Madras University, Chennai with a Bachelor's degree in Information Technology in May 2004. He worked for four years in the software field to gain industry knowledge. He joined North Carolina State University in Fall-2008 to pursue Masters in Computer Science. Alphonse plans to graduate in Spring 2011, upon completion of his research work.

ACKNOWLEDGMENTS

It is my pleasure to thank the many people who made this thesis possible.

First and foremost, I would like to thank Dr. Injong Rhee, my advisor for his insight, encouragement and patience in guiding me through this work. I also wish to thank Dr. Kyunghan Lee and Sankaraman Janakiraman with whom I have collaborated on this work. I would also like to thank Dr. Khaled Harfoush and Dr. Rudra Dutta for serving as members on my thesis defense committee.

I am grateful to all the members in Networking Research Lab(NRL) for their valuable inputs as well as support during my thesis work. I am especially indebted to my ex-roommates, current roommates for their emotional support, entertainment and the countless treats during this work.

Last and most importantly, I owe my thanks to my parents Dr.M.Selvanayagam & Hilda Selvam , my brother Adolph John Anthony and my sister Delphine Prescilla Anthony for their undying love, care, support and motivation. Without them, I would not have been able to reach this stage in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction

Peer-to-Peer(P2P) systems are the most popular means of sharing files, which are used by tens of millions of users to swap hundreds of millions of files daily, with BitTorrent(BT) being the most prevalent protocol among them. They are widely deployed and known to occupy 43% to 70% of Internet traffic depending on the regions of the world as of 2009 [37]. We tabulate the % of BT traffic(upstream and downstream) for each region in Table. 1.1 [38]. Even though the P2P traffic is not growing at the same rate as in the previous years, it does not show any sign of decline either. Due to the sheer volume of P2P traffic, use of random ports as well as encryption; it is difficult for the Internet Service Providers(ISP) to control it. It is often considered as a threat to other services and lowers the ROI (Return on Investment) for ISPs. The aim is to reduce the same over costly cross-ISP links of limited bandwidth (i.e., cross-ISP traffic) which is of top-priority.

Two diverse approaches have been developed for reducing P2P traffic. First approach involves an immediate solution from ISPs perspective: 1) Throttling Cross-ISP P2P traffic by deep packet inspection and sending fake TCP RST(reset) packets[6], 2) Caching contents[11] at the edge servers of the network. But both techniques do not fundamentally help in reducing the P2P traffic and also makes the ISP susceptible to lawsuits [41]. Second approach focuses on a fundamental solution which required modifications to the current BT systems. It involves a biased peer selection (e.g., Ono)[8] based on Content Distribution Network(CDN) redirections, latency measurements based on network coordinates (Vivaldi)[20] or iPlane[30]. However, existing approaches do not maximize the amount of data that is downloaded locally since they miss out on opportunities for cooperation between

Table 1.1: % of BitTorrent traffic during peak hours.

Region	Downstream during peak hours	Upstream during peak hours
USA	8.39%	34.31%
Europe	8.29%	29.97%
Latin America	6.8%	11.91%
Pacific Asia	16.91%	37.63%

local peers, by not modifying the piece selection algorithm.

1.1 Motivation

We define locality to be within the same Autonomous System(AS). An ISP could have multiple ASes, but not all ASes of the same ISP are connected directly to each other. The traffic could pass through intermediate ASes from different ISPs. Thus to reduce cross-ISP traffic the locality in our discussion is defined to be within an AS. Traditionally traffic within an AS is not restricted by bandwidth limitations, nor subjected to ISP throttling policies and oblivious to congestion faced by ISP egress routers.

In a conventional BT system, the standard peer selection algorithm utilizes the peers randomly selected by the *Tracker*. Since the peer list returned is ISP agnostic and the piece selection algorithm does not co-ordinate with other peers whom are within the same locality, there always exists redundancy in outgoing requests from inside the locality. We define it as; a peer requests a piece from outside locality ,when the same piece is available with another peer within the same locality. Thus the effects of biased neighbour selected are minimized as shown below due to their passive approach to this issue. None of the existing techniques account for the redundancy in the piece requests sent outside the locality which remains the major cause for the cross-ISP traffic.

Data Redundancy

For brevity, we refer to Ono as a BT client with Ono plugin. In a simple PlanetLab(PL) topology presented in Fig. 1.1(a), we measure the cross ISP traffic while using Ono and identify the room for further reduction by eliminating redundancy in traffic. We vary the number of peers located in L2 (USA West) region(locality) and log all incoming

bytes from outside the locality. We plot the results in Fig. 1.1(b). From the figure, it becomes clear that biased peer selection alone is not enough to reduce the redundancy in piece request. However, there is possibility that about 60% of redundancy in case of 4 peers can be eliminated by adopting collaboration among the local peers(active approach).

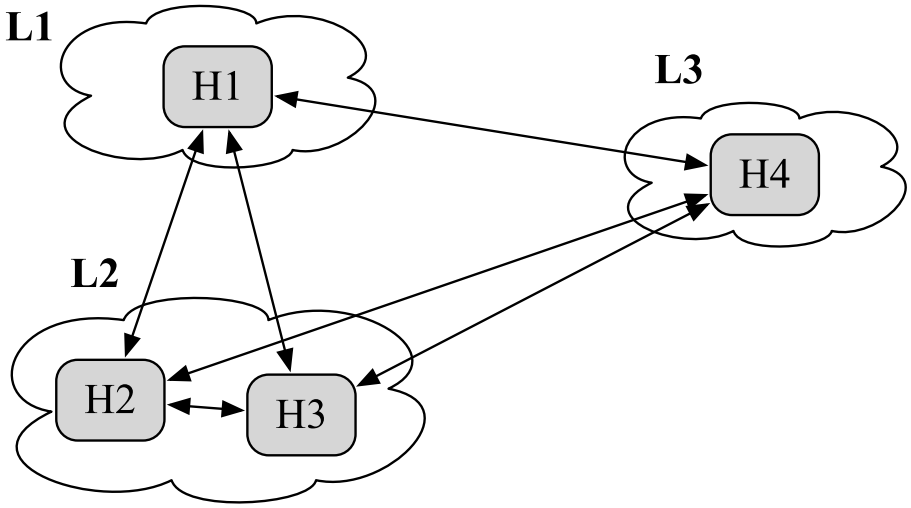
Live Swarm

Since all PL nodes are part of either the research network or the educational network which are a separate AS by themselves and it is highly improbable to find other peers within the same AS for a general torrent download. In order to identify the existence of local peers located in the AS, we run 5 different torrent downloads from a node within a residential ISP. All the torrents we use have more than 10,000 peers(including both seeds and leeches). We use the instrumented Vuze along with Ono plugin as in the previous execution in PL and join the swarms to share pieces. Based on the collected trace logs, we plot in Fig. 1.2 the distribution of piece requests and neighbours observed, with respect to the AS hop count. We can identify 6% of local peers from which only 3.63% of pieces have been requested. It is a strong evidence that locality is not fully exploited in downloading file pieces. There might exist three major reasons which could have resulted in large piece downloads from outside locality: 1) The peer did not identify these neighbours to be inside the locality 2) Even if identified,client did not find those peers to be INTERESTED [1], 3) Tit-For-Tat(TFT) [1] did not allow the peers to be UNCHOKED [1]. we further quantify the probability of finding local peers in Section. 6.

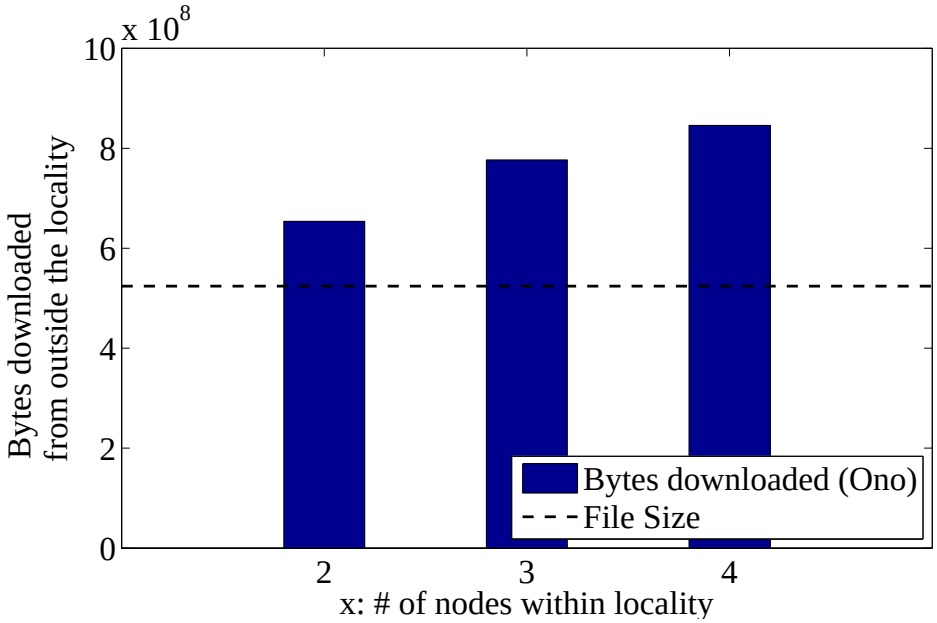
From these results we observe the existence of locality that can be exploited to reduce redundancy in traffic within a BT swarm, thus reducing cross-ISP traffic.

1.2 Contribution

Contracts are defined to split up workload between two peers interested in a common file such that, they only request mutually exclusive content from other peers collectively. The benefit of contract, is to allow collaborating peers to avoid redundancy in piece requests within the identified locality, which also allows distribution of workload. Thus to improve download completion time and compete with the existing local-rarest first algorithm, we propose a modified locality aware piece selection algorithm for contracted peers and a new

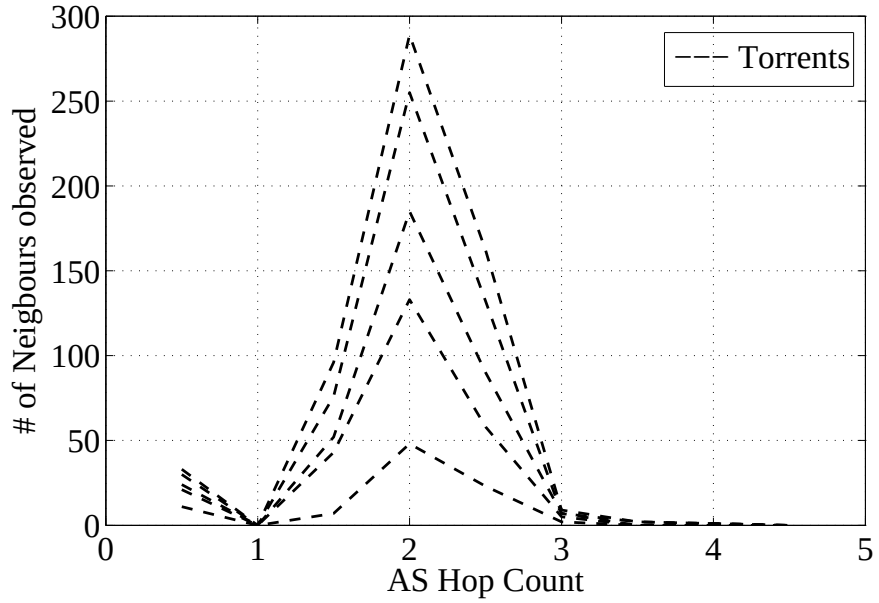


(a) Setup of PlanetLab nodes for motivation

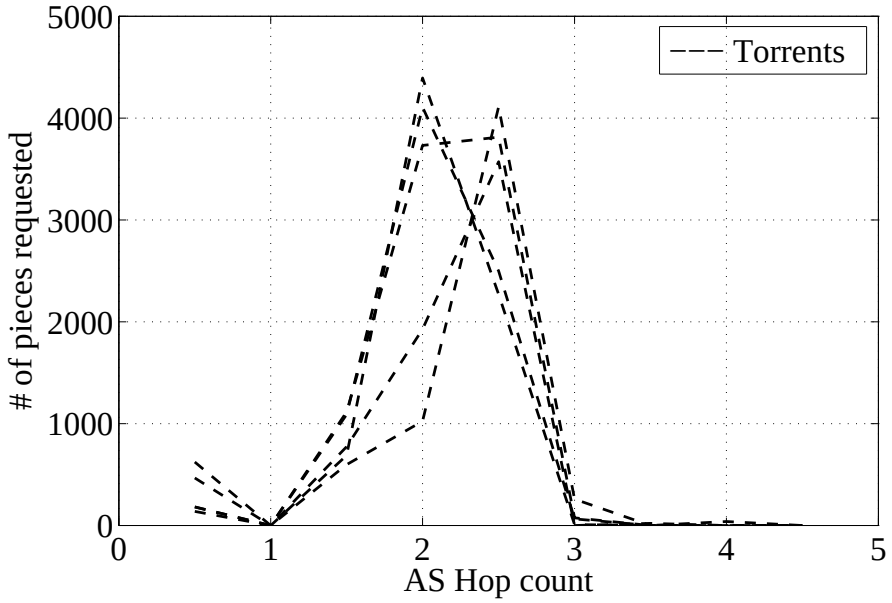


(b) Bytes downloaded from outside locality

Figure 1.1: PlanetLab setup and Bytes downloaded from outside locality



(a) Neighbours AS hop count distribution



(b) Piece requests AS hop count distribution

Figure 1.2: AS Hop Count distribution of the piece requests and neighbours from a residential node involved in a swarm.

protocol for setting up contracts called Swarm Over Swarm(SOS). Once the locality is identified, we split up pieces requested outside locality between local peers to avoid duplicates. Initial contract set-up between the local peers ,allows them to make a informed decision regarding the non-local piece requests which is pivotal to reduce cross-ISP traffic.

SOS consists of three major operations: 1) Local peer detection 2) Distributed job split-up 3) Local sharing. First, the detection algorithm identifies local peers by uniquely combining 4 different information from Ono, iPlane database[7], traceroute and Team Cymru's IP-to-AS mapping database[34] where *local peers* are narrowly defined to be the P2P nodes located within the same AS (i.e., in the locality) of an ISP. Second, the job split algorithm explicitly splits the downloads of file pieces among local peers exclusively. The split-up is the complete responsibility of the two nodes which are locally connected and they make their own contracts for the assigned duties. Thus, it is inherently distributed. Finally, the file pieces are locally shared to complete the download. we anticipate and leverage the opportunities for collaboration with nodes in the same locality for future piece requests thus maximizing its potential.

The goal of our work is to design a distributed, active and cooperative BT system that considers the locality of the peers; to minimize cross-ISP traffic by maximizing the amount of data that they download from peers within the same locality. Our proposed system utilizes Ono plugin along other filtering approaches to identify local peers as well as optimize locality aware piece downloads.

SOS is implemented in a popular BitTorrent client, Vuze and the performance of SOS is compared with that of Ono which is provided as a plug-in for Vuze. We conducted experiments distributing a large file to 100 PlanetLab nodes deployed in the world as well as downloading an Ubuntu image file from a real bit torrent network using multiple local laptops.

Main contributions of this thesis are summarized as follows.

- **Precise Local Peer Detection:** Detecting local peers by relying on the CDN database alone like Ono does not cover 100% of IP addresses because the locality is defined by the CDN network and that could cover multiple ASes. Infrastructure independent detection method suggested by TopBt generates non-ignorable overheads and is not reliable like the CDN database. Thus, we provide a technique determining locality by combining multiple information sources. Our technique guarantees nearly

100% coverage of all the IP addresses based on the empirical results from PL.

- **Distributed Download Job Split:** Without any central coordinator, we implemented a distributed job split-up algorithms relying on the individual contracts between any two peer pairs. The contract guarantees that the contracted pieces will be downloaded within the locality thus, peers can only focus on non-contracted pieces assigned to themselves. We proposed two contract strategies, *Altruistic* and *Fair* which can be adaptively used according to the altruism of peers. Peers request pieces based on a slight variation of the existing local rarest first algorithm [1] and the pieces are divided into two sets, contracted and non-contracted pieces to be downloaded from local and non-local peers respectively

- **Leave Prevention:** Two local peers involved in a contract always complete downloading together because of the tit-for-tat algorithm. Likewise, multiple local peers involved in multiple contracts among them complete downloading in a similar fashion. Thus, the situation that a peer leaves the system as soon as its download completes is automatically prevented in one of the variants of SOS. This prevents free-riding and ensures the availability of pieces until the completion of a download, thus influences the download positively.

We also analyse trace logs collected from over 10 popular torrents and present the observation to support our assumptions. The observations includes the 1) Availability of peers within the same AS for collaboration in temporal region and otherwise, 2) The effectiveness of CDN Relative network Positioning(CRP) technique to filter local peers within each swarm and 3) Relationship between AS hop count, latency and throughput.

1.3 Organization

The rest of the thesis is organized as follows. In Section. 2, we provide the background information on the BitTorrent(BT) protocol as well as discuss the relevance and the difference of related work to SOS in Section. 3. In Section. 4 we discuss about Ono and its limitations, which aim to motivate our filtering approaches for finding local nodes. In Section. 5, we illustrate the detailed algorithms and operations of SOS implemented in a BT client. In Section.6, we present the probability distribution of locality in temporal

domain by analysing trace logs collected from a live swarm and the experimental results using the implemented BT client are provided in Section. 7. In Section. 8 we summarize out work and provide details regarding future work.

Chapter 2

Background

In this chapter, we discuss about the architecture of the BitTorrent(BT) file sharing system along with its limitations. We also explain upon the reasons and implications about the high cross Internet Service Provider(ISP) traffic due to BT compared to other transfer mechanisms such as ftp and http.

2.1 BitTorrent

BitTorrent clients are implemented based on the BitTorrent(BT) protocol as proposed by Bram Cohen [1] and there are various client implementations of the protocol available as of today. Bittorrent [24], Transmission [26] and Vuze(Azureus) [25] are few the prevalent clients among the many available. This protocol harnesses the upload bandwidth available from every peer (within the swarm) involved in downloading a particular file. A user interested in sharing a file breaks it into multiple pieces (helps in paralleling the upload or download) and creates a torrent file with extension *.torrent*, which contains the information regarding the tracker's IP address (public server), file size, piece size, HASH values for each piece and comments. The torrent file is shared among the users interested in downloading this content file either via a public website or through IRC, E-mail to name a few. A tracker is a public server that the clients would contact on regular interval to get information regarding other peers involved in the download which form the BT swarm. The tracker is the only centralized part of the entire BT operation and it does not act as a bottle neck since there are no computation intensive processing required, other than to

respond back to the client with a list of IP addresses stored on a list specific to each torrent file (Swarm).

Due to this simple design of the Tracker, they always respond back to a new peer with a randomly selected list of peers from the same swarm without differentiating between alive, stale, local peers and otherwise; to avoid load on the tracker itself. After the client obtains a subset of the list from the tracker, it set-up connections and sends download requests to each of these peers based on the piece availability. Sharing of the pieces among the peers is done on a Tit-for-Tat [22] basis. Furthermore the client contacts the tracker on regular intervals (every 30 minutes) to updates its peer list. This enables the client to find new peers to download from, thus improving availability of pieces.

2.2 Cross ISP Traffic

To ensure fairnees in BT, a peer approximately downloads and uploads the same quantity of data (Tit-for-Tat) for each file and this happens with all the peers involved in the swarm. This relationship between peers is formed without considering the geographical location, underlying network or the ISP link costs. Normally ISPs rely on the traditional billing model that Clients will use HTTP traffic, which includes a small amount of upload traffic compared to the large amount in download traffic; which is different to the P2P model.In HTTP traffic model, the cost is offloaded to the servers hosting the HTTP content.In P2P model, the ISPs are forced to pay more on their egress routers with other ISPs to accommodate this traffic without getting their ROI from the end user running a P2P software.

Due to this, the ISP resort to traffic limitation to curb the load on their links during peak time. The traffic limitation techniques is detrimental not only to the BT clients ,as well as to the ISP due to loss of customers. To overcome this issue ,various methods such as Pyxida (measuring latency) ,Content Distribution Network(CDN) based redirection have been proposed to select peers based on certain beneficial criteria as opposed to using the random set returned by the tracker. The motive is to find peers closer to the client to help in reducing the cross-ISP traffic, as well as to reduce download time.

In the next Section. 3 we discuss about the related work and their limitations. Among these the most effective is the CDN Relative network Positioning(CRP) technique

that uses CDN based redirection to find peers within a swarm that are close to the BT client. CDN based redirection(Ono) also is discussed later.

Chapter 3

Related Work

BitTorrent(BT)[1] protocol requests pieces from a subset of peers randomly provided by a tracker, which consistently collects a complete set of peers interested in the same torrent file. BT exchanges file pieces with connected peers based on tit-for-tat to avoid free riders by adaptively choking or unchoking[2] them. The tit-for-tat algorithm coupled with random peer selection is known to become ISP-hostile as it generates high volume of cross-ISP traffic across multiple ASes.

Karagiannis et al.[3] presented a seminal work which studied the impact of peer assisted downloads on the ISPs backbone infrastructure by analyzing the packet trace logs collected in the Internet, as well as the tracker logs. Locality-unaware piece requests doubles the traffic at the ISP as well as their peak outbound traffic, since each peer now acts as a data source thus, affecting the revenue stream of the ISPs. They pointed out the potential for reducing cross-ISP traffic by making locality-aware decisions and caching data at the ISP egress points.

Implementations that include "Pyxida", an implementation of the *Vivaldi* algorithm, Internet measuring or mapping tools such as "iPlane" provide information regarding the latency, loss rate, hop count to make informed decisions regarding the peer selection. Pyxida calculates network co-ordinates for the peers using the spring relaxation, whose values are used to calculate the latency values which could be used to short list peers. Iplane is an Internet mapping tool that uses vantage servers to measure and map the Internet to generate an Atlas. Latency and loss rate could be used to short list peers but are plagued by issues similar to Pyxida. Techniques involving latency are proved to be incorrect by

Vishnumurthy et al.[13]. They have shown in their work, that latency as a property cannot be used to find local peers in a BT swarm.

The related work can be classified into two parts 1) Infrastructure Dependent and 2) Infrastucture Independent. Centralized approach, Caching and CRP technique come under infrastructure dependent model whereas TopBT comes under infrastructure independent approach.

There is a series of work that involves a centralized approach that are susceptible to a single point of failure, induced with performance impact due to the sheer volume of queries or requires extensive infrastructure to handle the query loads. Bindal et al.[15] have proposed a approach that includes modification to the tracker which adds complexity to the tracker software as well as modifications to the client software for identifying peers within the ISP. This essentially requires cooperation from the ISP. Aggarwal et al.[16] suggested the use of an Oracle services, that is used by the P2P client to ranks its neighbours (i.e., connected peers) based on AS locality, hop count and distance to the edge of the AS based on IGP[21] metric. Xie et al.[17] propose a node called *iTracker* embedded with an ISP that helps the client and the tracker to make an informed decision regarding the peer selection process.

Some papers take a different approach of setting up caches at the ISP egress points to avoid redundant traffic entering the ISPs. This method is discouraged by the size of the infrastructure required to maintain P2P content cache, since it has been seen that the size of the shared files has increased to the size of Terabytes. Furthermore it allows the ISPs to be susceptible to lawsuits, since that cache would infringe on various copyrights. Shen et al.[11] propose HPTP, a technique that tricks the existing HTTP caches at the ISP egress points to cache P2P contents either but it also does not overcome the known problems.

Ono[8] proposed by Choffnes et al. for a BT client, Vuze is designed to reduce cross-ISP traffic in swarms. Ono attempts to select nearby peers among a random set provided by the tracker by preferring a CRP(CDN based Relative network Positioning) ratio closer to one. The heuristic used by Ono is CDN redirections represented as CRP[9]. CDNs such as Akamai[36] perform extensive measurements of the Internet on a regular basis and measure path properties between various ASes. This information is recycled by Ono using CRP. The extent of the locality of two peers is identified by analysing the number of CDN requests from each peer redirected to the same CDN edge server. Ono reduces cross-

ISP traffic by trying to download more contents from nodes within the locality compared to the nodes outside the locality. We show that the gain that Ono pursues can be improved further with our algorithm.

Lin et al.[4] propose a technique to check local peers before requesting content from outside locality by using the local peers availability list. This technique helps in reducing cross-ISP traffic but does not avoid redundant requests due to BT's pipe-lining [1] technique which helps to avoid the latency between multiple requests. This involves breaking a piece into multiple blocks , but the granularity of availability list is in pieces. Also each peer follows the rarest first policy (RF) ,so each peer requests similar pieces due to non-availability which will result in redundant requests outside locality especially in case of Flash crowd scenario.

Ren et al.[14] have proposed a infrastructure independent technique, TopBT using traceroute and custom ping to determine the AS(Autonomous System) hops as link hops between peers to find "nearby" peers. They show reduction in AS hop count with respect to the neighbours selected compared to the conventional BT, but the improvement over Ono is unknown. Our locality detection algorithm hierarchically combines both approaches. We first utilizes the pre-collected information by the CDN to avoid the overhead incurred by traceroute or ping for Internet path already listed in the database.

In the aspect of collaboration among peers, 2Fast (Tribler) by Garbacki et al.[12] introduce the idea of collaborative download to increase the download speed such that each peer requests only unique pieces. In their work, the bartering peers (helpers) are selected based on the social connections that pre-exist between the users. This scheme is based on the incentives available for use with a social circle, which is based on their previous work. As they are not interested the reduction of cross-ISP traffic, the helpers are chosen irrespective of the locality, since the social relations they consider can even span multiple ISPs. Manual intervention is required by the user to control the level of confidence between the peers to allow the bandwidth to be shared.

Most of the previous work,discuss about finding local peers and leverage their bandwidth potential for faster download, which indirectly reduces the cross-ISP traffic. But there exists no active coordination among these peers ,as to which piece each peer identified should download in cohesion with locality. Our algorithm is first of its kind that combines the approach of selecting locality biased neighbours on the collaboration

mechanism between the peers to reduce cross-ISP traffic as well as improve on download time.

Chapter 4

Ono

In this Section we discuss about the Ono plugin, it use of the CRP technique and its limitations.

4.1 Introduction

Ono is an open source software plugin developed for Vuze(Azureus) BT client. This Plugin helps in calculating the CDN redirection ratios for all the peers returned by the Tracker for a particular torrent file. It uses the CDN based relative positioning technique (CRP) which internally uses the Cosine Similarity formula to calculate the final value; to identify if a peer in the swarm is either local to the corresponding peer that executes this calculation or otherwise. The Vuze plugin's Application Programming Interface(API) allows Ono to remove peers identified to be outside the locality. Thus enabling Ono to maintain a fair ratio of local and non-local peers within the neighbour list without affecting the availability of the pieces for download in the swarm. This section discusses about the CDN, CRP and its implementation in Vuze.

The major factors related to Internet that affect data transfer are latency, throughput and bandwidth. Bandwidth is a property that is under the control of the client. But throughput and latency to the content source can be controlled by the content providers by placing the content servers closer to the client. This directly affects the download rate for the client. Throughput and latency are affected by congestion, total number of hops between the client and the server hosting the content. If the content provider is able to host

the content at a server location closer to the client, it helps in reducing latency as well as finding a route with minimum hop count that avoids congested links which would improve throughput also. And this is exactly, how content distribution networks aim to achieve high throughput by replicating content across various edges server that are closer to the clients and redirecting content request to the appropriate server for each client by DNS redirection or URL rewriting in case of HTTP content. The CDN providers have developed various proprietary techniques to measure the latency and congestion status of the various route hops to find an edge server closest to the client that avoids all bad routes.

4.2 Biased Neighbour Selection

Each BT client on start up, would initiate a Domain Name Service(DNS) [39] request of *Type A* to a list of CDN Domain Name Servers(DNS) which is available to the client as part of its plugin bootstrap procedure. The providers of the CDN have their proprietary technique to measure latency and other parameters required to identify a edge server, a server possibly located in the client's ISP, or at the least a server capable of providing the content requested by the client with high throughput. Once the request is received by the top level CDN DNS, based on the already calculated measurement of the Internet, the DNS server finds out, among their edge server's which one is a most optimal match to provide content to the requesting client.

This information with the list of edge servers are reported back to the client as part of the DNS *answer* section. Each client receives its set of edge servers and keeps refreshing them in regular intervals to include the dynamic nature of the Internet. To calculate the Cosine similarity value, the client needs the edge server mapping of other neighbouring clients also to classify them based on locality. This is done by exchanging the ONO_RATIO messages between the clients that support the Ono plugin. In case of clients which do not support Ono plugin, the client uses the recursive DNS requests [39] as explained below. But most of the DNS do not support recursion due to security issues.

1. Issues a *Type NS* DNS request to obtain the authoritative server name and IP address.

2. Issues a recursive DNS *Type A* request to the top level CDN DNS Server which returns with the edge server IP list. If recursion is not supported, the *ANSWER* section return

null.

Cosine similarity formula is a mathematical measure to identify if two vectors are similar. When they are identical the output value is 1 and when there are orthogonal, it computes to 0. Consider the following example from the Fig. 4.1. C1, a BT client sends a DNS request to the CDN network whose IP address is already known. Any one of the CDN DNS servers S[1-3] respond back with an edge server IP address. In this case for C1, R1 is the closest edge server. Similarly C2 requests the CDN for the edge server IP. If C2 also had got R1 then C1 and C2 are within the same locality. If C2 had obtained any other edge server IP, either R2 or R3 then the cosine similarity value would have been zero. This is defined as the CRP technique. Using this redirection ratios Ono is able to identify local and non-local peers for biased neighbour selection.

Each client computes the *cosine similarity* value for each peer pair using the calculated edge server ratio of itself and the ratio provided by the other respective neighbour. This values ranges between [0,1] with value closer to 1, meaning that the peer pair is local. A threshold value is used to separate non-local and local peers. In Ono, once the peer's locality is obtained by the plugin, the peers is either dropped are allowed to connect for non-local and local peers respectively thus helping in reducing cross ISP traffic. We show in Section. 1.1 that just dropping non-local peers does not reduce cross ISP traffic.

4.3 Plugin

For the Bittorrent Client Vuze, the CRP implementation is provided as a plugin called Ono. There are three main modules to the Ono plugin that ensures a biased selection of peers for the neighbour set provided by the Tracker. The Vuze Plugin API provides the following API to facilitate Ono

- It get's intimated, whenever a new peer gets added to the neighbour list by Vuze.
- It has the ability to drop peers, if required via the plugin API.

The three main modules are defined as

1. The *Digger* daemon thread instance, is responsible for contacting a central server for obtaining the list of Akamai CDN servers. This list is maintained by Aqualab on their

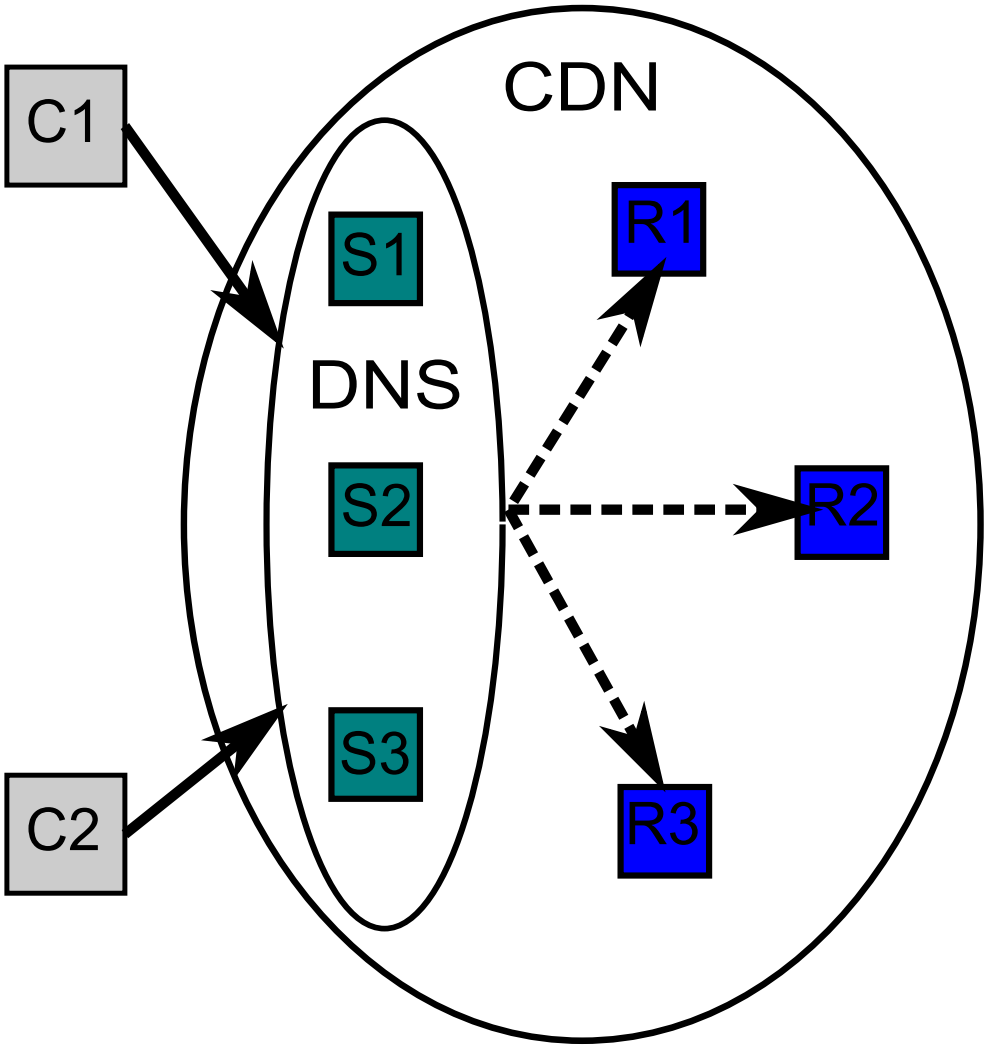


Figure 4.1: C[1-2] represents the BitTorrent client, S[1-3] represent the top level DNS servers that respond back with closest edge server IP addresses and R[1-3] represent the edge server hosting content.

website [40]. Since the list is obtained while bootstraping Ono, even if the CDN server names are modified, the latest list is obtained by issuing a http GET request. Based on this information, *Digger* instance calculates the edge server mapping by issuing DNS requests of *Type A* for this client and updates its database on regular interval to accommodate the dynamic nature of the Internet.

2. The *CDNClusterFinder* calculates the cosine similarity value for every peer available in its database based on the stored edge server mapping and the mapping obtained for the other peer respectively. Once the value is updated, an event is created, which either allows the peer to be part of the neighbour list or dropped while still maintaining the ratio between local and non-local peers to avoid non-availability of pieces.

3. The *OnoPeerManager* is the database that stores the cosine similarity values as well as the edge server mapping along with other peer information, which ever has come in contact with the vuze client.

4.3.1 Exchange of Ratios

After the Ono plugin calculates the edge server ratios, it has to obtain the ratio from other clients to compute the locality. To allow this, a new Vuze message type *ONO_RATIO_MESSAGE* is registered and used to transfer the edge server mappings to all clients that support Ono plugin.

4.4 Modification to Ono

As part of our implementation to support collaboration among peers, we have modified Ono plugin as follows

- The edge server mappings are sent for every 30 second period to all peers connected to the Vuze client, to ensure that *SOS* has the correct updated information to setup its contracts. Initially the edge server mappings are sent only once during bootstrap.

- Added new methods to the existing plugin, so that the internal data structures of Ono, especially the *OnoPeerManager* are accessible by the Vuze ,to help in filtering

out local peers from the complete neighbour set. Refer to Appendix A for the code changes.

Chapter 5

Swarm Over Swarm

In this chapter, we introduce the SOS algorithm and its variants that enable coordination among local peers in a swarm to help reduce redundancy in downloads. This helps to achieve reduction in cross-ISP traffic. The new algorithm runs in parallel with BT's TFT algorithm for non-local peers thus maximizing the use of the available bandwidth in the swarm.

5.1 Introduction

SOS algorithm consists of three major phases 1) Precise local peer detection, 2) Establish Contracts and 3) Local piece sharing. In the first phase we filter out peers with whom we can establish our contracts. In the second phase, for each identified peer to be local; we setup contracts that establishes the workload that need to be shared to complete the download. Hence contracts can be defined as split up of incomplete pieces common for both peers to be downloaded from outside locality, in addition to exchange already downloaded pieces. After the contracts are established the pieces yet to be downloaded are categorized as 1) Non-contracted piece(NCP): Piece that needs to be downloaded from outside locality. 2) Contracted piece(CP): Piece that needs to be downloaded from any of the peer with whom contracts have been established(inside the locality). The NCP is the responsibility of the concerned peer to download the piece within the locality and NCP requests of all the peers within the contract would be mutually exclusive thus avoiding redundancy. Peers proceed with NCP to be downloaded from outside locality using the Local

Rarest First(LRF) [1] policy of BT. The third and final phase of SOS involves swapping of CP within locality using LRF amongst them.

Each of the various phases of the algorithm are discussed in detail below. For sake of brevity we refer Native BT with Ono plugin as Ono, SOS(Altruistic) as SOS-A and SOS(Fair) as SOS-F respectively.

5.1.1 Precise local peer detection

Before describing our algorithm, we provide a summary of the conventional algorithm. In that, once a peer parses the torrent file and obtains the IP of the tracker, it contacts the tracker at regular intervals(every 30 minutes) to obtain the peer list. In general, to avoid performance impact on themselves, trackers always return a random set of unprocessed IP addresses back to the requesting peer which is ISP agnostic. Existing techniques such as CRP, network co-ordinates have been used as filters to avoid high latency peers while requesting pieces. Among the existing techniques, CDN-based Relative network Positioning(CRP) is most effective since they recycle the CDN view of the Internet provided by Akamai and other CDNs for identifying low latency peers. CRP relies on the property that if two peers get redirected towards the same content edge server then, they are in proximity to each other. Depending on network traffic in the Internet, the peers could be redirected to multiple edge servers over a period of time. To normalize this result, they use the cosine similarity formula [8] which provides a value ranging from 0 to 1. If the ratio is closer to 1 then, the peers are in close proximity to each other.But this proximity is based on location with respect to the edge server. Also low latency does not necessarily imply proximity.

However, in locations which do not have high penetration of the CDN edge servers, the granularity of the locality measured by the aforementioned technique is insufficient. For an example, a peer in an ISP A would be paired up with a peer in an ISP B, since there might be only one edge server catering to the content needs based on the CDN company's requirement. Implementing collaboration by pairing these two nodes in two different ISPs whose path might have more than one AS hop, would be meaningless in achieving cross-ISP traffic reduction. To avoid that issue, we propose a filtering technique that uses the information from *iPlane* database as well as traceroute and *Team cymru* database. Based on the following procedures and criteria, we determine the locality in which peers to collaborate

with should exist together.

We use the value 0.98 in Step I, which has been empirically observed to be effective in filtering non local nodes within PL. We don't use the value one, since there could be nodes in the edges of an AS that could be redirected temporarily to a different CDN edge server within the same or different AS due to transient network conditions. Ono uses a lower cosine threshold value of 0.15 to differentiate between non-local and local peers which works out to effective due to normal BT's peer selection algorithm to filter out low throughput nodes. Based on our observation within PL, we have found that with a value less than 0.97 results in nodes with more than one AS hop between them. This especially occurs in areas with low CDN edge server penetration. Since our nodes set include a wide selection of the geographical area, we have to limit ourselves to a larger value of the cosine similarity close to one, to reduce the size of our filtered local peer set. So at the end of this phase we have identified local peer with whom we can establish contracts, that would help reduce redundancy within the locality.

5.1.2 Distributed download job split

Once local peers are identified, we let the peers to collaborate with each other. This involves setting up a separate TCP connection with the local peer and exchanging information regarding the collaboration contract. The information exchanged essentially includes 1) pieces already contracted with other peers, 2) pieces yet to be downloaded and 3) the pieces already downloaded. Relying on this information, peers establish a contract mainly for the pieces yet to be downloaded in both nodes. For the already downloaded pieces, we apply two different strategies, *Altruistic* and *Fair* which will explained later. In the following control flow shown in Fig.5.1, we show the detailed packet exchanges implemented in SOS. The detailed procedures are described in Algorithm 5.2. For convenience, we assume that *peer A* is the initiator of the contract, who identified *peer B* as a local peer:

As our algorithm is totally distributed and do not aim at achieving even split to all the collaborating peers, it is possible for one peer to complete its shared workload faster due to its superior capacity, uneven split by multiple contracts or uneven split by already downloaded pieces. This occurs usually when there are more than three peers involved in a contract as shown in Fig. 5.2. We intentionally let them happen to keep our protocol as simple as possible so as to minimize the overhead and make it scalable. However, it is

Algorithm 5.1: SOS : Locality detection

(I) CRP : The cosine similarity ratio returned by the Ono plugin using the CRP technique must be greater than 0.98. We reuse the value already calculated by the Ono plugin as part of its procedure to identify non-local peers and reduce transfers between them. This helps us to exclude a large number of peers from the candidate set.

(II) iPlane : Peers selected by (I), would then be used as the destination IP address in an xml query to the iPlane database server for the AS hop count information. If the AS hop count is 1, it means that both the peers are within the same AS and they are selected to proceed forward.

(III) Team cymru : If iPlane is not able to predict or provide the AS path information, we convert the IP to the corresponding AS number based on Team cymru database and verify if there are peers in the same locality based on the AS number.

(IV) Traceroute : This step is optional and is only executed in case only if (II) and (III) fail. We execute traceroute between the node pairs and obtain the router IPs which are mapped to their AS numbers using the RADb or Team Cymru as required. We use simple techniques to resolve issues related to non-responsive routers. But, (II) and (III) are capable of providing the AS hop information for 100% of the IP addresses in the world based on PL experiments and live swarm analysis thus, (IV) is rarely used.

Algorithm 5.2: SOS : Job split-up

(I) Peer A initiates a TCP connection to Peer B and executes an handshake, followed by request for contracted pieces of Peer B.

(II) Peer A identifies the unfinished (i.e., not yet downloaded nor contracted) pieces in both A and B and splits them up evenly. This represents the amount of work (i.e., duty) to be done.

(III) In addition to (II), Peer A or B can offer additional pieces for free, which has been already downloaded only if both of the peers are in *Altruistic* mode, else there is *Fair* mode in which no additional pieces are given as part of the contract.

(IV) Peer A sends back the contract as well as the offered pieces to Peer B. If Peer B satisfies with the contract, it sends ACCEPT back to Peer A.

(V) By sending HANDSHAKE, the contract is established. A typical locality-based piece selection algorithm is used to exchange pieces.

(VI) If a peer is in **altruistic** mode and completes its duty before other peers in the contract complete, the completed peer reconnects to other peers in its contract to renegotiate (i.e., split and receive) their remaining work again thus, the completed peer contributes to boost the download time or others.

The step (VI) continues until the download is completed for all contracted peers within the locality.

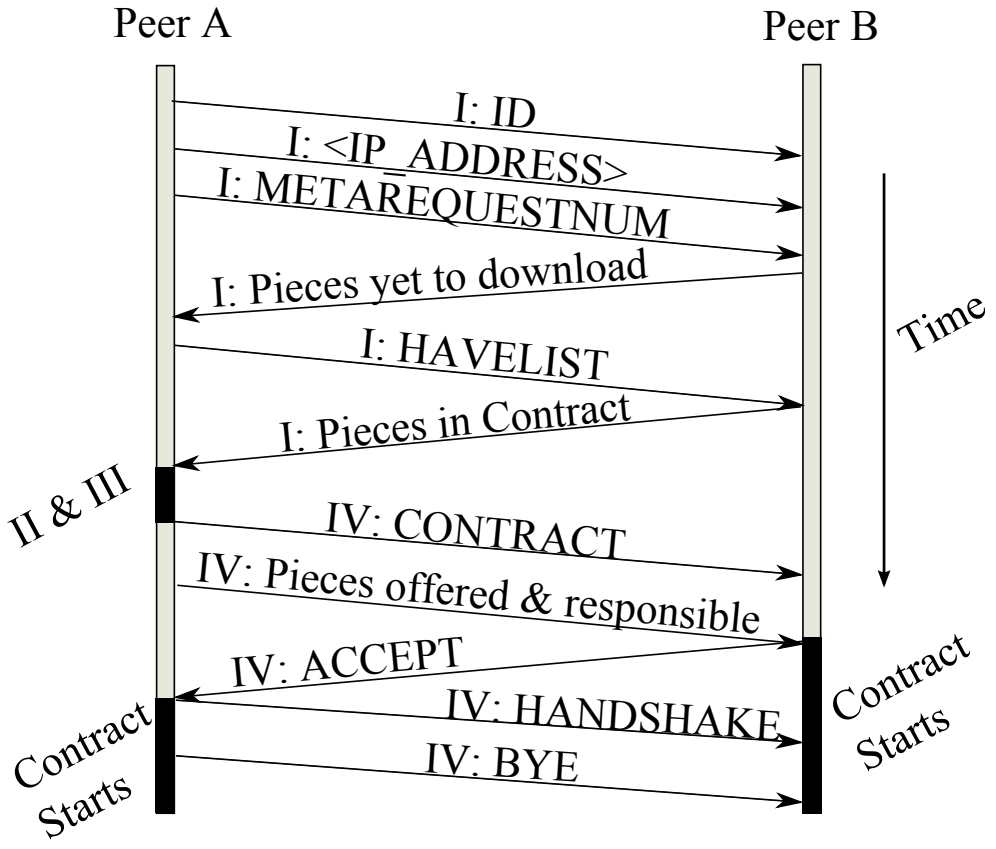


Figure 5.1: Control flow between peer A and B to establish a contract.

obvious that the completion time distribution of SOS is much more even than native BT or Ono because of the contract. This bring about the significant benefit under a circumstance that peers are not selfish enough to shut down their BT client as soon as they complete.

We do not aim to split the workload evenly by the number of collaborating peers, We always ensure that any given single instance of the contract is always established between two peers only. Since at the time of establishment, we do not know if there would be any other local peer that we would connect with, in the future. This uneven split of work is taken care of by the *Renegotiation* protocol described in (VI). to apply tit-for-tat as well as compensating for the difference in performance and bandwidth of the peers.

We illustrate more detailed job split procedures for both Altruistic and Fair algorithms. Their procedures are the same when there is no already downloaded pieces which is shown in Fig. 5.2 (a). In a case that some pieces are already downloaded, the procedures are

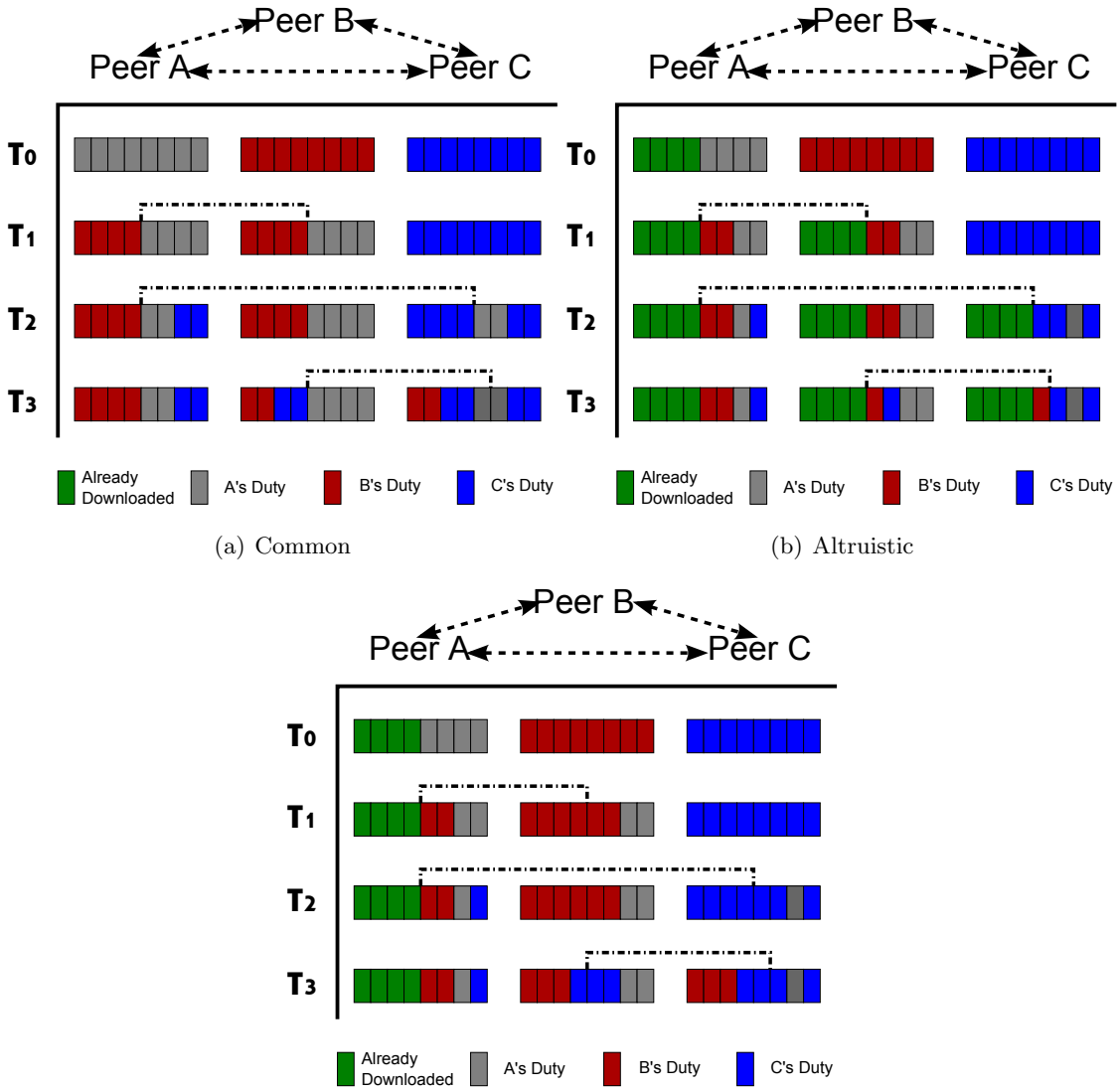
differentiated as shown in Fig. 5.2 (b) for Altruistic and Fig. 5.2 (c) for Fair. In Fig. 5.2(a) At $t0$, each peer is supposed to download the whole pieces by itself. At $t1$ and $t2$, Peer A, B and C establish contracts consecutively thus, the duties for downloading specified portions are distributed to Peer A, B and C. (Dotted line represents instantaneous connections among nodes.) b) If there is already downloaded pieces in Peer A at $t0$, while split goes on over Peer A,B and C, Peer A altruistically shares the pieces, thus overall workload to Peer B and C are reduced. Completion time is obviously faster than Fair. (c) In this algorithm, Peer A does not share the already downloaded pieces to other peers.

There are two key difference between these two variants.

1. SOS-F follows the TFT rule strictly and does not upload additional peers to the contracted peers other than the distributed workload. Where as SOS-A uploads more data than it downloads if it has pieces required by the other peer. As shown in Fig. 5.2(b), we see that Peer A gives pieces [1-4] to peer B in addition to piece [5-6] as part of the distributed workload. But in case of SOS-F Fig. 5.2(c) peer A only gives pieces[5-6] as part of the contract and not the already available pieces, thus not showing altruism within the locality.

2. In SOS-A, we have the *Renegotiation* protocol that occurs, after any of the peer within the contract is able to complete its workload faster compared to others. It has been elaborated on STEP VI in Algorithm. 5.2.

To avoid deadlocks in piece requests, the peer maintain a marked distinction between pieces that are downloaded and pieces that are contracted to be downloaded by a neighbour. A peer can contribute pieces as part of its contract only, if the peer already has downloaded the piece. Assume that peer A and B have already established a contract and piece 8 will be downloaded by B into the locality. when peer C establishes the contract with A, A cannot provide piece 8 to C as part of its SOS (altruistic) contract since A is yet to get it from B. If it was shared to C as part of the contract then if C forms a contract with B, it is possible that B might offload that piece download to C thus in eventuality no peer within the contract download the piece into the locality. Thus as part of the contract ,we never shared pieces that are contracted to others ,ever if we know that we will get that piece in the future from a another peer.



(a) Common

(b) Altruistic

(c) Fair

Figure 5.2: Distribution of workload among peers within a locality

5.1.3 Local sharing

The peers involved in contracts are not subject to the existing BitTorrent's TFT, CHOKING and SNUBBING policies since the fairness of the piece exchange is already considered in the contract. The existing LRF algorithm requests pieces by giving high priority to rarer pieces which are calculated by computing the availability of the pieces over multiple neighbours. For contracted pieces, the LRF is not applied either since the source of the piece has already been identified and it can be downloaded only after the source completes the download for that piece.

The existing technique would maintain a rarest value that is dependent upon the set of pieces, yet to be downloaded. In case of contracted peers, the rarest value is computed and maintained only for the subset of the non-downloaded pieces, which does not include the contracted pieces. This ensures low availability pieces are always downloaded faster within locality. Let's take an example of a file with 10 pieces and assume that peer A and B are contracted such that A downloads pieces IDed 1-5 and B the rest. Also assume for the swarm that piece ID 8 would be the rarest piece and all peers within the swarm try to replicate this piece 8 faster. But during this time frame, A based on its responsibility would have a different rarest piece thus effectively replicating content faster ,if it is able to find that piece in its neighbour set.

But this does not change the piece availability within the swarm because the piece within the contract is still rarest in the swarm; the other peers with that contract will apply LRF since it has to download this piece from outside locality and once the piece is available within locality, it is exchanged among the peers. The above property ensures that the availability is preserved. When there are more than two peers involved in a locality based transfer, the pieces can be requested across any peer within the locality as long as the requests are sent outside with unique piece IDs. For example, if peer A and B establish a contract such that peer B would get the piece ID:5 from A as part of the contract. If peer A had subcontracted the non-local request of piece 5 to peer C , then peer B would be able to get that piece from peer C directly, if peer C also has also established the contract with B else peer B would have to wait until peer A downloads from C. This feature is allowed in both Altruistic and Fair modes.

Even if the contracted peer exits the system without completing the download,

the contracts time out and either the pieces become part of a new contract or follow the native Rarest First(RF) policy among the non-local peers within the swarm. The swarm does not get partitioned into local swarms by establishing contracts, since the Ono plugin ensures that there exists a good ratio between non-local and local peers to avoid, reduction in piece availability as well as disjoint swarms.

The existing rarest first algorithm ensures that piece availability is maintained within the swarm by replicating the rarest pieces first with high priority. For peers within a contract, the contracted pieces are of all same or low priority compared to non-contract pieces. This is so, since the peer has delegated the responsibility to request the piece within the locality to another peer. While for non-contracted pieces, the regular RF algorithm is applied.

This concludes the various stages of the SOS algorithm. In Section. 7 we evaluate this implementation in PL and live swarm. The changes to the code are mentioned in Appendix.B.

In the next Section. 6, we discuss about properties observed in the various live swarm and whether the observed inherent properties support the deployment of SOS in the real world.

Chapter 6

Live Swarm Analysis

In this chapter we analyse trace logs collected from different large BT swarms and discuss about the occurrence of local peers. We show that the inherent properties observed in the live swarm in the real world supports the deployment of SOS. We discuss about the other various properties that include 1) AS hop count, 2) Latency and 3) Throughput distribution among the peers. We also discuss about the effectiveness of using CRP technique as a filter in SOS algorithm to filter local peers.

6.1 Setup

We logged trace information for 10 unique torrent files ,which had a large swarm size from the site "http://btjunkie.org/" based on the swarm statistics provided along with the torrent file. On an average each swarm had over 10,000 nodes involved in sharing the file pieces. Even though btjunkie site mentions swarms varying of size greater than 50,000 nodes, it is not always accurate since the trackers hold stale information and do not update them,if the peers exit the swarm.

We have implemented and set up an IP address collecting software from the tracker called *ConTracker* on 22 different PL nodes across different geographical locations. Its duty is to parse the given torrent file, join the swarm and collect the neighbour information; but does not download any data. We used this technique to get a global view of the swarm and the collection of IP addresses from various vantage points within the PL setup. The *ConTracker* contacted the tracker for every 30 minutes, for over a period of 20-23 hours to

get the list of IP addresses. From the obtained IP:PORT information, we try to establish a TCP connection with the peer to ensure that the BT software is available at that port. This technique helps us to avoid stale peer information from the tracker that would have polluted our results. The steps involved in the process are provided below.

1. *Parse the torrent file and get the address of each tracker.*
2. *For each tracker, contact the tracker and obtain 50 IP addresses of peers involved in the swarm.*
3. *Try to establish a TCP connection to the port number provided by the tracker, to ensure that the BT/Peer is alive at that time.*
4. *Execute Step 3 for the IP address collected in the previous Iteration.*
5. *Sleep for 30 minutes.*
6. *Back to Step 2.*

6.2 Results

6.2.1 Occurrence of local peers

We define locality as peers within the same AS. Based on this premise we analyse the trace logs to find total number of unique peers within the same AS. We use Team Cymru's IP to AS mapping [34] information to map each IP observed to its AS number. Each AS was given a unique ID and ranked on the quantity of peers within the same AS. Based on that information we plot the Fig. 6.1. Any point on x-axis represents a unique AS identified from the swarm and the y-axis represents the total number of peers within the same AS. We have identified over 3000 unique ASes and maximum number of observed IPs within each AS varies from 1100 to 7000. From the graph we can see that, at least 100 unique ASes have at least 100 unique peers in all the swarms. This graph includes all the peers returned by the tracker(including the non-responsive peers).

The above information does not provide the availability of local peers in temporal region. To analyse locality with respect to time we plot the graph with x-axis representing time in minutes; the duration in which the trace log was collected and y-axis represents

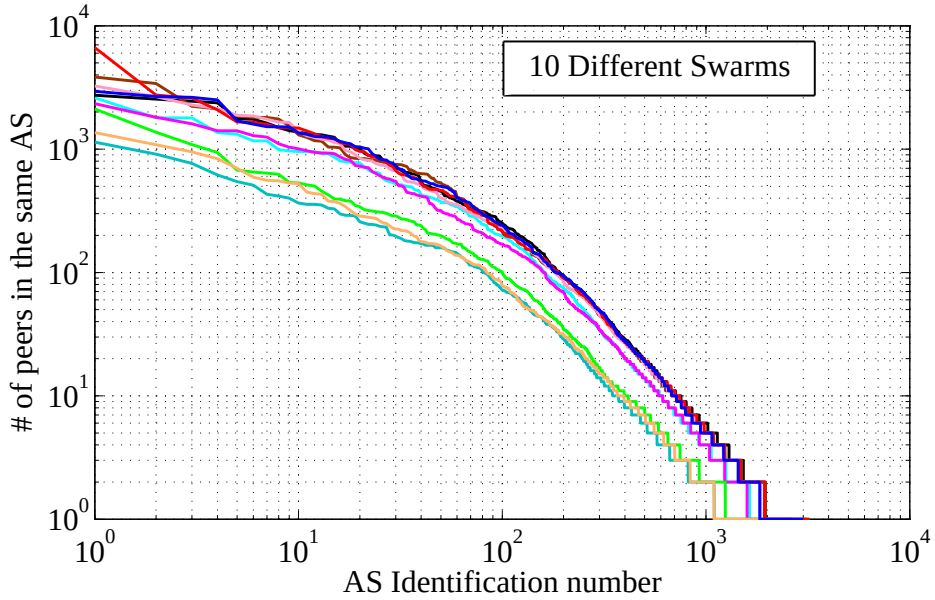


Figure 6.1: The number of peers in each AS for 10 different torrents ranked by the observed size.

the probability of finding an observed AS with more than one peer. Similar to the above technique we mapped all peers to their respective ASes, then counted the AS with only one peer. Let x be the number of AS with only one peer and y be the number of unique AS observed within a swarm. Then *1 - x/y* represents the probability of finding more than one peer within the same AS. We plot this information in Fig. 6.2 as observed over time. As seen in the graph, we are able to show the existence of 50% probability in finding more than one peer within a swarm at any given point in time. This shows the occurrence of local peers within any observed AS.

6.2.2 AS hop count distribution

For one of the observed swarm, we represent the distribution of AS hop count of the various peers. We randomly selected 1000 IPs from one of the swarm and then generated all-pairs amongst them. we have over 499,500 source,destination pairs from the generated set. We used a combination of iPlane database and Team Cymru's IP to AS mapping to obtain the AS hop count information to extract the network distance information among peers. Specifically, for a given pair if iPlane is not able to provide or predict the information, we

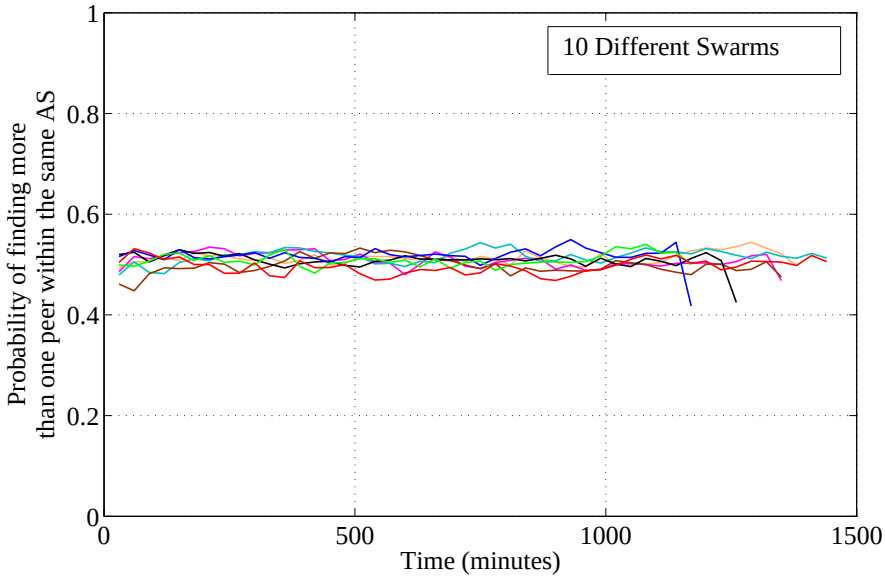


Figure 6.2: Probability of finding more than one peer within the same AS, in the same swarm.

use the corresponding AS number provided by Team Cymru to find already existing entry returned by iPlane and reuse the AS hop count information. This helps us to improve the probability of identifying the AS hop count for a given pair.

Using the above technique we were able to obtain AS hop count information for 80% of the source-destination pairs and plot the Fig. 6.3. From the graph we observe that 38% of the peer pairs have a hop count of zero thus both IPs are within the same AS. This helps our SOS protocol to be deployed easily in the real swarm due to the high availability of local peers since depends on local peers to establish contracts.

6.2.3 Is CRP an effective filter?

As mentioned in Section. 5 we use Ono plugin which uses the CRP technique to act as a filter to short-list peer IPs provided by the tracker. If this approach drops peers that are local due to its inefficiency, SOS would miss out on opportunities to reduce redundancy. And hence we quantize the effectiveness of CRP technique to be used as a filter in SOS.

To quantize the efficiency of the CRP technique, we use the recursive Domain

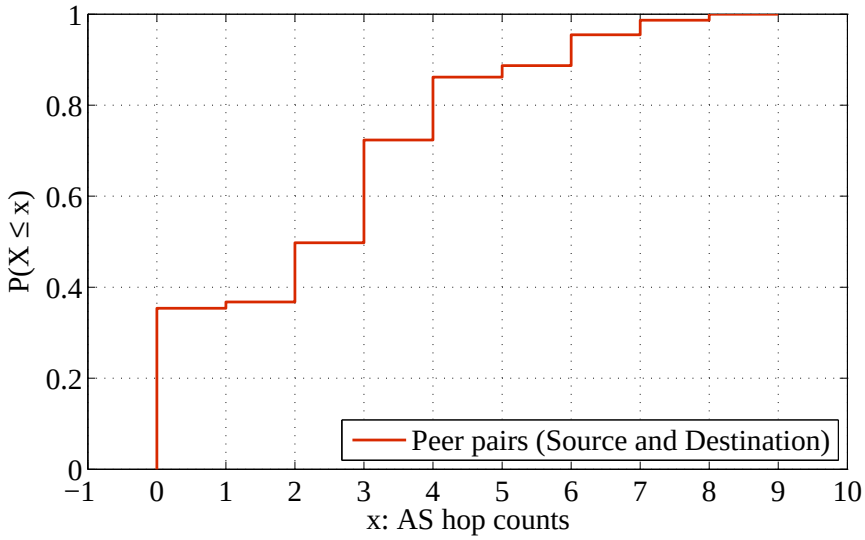


Figure 6.3: CDF of AS hop count for the 499,500 IP pairs, for which iPlane gave us results for 394,656 pairs.

Name System(DNS) queries to obtain the edge server names for one of the swarm's trace log. This technique is used by Ono to measure edge server ratios for a BT client that does not have Ono plugin installed. We use this technique since we do not have access to the client with that IP.

The steps for this technique are described below.

1. Query the local DNS to obtain the authoritative DNS server name for a remote peer's IP address.

2. Issue a recursive DNS query to the authoritative server for resolving the CDN server's canonical name which responds with the list of CDN edge server close to the remote peer.

Due to security reasons many of the authoritative DNS do not support the recursive DNS queries[35]. The successful queries were 12.5% from which we categorized them into the 305 edge server, that were identified as part of CDN redirections. For the peers redirected to the same edge server we created all pairs amongst them and calculated the AS hop count by using the same technique described above. For example, if we had three unique IP

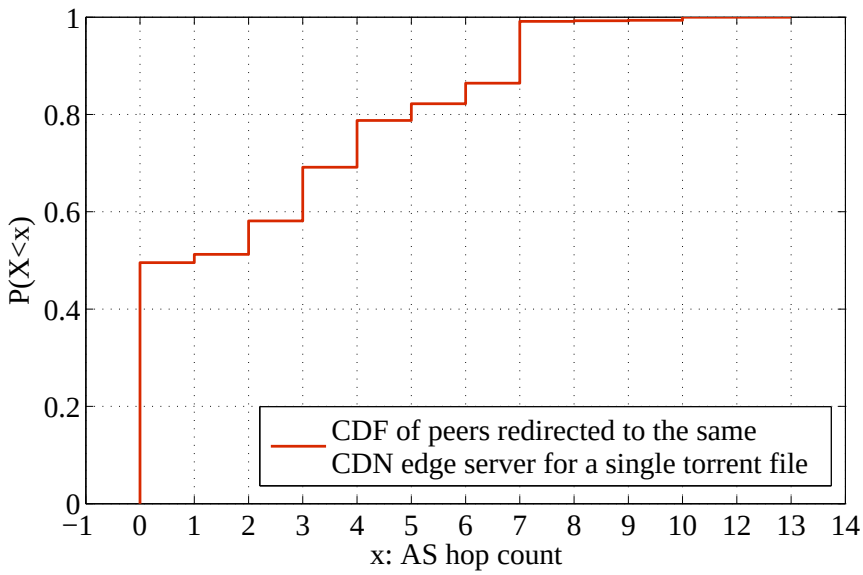
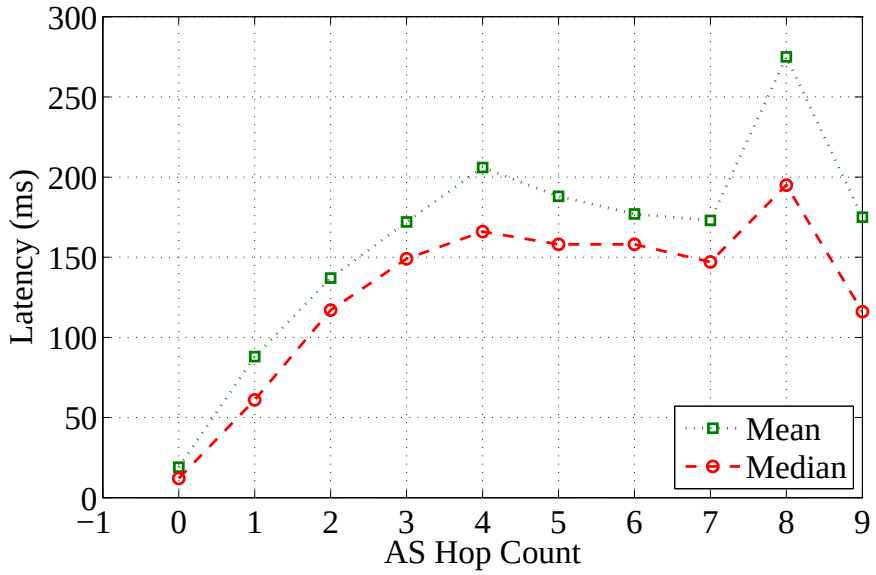


Figure 6.4: CDF of AS hop count for peers redirected to the same CDN edge server.

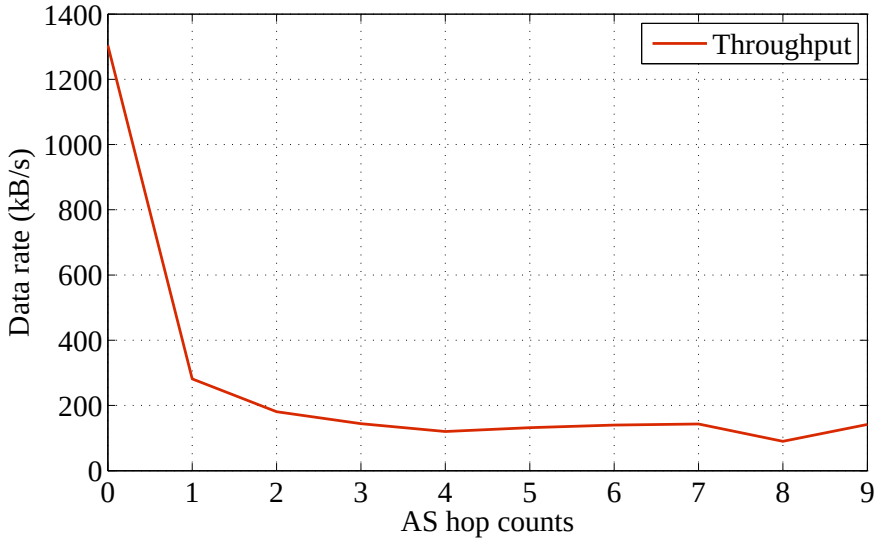
addresses associated with an edge server, then we created three unique IP pairs. Based on that information, we plot the CDF of the AS hop count among peers that were redirected to the same edge server in Fig. 6.4. The result shows that about 50% of pairs are within the same AS thus proving, CRP technique is a effective filter to short-list peers and does not lose out on local peer information.

6.2.4 Relationship between AS Hop count and Throughput

To represent the relationship between AS hop count and Latency, Throughput we use the latency information obtained from the iPlane for the 1000 random IP pairs to plot the Fig. 6.5. In Fig. 6.5 a). We can see the increase in latency as the AS hop count increases. But, after the hop count five we see a drop in latency. This comes from the limitation that the iPlane database has insufficient number of instances for AS hop counts more than five. But we can clearly see that latency value is directly proportional to the AS hop count. Also we do not have direct access to the peers involved in the swarm and hence we cannot measure the throughput directly between the public peers observed in the BT swarm. So we used the TCP throughput Formula. 6.1 introduced in [33, 19] to calculate the



(a) Latency



(b) Throughput

Figure 6.5: Latency and throughput for corresponding AS hop counts.

throughput based on the latency returned by iPlane. The loss rate returned by iPlane is always zero, so we used 0.5% loss rate [32] in the equation, Maximum Segment Size(MSS) 1460 Bytes along with Round Trip Time(RTT) provided by iPlane for the formula. As we can see in Fig. 6.5 b), the increase in AS hop count, deteriorates the TCP throughput thus, choosing local peers potentially is beneficial in improving download completion.

$$TCPthroughput = \frac{MSS * 1.2}{RTT * \sqrt{lossrate}} \tag{6.1}$$

In this section we have discussed about the various properties observed in a live swarm that are conducive towards the deployment of SOS in the Internet. 1) Availability of local peers in a swarm, 2) Local peers have low latency and 3) CRP technique helps to short list peers from the large set returned by the Tracker.

Chapter 7

Evaluation

In this chapter we discuss about the various experimental setups and the evaluation of SOS algorithm.

7.1 Setup

We use over 100 randomly selected PlanetLab nodes geographically distributed for our experiment. We implemented SOS on a Java based BT client Vuze, with CRP implementation already available as Ono plugin. We installed our implementation with two modes, Altruistic and Fair on each of the selected PL nodes. The implementation involves: 1) Modification to the rarest first algorithm; the priorities of contracted pieces are considered separately from the non-local pieces, 2) Locality detection techniques and 3) Addition of contract setup, store and complete procedures. We have made two modification to Ono 1) The cosine similarity ratio calculated by Ono is made accessible within the Vuze core and 2) The edge server ratios is regularly updated with the neighbours to avoid missing out on candidate local peers. SOS is provided as a complete application and not as a plugin since it needed code changes in the core piece selection algorithm. The gist of the setups used are tabulated in Table. 7.1. We used a public tracker [28] to host the torrent file to share file content of size 500MB(2000 pieces). We also configured an initial seed within PL and emulated the flash crowd.

Since some of the PL nodes timeout while executing the experiment due to environment issues, we were not able to find the average among the various iterations since

Table 7.1: Evaluation Setup

Environment	Throughput Limitation	File size	Number of Nodes	Comments
PL	NO	$\tilde{7}$00MB	10 + public peers	Live swarm
PL	NO	500MB	100	Randomly selected peers(Flash crowd)
PL	YES	500MB	100	Randomly selected peers(Flash crowd)
Residential ISP	NO	$\tilde{4}$00MB $\tilde{2}$00MB	2 + public peers	Live swarm

some nodes do not exist in all the iterations. To overcome this disadvantage we aggregated the data for all iterations and plotted the CDF of peers with time. To compare cross-ISP traffic in bytes, only the common nodes across the algorithm iterations were considered for plotting the CDF.

Also we define cluster size as the total Number of nodes within locality that are collaborating amongst themselves.

7.2 Without Throughput Limitation (WOTL)

We evaluate SOS without throughput limitation initially to avoid bias due to bandwidth limitations.

Cluster Size

For the random selection of 100 PL nodes we show the distribution of local nodes within the swarm in Fig. 7.1. we observe that 40% of the nodes within the swarm in both SOS-A and SOS-F were able to find at least one node to collaborate with. This is similar to the observation in Fig. 6.3 from the live swarm analysis.

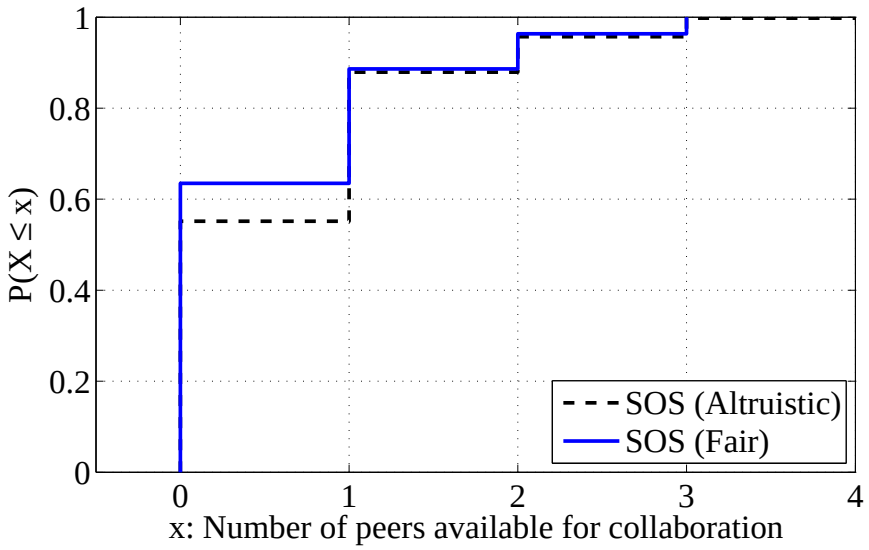


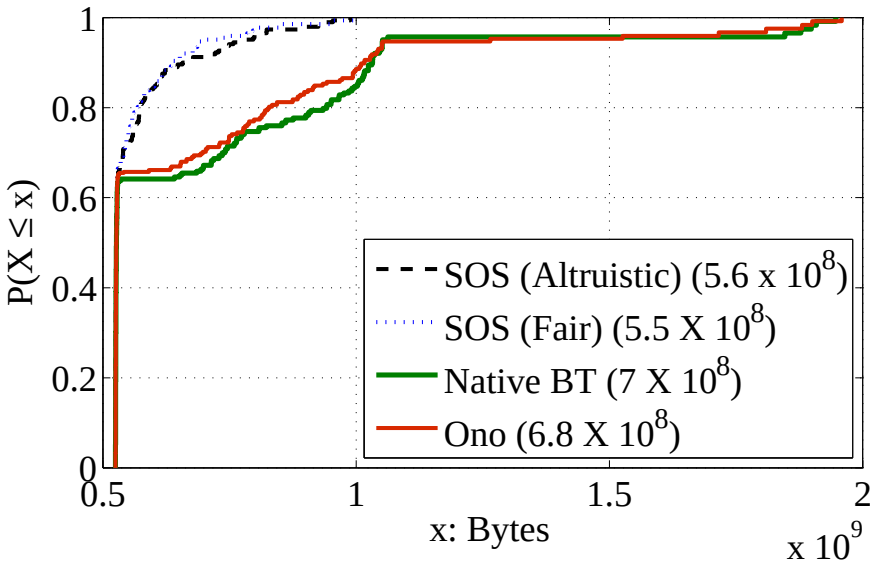
Figure 7.1: WOTL: CDF of peers within the swarm available to collaborate.

Downloaded Bytes

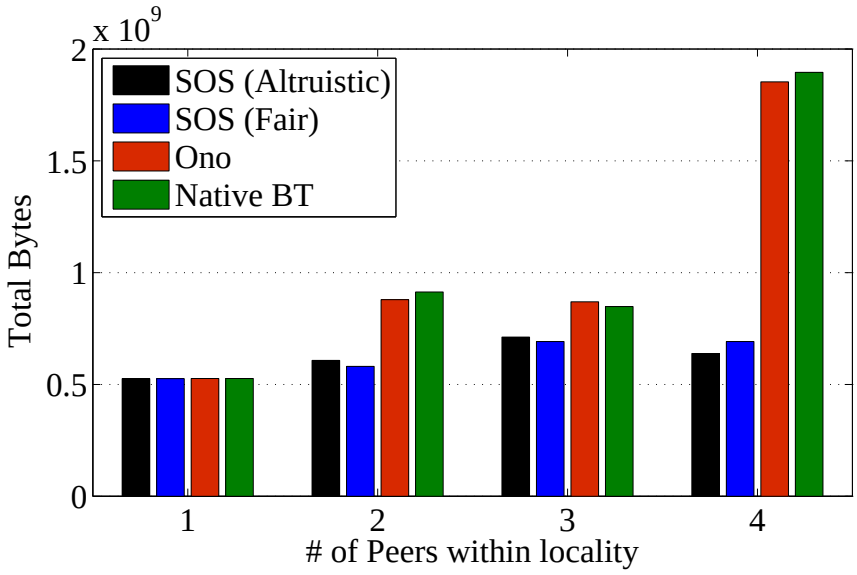
We executed 5 iterations for each algorithm and compared the performance of Ono, SOS-A and SOS-F with Native BT. We plot the performance comparison with respect to 1) Reduction in redundancy and 2) Mean bytes downloaded from outside locality in Fig. 7.2 (a) and (b) respectively. As observed in Fig. 7.2(a) for SOS, 40% of the nodes in the swarm are able to identify locality and reduce the redundant traffic; In extreme cases with cluster size of 4 , SOS reduces redundancy by half in comparison with Ono and Native BT. For the rest 60% whom do not find nodes within their locality, the actual file size is downloaded from outside the locality. This information is correlated by the findings in Fig. 7.1. The impact of reducing the redundancy helps in reducing traffic from outside the locality. Finally in Fig. 7.2(b) we show the mean bytes downloaded from outside locality for each cluster size and SOS is effective in reducing redundancy, outperforms other techniques for all observed cluster sizes within the swarm. Important observations from this setup are tabulated in Table. 7.2.

Table 7.2: WOTL: Comparison with Native BT

Average	Ono	SOS(Altruistic)	SOS(Fair)
Cross-ISP traffic reduction	2.8%	20%	21.42%



(a) CDF of peers: Downloaded bytes from outside locality.



(b) Mean bytes downloaded from outside locality for each available cluster size.

Figure 7.2: WOTL: Comparison of bytes downloaded from outside the locality.

Download Completion Time

As observed in Fig. 7.3, download completion time for SOS is slow compared to Ono or BT. We have observed that the increase in AS hop count results in high latency in Fig. 6.5 which affects the throughput of the data flows. We also set up contracts with nodes within the AS to avoid redundant requests outside the locality but also to exploit the property; latency among the nodes within the same AS would be less compared to a node outside the AS. This is due to 1) data flow does not cross the AS egress routers which are policed by bandwidth limitations as well as traffic policies. 2) avoids congestion on the egress routers. But PL is based on the educational network in which the relationship between AS hop count and latency is not similar to the real world. Thus downloading content from a node with AS hop count 3 is same as AS hop count 1(including source and destination) with negligible time difference which is the cause of slow download completion time for SOS in comparison with existing technique.

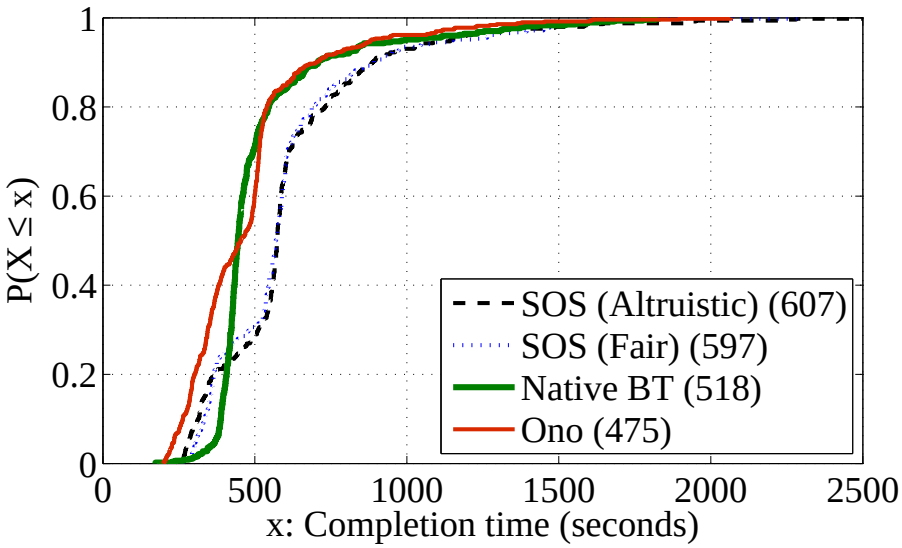


Figure 7.3: WOTL: CDF of peers within the swarm able to collaborate.

Table 7.3: WTL: Comparison with Ono

Average	SOS(Altruistic)	SOS(Fair)
Cross-ISP traffic reduction	17.69%	17.53%

7.3 With Throughput Limitation(WTL)

Most of the nodes within PlanetLab are connected by GREN, an educational network which includes Internet2 [29] in US and Dante in Europe. Thus characteristics observed while analyzing the trace information collected by PL nodes for studying P2P systems would be way off with respect to the general global Internet. In order to asymptotically emulate the Internet with PL, we use the throughput mapping of loss rate and latency information shown in Fig. 6.5 depending on the AS hop counts. This artificially limits the bandwidth of Vuze between PL node pairs not located in the same AS.

The throughput limitation is applied before the peer starts to request data by executing locality detection technique given in Section. 5 thus, we can exclude any disturbance for limiting bandwidth. To avoid the recursive locality detection operations and resulting loads to routers as well as database servers, we cache the AS hop count information and recycle it through the experiments. Note that the cached information is valid through our experiments considering the time-scale of changes in AS hop counts.

we have used the similar set of PL nodes as seen in the previous setup, hence we observe 40% of the nodes have the ability to form local clusters. This information is represented in Fig. 7.4. The reduction in downloaded bytes from outside locality are similar to our analysis from the previous Section. 7.2. We represent the bytes downloaded non-locally as well as the average bytes per cluster size in Fig. 7.6(a) and (b). The important observations are tabulated in Table. 7.3.

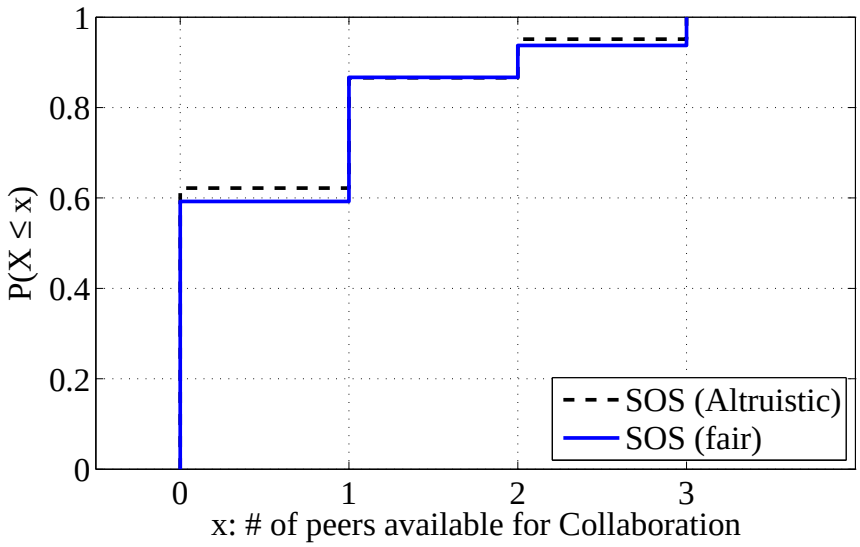


Figure 7.4: WTL: CDF of peers within the swarm available to collaborate.

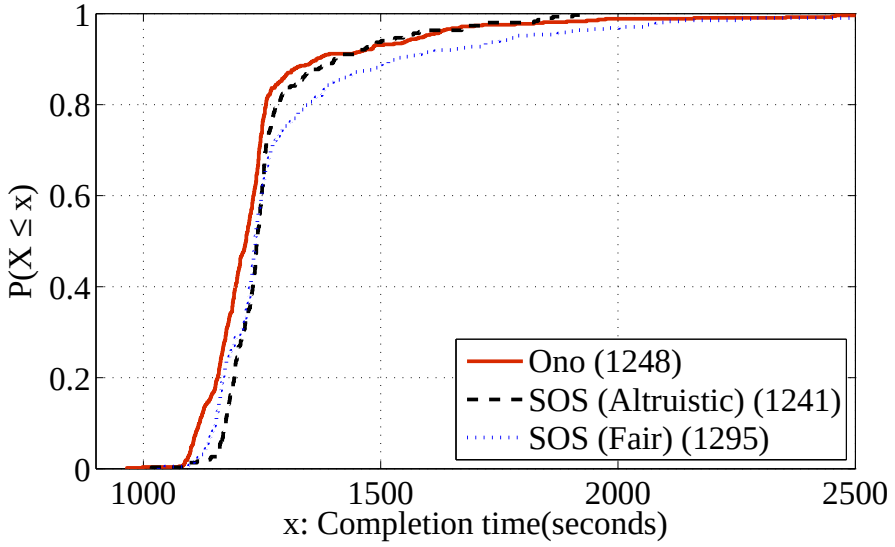
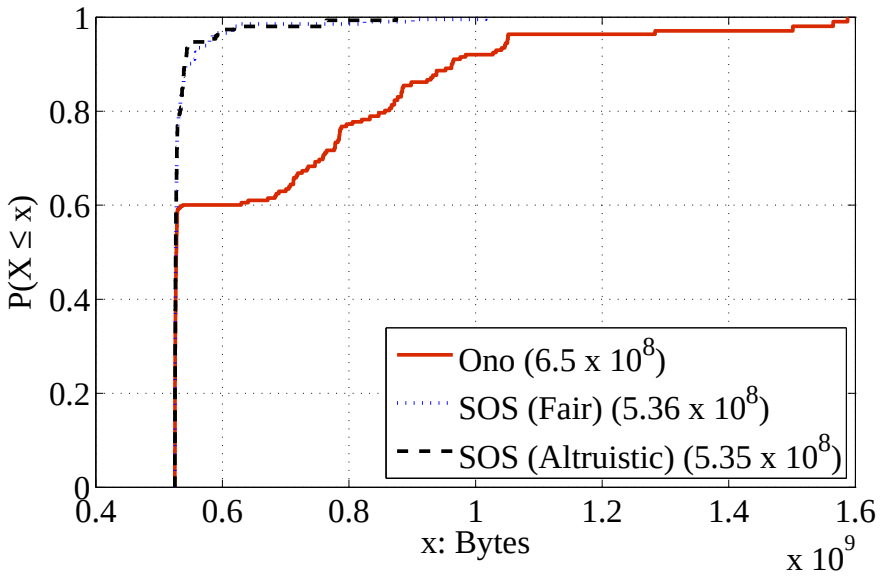
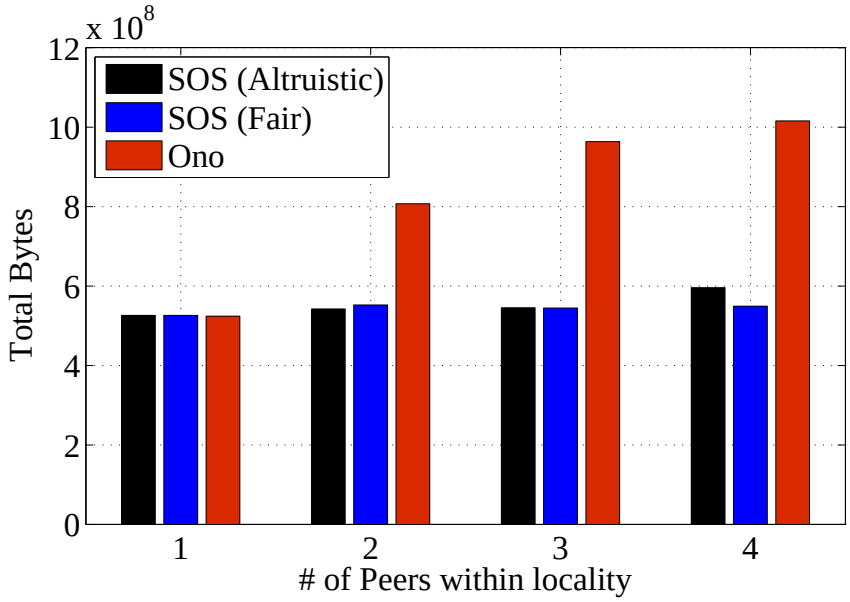


Figure 7.5: WTL: CDF of peers within the swarm able to collaborate.



(a) CDF of peers: Downloaded bytes from outside locality.



(b) Mean bytes downloaded from outside locality for each available cluster size.

Figure 7.6: WTL: Comparison of bytes downloaded from outside the locality.

Download Completion Time

To emulate real world Internet in PL, we enforced bandwidth limitations. From the modified setup we observe in Fig. 7.5 that the SOS download completion time converges with Ono. We attribute few reasons as to why SOS is not able to achieve a large % reduction in completion time compared to Ono 1) The cumulative bandwidth of all non-local connections for Ono is greater than that of SOS, since we establish contracts for at least 50% of workload. Due to this we lose out on time. But this is a trade off that can be sacrificed to reduce cross-ISP traffic, thereby reducing the congestion on the various egress routers. This is conducive to other traffic including FTP [42] and HTTP [43] in the Internet.

7.4 Varying Local nodes

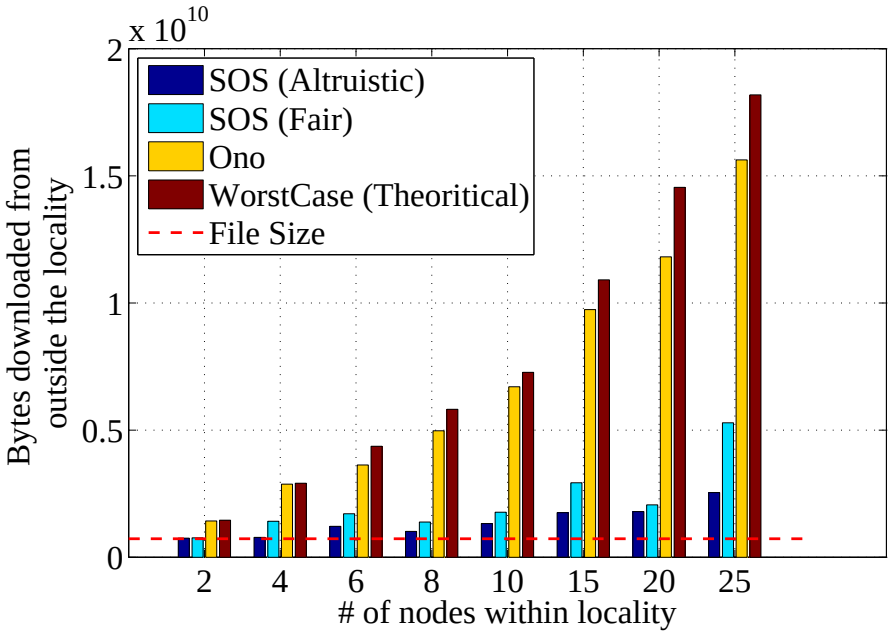


Figure 7.7: Comparison of non-local bytes for various cluster sizes

Up till now we have shown that SOS out performs Ono and Native BT by improving local transfers and reducing redundancy from non-local nodes thus reducing cross-ISP traffic. We have only been able to observe cluster with a maximum size of four within PL.

To quantize the performance with cluster size greater than four we conducted another set of experiment. We measured the reduction in redundancy for SOS compared to Ono for each cluster size ranging from 2 to 25 and the results are plotted in Fig. 7.7.

To conduct this experiment we have to identify a cluster with nodes more than 4. But within PL we were not able to find locality with nodes more than 4 (i,e) within the same AS. To overcome this disability, we simulated nodes to be part of a locality by creating contracts among them irrespective of their AS number or their hop count. The varying throughput within the nodes is attributed to the homogeneous link ratios between peers in a BT swarm. The following changes in code were made to ensure that the setup was unbiased towards any specific algorithm.

We injected the identified peers to be local while bootstrapping the nodes 1) So they establish contracts with these nodes immediately in case of SOS. 2) To ensure that Ono does not disconnect these peers due to their cosine similarity value, and for these peers to be always part of the preferred list.

Using the above changes we joined a public swarm to download the Ubuntu ISO image of size $\tilde{7}$00MB and plotted the redundant traffic from within this simulated locality. As seen in Fig. 7.7 we observe that SOS outperforms Ono irrespective of the cluster size.

With a cluster size of 25 we observed that many of the nodes were not able to establish contracts with all the nodes within the locality due to unavailability of pending pieces. Consider a scenario in which a file is split into 2000 pieces. By dividing the workload equally among any peer, a peer will distribute its workload to a maximum of 10 peers after which it will not have any other pieces to distribute further even if there were local nodes available. In this case the contracts cannot be established since there is no more work load to distribute. Thus in the Fig. 7.7 the increase in cross-ISP traffic for nodes 20 or 25 compared to small cluster size is attributed to this. The vast difference in SOS-A to SOS-F for cluster size of 25 is because of the size of contract and SOS-A does not follow TFT strictly. Irrespective of this difference, we are able to get a consistent 80% and 70% reduction in redundancy in SOS-A and SOS-F for cluster sizes greater than 10.

We restricted our tests to a maximum of 25 nodes only, since 50 is the maximum number of parallel connections supported by Vuze and Ono ensures that 50% of the available connections are local and the rest non-local. This property is enforced to maintain the availability of pieces within the swarm and avoid disjoint swarms.

From this test we can conclude that SOS algorithm is scalable and consistent in reducing redundancy irrespective of the cluster size.

7.5 Residential Network(RN)

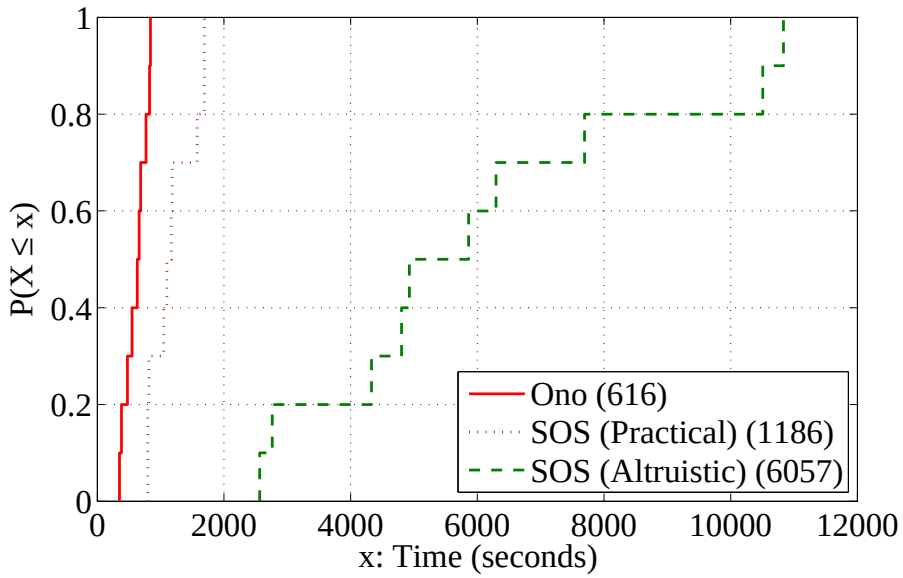
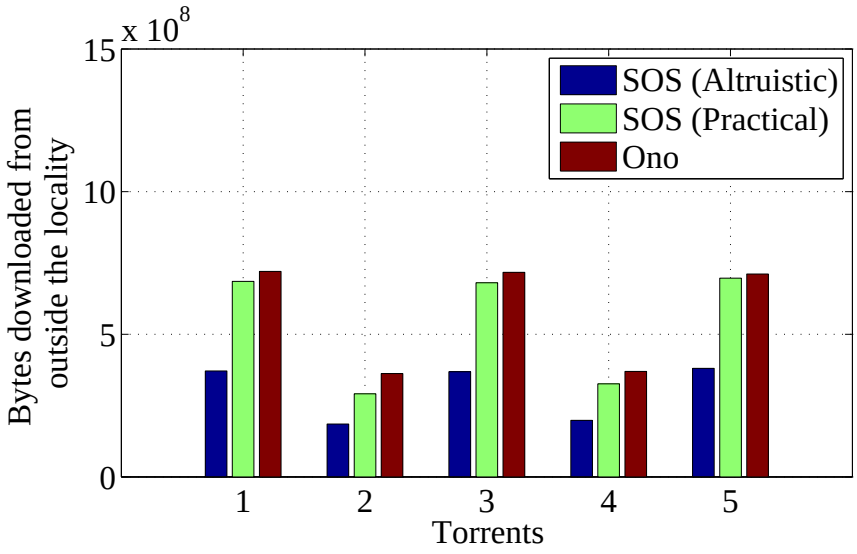


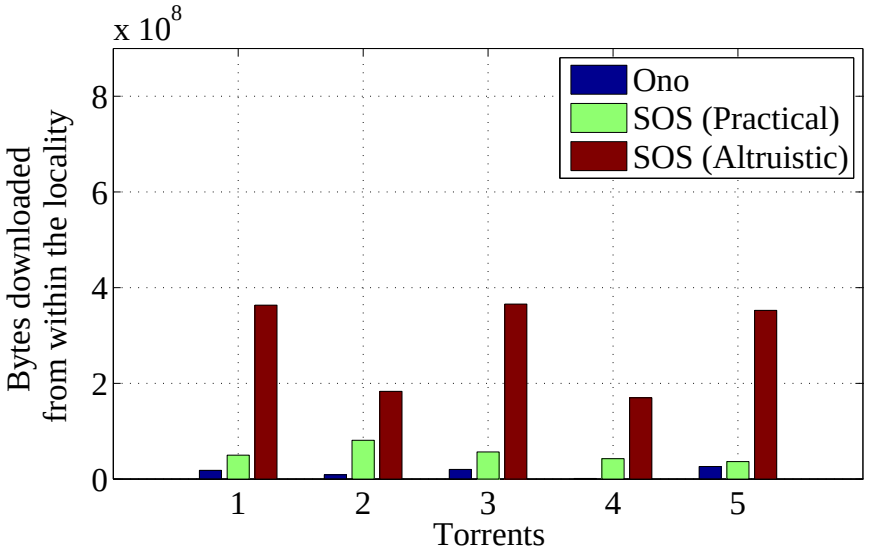
Figure 7.8: CDF of download completion time for 5 different torrents.

We deployed two peers A and B within the same AS with the SOS protocol and joined over 5 active swarms and logged the following information. 1) Did the two nodes within the same AS identify themselves to be local and setup contracts. 2) After the contracts are established; the reduction in cross-ISP traffic compared to Ono. 3) Difference in download completion time 4) Other advantages to the peer implementing the SOS protocol.

We manually inject the peers A and B into each others active peer list to ensure that these peers are selected as neighbours. For each neighbour, then based on the exchange of cosine similarity values and local peer detection, SOS is able to detect local peers with 100% success rate for all torrents. Once local peers are detected, the contracts are established using SOS protocol. Based on the empirical results as seen in Fig. 7.8, we have been able to find that the collective download rate from non-local peers is high compared



(a) Downloaded bytes from outside the AS



(b) Downloaded bytes from inside the AS

Figure 7.9: Comparison of both non-local, local bytes for 5 different torrent files with respect to two nodes

to the download rate from the local peer with which contracts are established. To overcome this disadvantage, we propose a variation to the contract protocol to facilitate reduction in contract size once the non-local pieces are downloaded. This enables the SOS to achieve similar download time as that of the existing BT protocol but also achieve a increase in reduction of cross-ISP traffic compared to Ono.

The new variation is called SOS(Practical) that aims to achieve 1) Incremental deployment in the real world; Since all the local peers detected would not be deployed with SOS protocol, thereby losing out on TCP connections. 2) Better download completion time compared to SOS-A or SOS-F.

SOS(Practical)

From our observation of the logs, we find that non-local pieces download from outside the AS occur at a faster rate compared to local pieces shared among the contracted peers. Due to this reason, the local sharing becomes the bottle neck for download completion. This occurs since 1) The other local peers identified do not have SOS deployed. To facilitate the smooth transition of SOS algorithm in the real world and not to penalize the peers with SOS compared to Ono, we introduce SOS(Practical). For the sake of brevity we call SOS(Practical) as SOS-P.

After the non-local piece download is complete, the remaining local pieces yet to be downloaded from the contracted peer is renegotiated such that the remaining workload is divided into two parts based on the non-local and local download rate. Assume that if the file has 100 pieces, and 50 pieces were part of the contract. Assume that *peer A* executes and 1)50 pieces were downloaded non-locally while only 10 pieces have been downloaded from the contracted *peer B*. 2) Then the remaining workload is split into two parts with the ratio 5:1. 3) In this case *Peer A* will continue to download 35 more pieces from outside locality and continue to get retained 5 pieces from the other contracted peer. This renegotiation step continues until the download completes successfully. This enables incremental deployment in the real world.

Since we execute this test only for flash crowd scenario, SOS-A and SOS-F act similarly; hence we compare Ono only with SOS-A and SOS-P. In Fig. 7.9 we show that SOS-A is able to reduce nearly 100% redundancy and downloads only the actual file size within

Table 7.4: RN: Comparison with Ono

Average	SOS(Altruistic)	SOS(Practical)
Cross-ISP traffic reduction	48%	7%
Improvement in local traffic	1827%	258%

the locality even though there are two nodes downloading the content. SOS-P is not able to achieve similar reduction in redundancy due to the renegotiation of the contracts. But it performs better than Ono in all tests. The important results are tabulated in Table. 7.4

In Fig. 7.10(a) we plot the Number of local peers(excludes SOS peers) and the aggregated bytes downloaded from non-SOS peers in Fig. 7.10(b). Even this downloaded bytes contributes towards the local traffic since the nodes are within the same AS. From this we observe that local peers exist and we are not able to leverage their locality due to non-availability of SOS protocol in those peers. To overcome this drawback, as explained above we use SOS-P and show its performance is better than Ono in reducing redundancy with a slight degradation in download completion time.

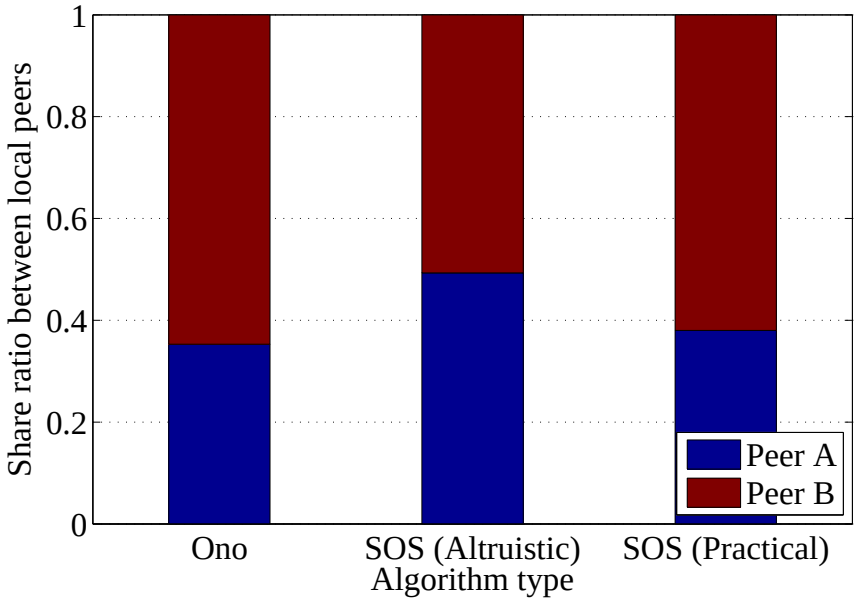
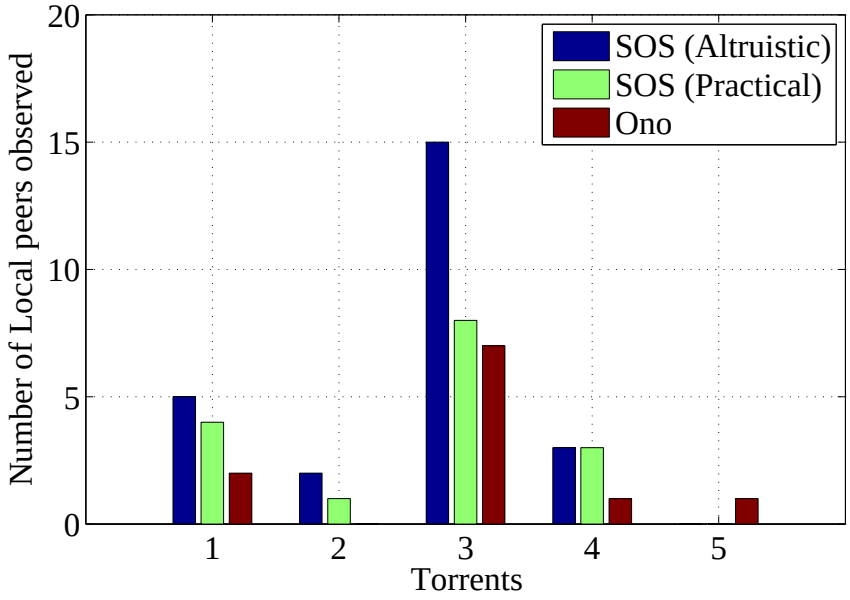
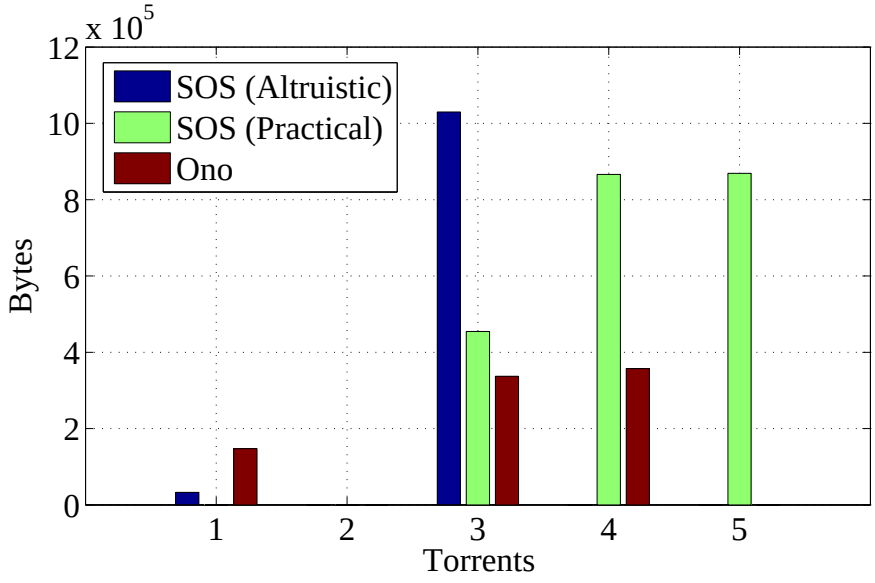


Figure 7.11: Sharing ratio between the collaborative peers.

Fig. 7.11 shows the average sharing ratio between SOS and Ono. We observe that



(a) Number of local peers available within the same AS, excluding the collaborated peer



(b) Downloaded bytes from non-SOS local peers

Figure 7.10: Comparison of local peers observed and their contribution with respect to two nodes

SOS-A has a good sharing ratio since we split the workload equally in flash crowd scenario. But since the cumulative download rate in Peer A and B for non-local pieces are the same we observe that the sharing ratio of SOS-P is similar to that of Ono. This shows that the use of SOS does not affect the sharing ratio between the contracted peers thus proving to be conducive for the swarm.

Advantages of SOS

The advantage of SOS, is not only about the reduction in cross-ISP traffic but also regarding the network load on the peer. After downloading its half of the non-local pieces, the peer does not have to contact the tracker to get a new set of peers to ensure that the other half of the local pieces to be downloaded. Because it knows for sure that the other half would be provided by the contracted peers. The contracts can be enforced by variety of ways that include TFT as well as encryption to avoid free riding within the system. Few major advantages of SOS are provided below:

1. SOS protocol reduces the congestion on the AS egress routers, by reducing BT traffic across ASes. This is conducive for other protocols such as FTP, HTTP and media streaming. Due to the aggressive nature of BT protocol other Internet protocols suffered due to lack of throughput. SOS would help to mitigate this issue.

2. In a normal BT download, the client keeps connecting to the Tracker to get new set of neighbours with which he can connect, to get the pieces. In SOS, the number of requests to the tracker would be halved since the workload is distributed.

   For the 50% of the total pieces which are outside the contract, the client would contact the tracker to obtain the peer list that might contain the required pieces. For the other 50% which are part of the contract the peer in the other end of the contract takes responsibility. And since both peers download simultaneously from outside locality the probability of missing out on availability of pieces (due to peers leaving the swarm on completion) is reduced, compared to a normal BT protocol.

3. Using SOS protocol by the users helps to reduce costs for the ISP, which can be redirected by the ISP to improve on local infrastructure within the AS to improve customer service.

4. When there is less traffic on the egress routers, there is no need for the ISP to police its links to either drop P2P packets or prioritize traffic thus inherently supporting net neutrality.

Chapter 8

Conclusion

This work has examined the possibility of 1) Identifying peers within a locality with high accuracy, 2) setting up mutually exclusive contracts amongst them to reduce their (i) workload, (ii) redundancy in piece requests from inside the locality, 3) Altruistic, a new variant of the TFT; locality aware local sharing of pieces. Contracts reduce redundancy, which help in reducing cross-ISP traffic. We have designed and implemented the contract protocol, SOS and evaluated its performance in Planetlab and deployed it in the real world. Our results indicate, SOS helps in reducing cross-ISP traffic effectively compared to the existing prevalent technique Ono, but with a small degradation to the download completion time. We observe for a two peer setup, the upper bound is 48% and lower bound is 7% reduction in redundancy. We also show with increase in SOS peers within locality, it further reduces the download completion time and thus able to compete with existing ISP-agnostic approaches. The protocol is distributed, scalable and the complexity of the algorithm is always between two peers. we have also analysed public swarms trace logs to show the high occurrence of local peers which allows the protocol to be easily deployed in the real world. We believe there are several additional features that can be implemented to make SOS suitable for public consumption. One interesting direction is to filter peers for contracts not only based on locality but also consider the available throughput with that peer. This enables faster transfer thus making SOS a good choice for download acceleration also in addition to reducing cross-ISP traffic. The ultimate aim of this BT system is to reduce redundancy in file pieces within locality while not sacrificing download completion time.

Bibliography

[1] Bram Cohen.(2003) *Incentives Build Robustness in BitTorrent.*

[2] Bram Cohen.(2005) *BitTorrent documentation: Protocol.*

[3] Karagiannis, Thomas and Rodriguez, Pablo and Papagiannaki, Konstantina. (2005) *Should internet service providers fear peer-assisted content distribution?*

[4] Minghong Lin and Lui, J.C.S. and Dah-Ming Chiu. (2008) *Design and analysis of ISP-friendly file distribution protocols*

[5] Aggarwal, Vinay and Feldmann, Anja and Scheideler, Christian. (2007) *Can ISPS and P2P users cooperate for improved performance?*

[6] Dischinger, Marcel and Mislove, Alan and Haeberlen, Andreas and Gummadi, Krishna P. (2008) *Detecting bittorrent blocking*

[7] Madhyastha, Harsha V. and Isdal, Tomas and Piatek, Michael and Dixon, Colin and Anderson, Thomas and Krishnamurthy, Arvind and Venkataramani, Arun. (2006) *iPlane: an information plane for distributed services*

[8] Choffnes, David R. and Bustamante, Fabián E. (2008) *Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems*

[9] Su, Ao-Jan and Choffnes, David and Bustamante, Fabian E. and Kuzmanovic, Aleksandar. (2008) *Relative Network Positioning via CDN Redirections*

[10] Shay Horovitz and Danny Dolev. (2008) Liteload: Content unaware routing for localizing p2p protocols. *Proceedings of International Workshop on Hot Topics in Peer-to-Peer Systems*

[11] Guobin Shen and Ye Wang and Yongqiang Xiong and Ben Y. Zhao and Zhi-li Zhang. (2007) HPTP: Relieving the tension between ISPs and P2P. *Proceedings of IPTPS*

[12] Garbacki, Pawel and Iosup, Alexandru and Epema, Dick and van Steen, Maarten. (2006) 2Fast: Collaborative Downloads in P2P Networks. *Proceedings of IEEE International Conference on Peer-to-Peer Computing*

[13] Vishnumurthy, Vivek and Francis, Paul. (2008) On the difficulty of finding the nearest peer in p2p systems. *Proceedings of ACM IMC*

[14] Ren, Shansi and Tan, Enhua and Luo, Tian and Chen, Songqing and Guo, Lei and Zhang, Xiaodong. (2010) TopBT: a topology-aware and infrastructure-independent bittorrent client. *Proceedings of IEEE INFOCOM*

[15] Bindal, Ruchir and Cao, Pei and Chan, William and Medved, Jan and Suwala, George and Bates, Tony and Zhang, Amy. (2006) Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. *Proceedings of IEEE International Conference on Distributed Computing Systems(ICDCS)*

[16] Aggarwal, Vinay and Feldmann, Anja and Scheideler, Christian. (2007) Can ISPS and P2P users cooperate for improved performance?

[17] Xie, Haiyong and Yang, Y. Richard and Krishnamurthy, Arvind and Liu, Yanbin Grace and Silberschatz, Abraham. (2008) P4P: provider portal for applications. *Proceedings of ACM SIGCOMM 2008*

[18] Hong Zhuang and Yinlong Xu and Yuchong Hu and Xiaobin Lin. (2009) A Peer Selection Mechanism in P2P Networks Based on the Collaboration of ISPs and P2P Systems. *Information Science and Engineering, International Conference on*

[19] Mathis, Matthew and Semke, Jeffrey and Mahdavi, Jamshid and Ott, Teunis. (1997) The macroscopic behavior of the TCP congestion avoidance algorithm.

[20] Jonathan Ledlie and Paul Gardner and Margo Seltzer. (2007) Network coordinates in the wild. *Proceeding of USENIX NSDI*

[21] S. Halabi. (2000) Internet Routing Architectures. *Cisco Press*

[22] Bram Cohen. (2003) Incentives build robustness in BitTorrent. *In Proc. of IPTPS.*

[23] T. Mennecke DSL Broadband Providers Perform Balancing Act. *http://www.slyck.com/news.php?story=973*

[24] Bittorrent. *http://www.bittorrent.com*

[25] Vuze. *http://vuze.com*

[26] Tranmission. *http://transmissionbt.com*

[27] PlanetLab. *http://www.planet-lab.org/*

[28] Public Tracker. *http://tracker.openbittorrent.com/announce*

[29] Internet2. *http://www.internet2.edu/about/related-projects.html*

[30] iPlaneServer. *http://iplane.cs.washington.edu/*

[31] RADb. *http://www.radb.net/*

[32] SLAC. *http://www.slac.stanford.edu/comp/net/wan-mon/thru-vs-loss.html*

[33] IETF. *http://www.ietf.org/rfc/rfc3649.txt*

[34] Team Cymru. *http://www.team-cymru.org/Services/ip-to-asn.html*

[35] US-CERT. *http://www.us-cert.gov/reading_ room/DNS-recursion033006.pdf*

[36] Akamai. *http://www.akamai.com/*

[37] IPOQUE: Internet Study 2008-2009. *http://www.ipoque.com/resources/internet-studies/internet-study-2008_ 2009*

[38] TorrentFreak. *http://torrentfreak.com/bittorrent-still-dominates-global-internet-traffic-101026/*

[39] DNS. *http://www.ietf.org/rfc/rfc1034.txt*

[40] Aqua-lab *http://www.aqua-lab.org/ono/cdn_ names.properties*

[41] Comcast Lawsuit *http://torrentfreak.com/comcast-sued-over-bittorrent-traffic-interference-071114/*

[42] FTP. *http://www.faqs.org/rfcs/rfc959.html*

[43] HTTP. *http://www.w3.org/Protocols/rfc2616/rfc2616.html*

Appendices

APPENDIX A

Modification to Ono

Ono is a plugin designed for Vuze (P2P) client. Due to this nature, Ono would be able to access the information from the Vuze data structures as defined in the Vuze Plugin API's and the information flows restricted on these predefined set of entry points. In our implementation we need the edge server's redirection ratio, to identify peers within or outside the locality. Even though Ono does this selection for all peers for each swarm internal to Ono,it is not possible to differentiate between local and non-local peers by Vuze without the additional information or help. To facilitate this we added additional code in Ono and Vuze to support a periodic update request from Vuze regarding the ratios collected for all the peers irrespective of the swarm by Ono.

For our implementation, we had two options, either to add more entry points in the Vuze Plugin API so that we would be able to have access to Ono's internal data structures, else use Vuze plugin's IPC API to communicate with Ono. Irrespective of whichever option we select, it involved modifying the Ono sourecode, since the Ono plugin did not allow outside reference to the internal data structures that stored the edge server redirection ratios.

Since this specific need to interact with Ono is not helpful to others, we went ahead with the second option to meet our specific needs. We used the Vuze's IPC mechanism to obtain the reference to the Ono Plugin instance, which is later used to call certain new methods mentioned below, added as part of the Ono sourcecode modifications.

- ipc_getAllRatios()
  Returns an associative array for all peers ,with peer ip has the key and the list of edge server ratios as the value.

- ipc_getMyRatios()
  Returns an associative array for the calling peer, with a list of its edge server ratios else null.

- ipc_getGoodCDNNames()
  Returns a list of CDN Names obtained from the aqualab(add reference to the URL) site.

- ipc_getRatio(String ip)

  Returns as associative array for the given IP, a list of its edge server ratios else null.

- ipc_getCosineSim(String peerIP)

  Returns the cosine similarity ratio as defined in the paper(add reference) for the given peer IP else null.

APPENDIX B

Modifications to a BT client

The contracted peers needs to be treated differently, compared to other neighbours because of the prevailing contracts between them. Hence we have modified Vuze, a BT client in the following ways to be conducive towards contracts.

1. Vuze maintains an active cache of all disconnected peers,to allow fast reconnect within a defined time span. Thus in case of contracted peers, even the contracts are stored in the cache until either it is removed from the cache due to ageing, which would occur if the contracted peer never reconnects again or times out or reconnect occurs. In case of ageing the contract is terminated and the pieces become available for future contracts.

2. To allow altruism, low performing peers are disconnected on regular intervals to allow new peers within the neighbour set. We exclude the contracted peer from this list since data rate is a property used to identify low performing peers.

3. Maintain a persistent cache to store AS hop count information for connected peers to avoid repeated requests to either the iPlane server or Team cymru database on reconnects or across download.