Understand the Similarity of Internet Service Providers via Peer-to-Peer User Interest Analysis

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Prateek Joshi

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Haiyang Wang

June 2019

Acknowledgements

First and foremost, I would like to thank my advisor, Dr. Haiyang Wang, for giving me the opportunity to work on this thesis. He has always been forthcoming in helping me out of tight spots during my research. Additionally, I would also like to thank Dr. Imran Hayee and Dr. Andrew Sutton for agreeing to be part of the review committee and taking the time to go over my thesis. Finally, I would like to thank all my friends and professors who have helped me learn and grow, inside and outside the classroom.

Dedication

I dedicate this research to my parents Vandna and Vinod Joshi, and my sister Prekshya for their perpetual love and support. Also, to the many friends that I have made over the past two years during my stay in Duluth.

Abstract

Internet traffic continues to exhibit exponential growth in the past few years. This forces Internet service providers(ISPs) to continuously invest in infrastructure upgrades and deploy traffic management techniques, such as caching and locality, to fulfill the increasing user demand. To help ISPs better manage their infrastructures, it is important to compare and understand the similarity of their user interests. However, such a comparison is challenging because the ISP data is hard to obtain, not to mention the related modeling and analysis issues.

In this thesis, we aim to understand the ISP similarity through an extensive analysis of Peer-to-Peer(P2P) user interest. To collect the P2P dataset, we develop a tool to automatically download BitTorrent's meta-info(torrent) files on the Internet. This tool also helps us to collect important peer and content information in these BitTorrent swarms without uploading any copyrighted files. As a result, we successfully obtained 16,697 active peers from 1,721 torrents in 1,097 unique Autonomous Systems(ASes). After that, we adopt the classic statistical and clustering approaches to compare their different user interests. Our research for the first time shows the existence of cloud users in such real-world content distribution systems as BitTorrent. The model analysis further indicates that we can adopt similar traffic management approaches (e.g., caching similar contents) across geographically closer ASes.

Contents

List of Tables

List of Figures

1 Introduction

The Internet has become an indispensable part of our daily lives. It's so ubiquitous that as of March 31, 2019, more than half the earth's population is using it. [11] As Internet penetration improves, internet traffic is going to substantially increase with estimates of the annual global IP traffic to reach 4.8 *zettabytes* (ZB) per year by 2022, or 396 *exabytes* (EB) per month. For reference, the global IP traffic was at 1.5 ZB per year or 122 EB per month in the year 2017. [5] Management of this burgeoning traffic has become quite challenging for Autonomous Systems (AS) like Internet service providers (ISP). In order to serve user requests, ASes need to route packets within and outside their network. When sending packets outside their network, ISPs incur a transit cost. [10] Also since the resources are outside the network there is also a routing overhead which causes a performance hit.

When we break down the constituents of network traffic on the Internet, the consumer IP traffic between two different ASes forms a major part. Investigating further, video and file - sharing form a significant portion of this inter-AS traffic with both constituting 96 EB per year and 672 EB per year respectively. File sharing is mostly Peer-to-Peer (P2P) traffic from systems like BitTorrent. Also, many video streaming services implement a P2P infrastructure. This combined traffic is going to increase to 2.96 ZB per year by 2022. [5] To better manage this exploding P2P traffic, it is paramount to understand the user interest within Content and Internet Service Providers. By modeling the user preferences these autonomous systems can implement caching strategies for common resources and greatly reduce cross-network traffic and also improve performance. Also, networks with highly similar user interest pattern can potentially share their cache infrastructure. The other benefit of studying P2P users is the granular details that are available. P2P users can be studied in a lot of detail as the metadata of the files being shared and the peers participating in sharing those files can be measured.

On the corollary, studying enterprise infrastructure like cloud systems is difficult due to the propriety software running on the distributed hardware which makes the passive study of the users preferences difficult. What makes it even more difficult is that most of the major players are shielded by service level agreements. So big corporations like Google, Microsoft and Amazon do not give out their network details to the public. [20] [8] Because of these restrictions, most of the previous studies have used indirect methods to study and analyze user preferences. For example, studying resource usage behavior or log data to study mobile user interest [1] [22] This also makes our study unique and different from previous studies as we try to investigate the patterns in P2P user interests by creating a web crawler that collects a massive amount of data by participating in peer swarms sharing torrent files (without actually downloading the files). Having a large data set, coupled with the granular control over measurement allowed us to make interesting observations in user preferences among P2P users.

With this study, we set out to investigate whether there is any similarity pattern between the interest of users in different Autonomous Systems. What we stumbled upon was the presence of a large number of cloud users who have shown a largely similar interest in content across ASes. Also, further analyzing the data leads us to interesting insights into how geographic proximity might be a big factor governing content preferences. In this study, we also find several other observations and try to breakdown and understand the overall interest of users in P2P users from around the world.

The structure of this thesis is as follows: Section 2 contains all the background knowledge which will be required to deeply understand this research and also previous research in the area related to this research. Section 3 contains the implementation details of how the data was captured and what overall observations we made. Section 4 contains insights that we gained on the collected data as well as the statistical model that we created to put our observations into perspective. Section 5 summarizes our findings and concludes this thesis.

2 Background

Distributed Computing's recent growth has contributed significantly to the increase in internet traffic with Cloud Computing and Peer-to-Peer (P2P) computing at its forefront. Armed with the benefits of ease of use and scaling up of resources on demand, Cloud Computing has helped smaller businesses to compete with the behemoths of the web world and establish themselves. The cost benefits also have been the major factor in their popularity. The cost is scaled commensurately with the amount of service used. Another distributed computing paradigm, P2P networking, provides even further cost benefits albeit lesser guaranteed Quality of Service. In this section, we go into details behind the terms used in this thesis.

2.1 ISP and Traffic Management

The Internet can be visualized as a wire. Two computers connected to this wire can communicate. A Server is a special type of computer that can connect directly to the Internet and hosts web file in its hard drives (or other kinds of storage). A Client is a computer that is not directly connected to the Internet. Instead, it gains access to the Internet via an Internet Service Provider (ISP) which manages several Clients. At each junction, the Internet is connected via routers that manage the traffic between computers. Each device connected to the Internet has a unique IP address. When a Client requests a resource (like a website) that is in a Server, the information is transmitted between the Client and Server via packets. Packets are small chunks of

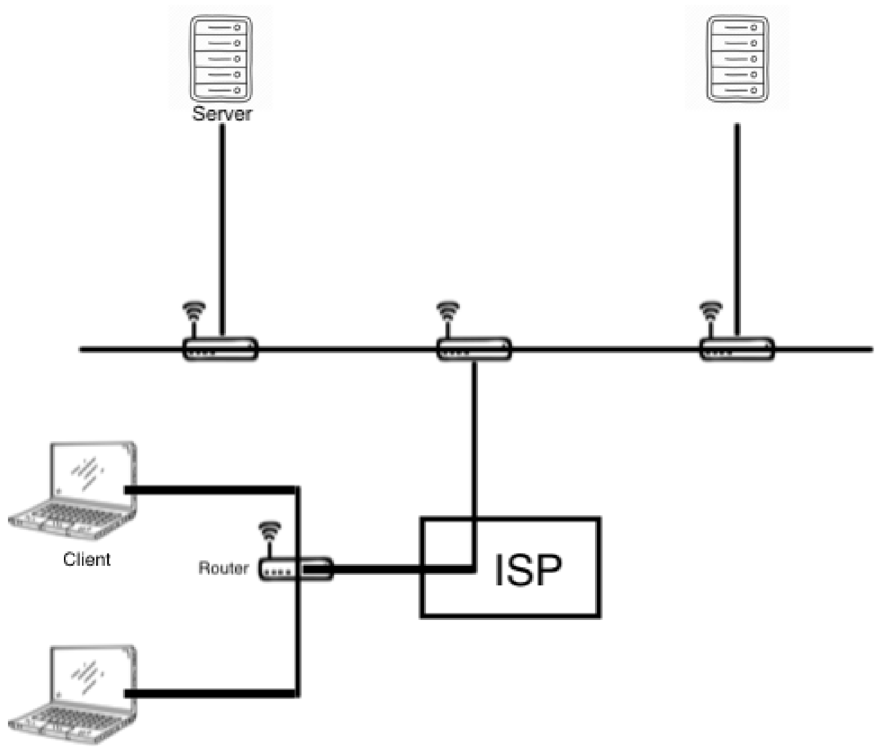

Figure 2.1: A figure showing how the Internet service provider fits into the global Internet infrastructure.

information that together form the complete request/response. These packets can follow the path of least congestion to reach their destination. The receiving computer reorganizes these packets to infer the message. These packets are what constitutes internet traffic.

2.1.1 Autonomous Systems

The Internet is often incorrectly thought of as one big network of different computers. In reality, the Internet is a network of computer networks. These independent

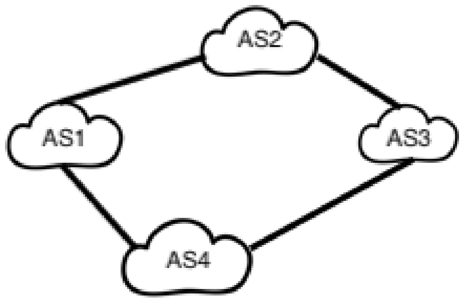

Figure 2.2: A figure showing how Autonomous Systems together form a network and how Border Gateway Protocol works.

networks are called Autonomous Systems (AS). These ASes can be education institutes like the University of Minnesota or large level 3 ISPs like Comcast or enterprise organizations like Amazon. These ASes manage the infrastructure within their network and co-operate among each other to transmit information. Inter-AS traffic is administered by a protocol called the Border Gateway Protocol.

2.1.2 Border Gateway Protocol

When a Client requests resources from outside its own AS's network, the packets that are transmitted between ASes are governed by Border Gateway Protocol (BGP). For example from Figure 2.2 if $AS1$ wants to send/receive packets to/from $AS3$, it needs to know the path to reach the other AS. In BGP, each AS has a router that controls traffic going outside its own AS. Routers of all the ASes in the Internet broadcast their neighbour list to all of their neighbors. These neighbouring ASes use the transmitted messages from all other ASes to build their routing table which contains information to reach every other AS on the Internet. The routing table also stores information like how far the other ASes are and other metadata.

2.2 Distributed Systems

This a technique wherein several computers work together to solve the same problem. These group of computers communicate among themselves by passing messages over the network. For a third party outside this distributed system, the group of computers would be abstracted as one computer.[13]

2.2.1 Cloud Computing

Cloud Computing is a type of utility computing (with the other one being grid computing). As the name suggests it provides computing services based on utility. A big spectrum of enterprises are moving their applications to the cloud because of the better performance and reliability. [15] The infrastructure and all its quirks are managed by the service provider and you only have to pay for the amount of service you use. Cloud Computing harnesses the power of distributed computing. The Cloud is a group of computing resources which are remotely located and provision a subset of these resources to users who can be anywhere in the world as long as they are connected to the Internet. This provisioning is given with practically no initial infrastructure investment from the user. Additional resources can be provided on demand with the cloud having virtually ”infinite” resources.[17] There are three popular types of services depending upon the level of control provided, namely: software as a service, platform as a service and infrastructure as a service.

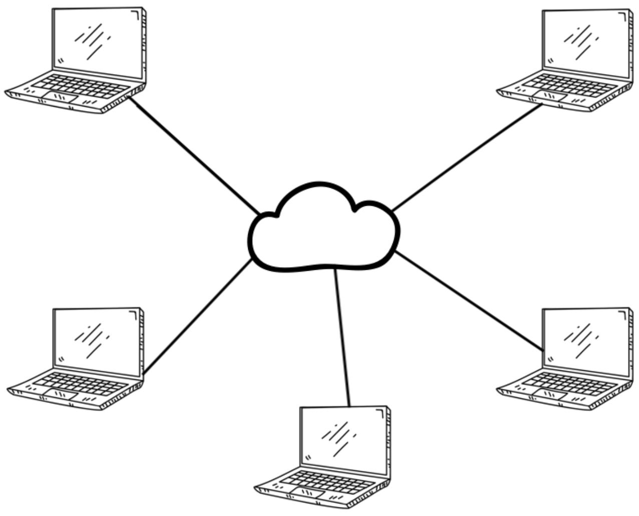

Figure 2.3: A figure showing how the Client Server model works in Cloud Computing.

2.2.2 Peer-to-Peer Computing

Earlier, the standard for content distribution was dominated by the centralized paradigm. The Client-Server model provided a simplicity that was easy to implement and research. Peer-to-Peer (P2P) computing solved two major issues faced by this standard:

- Single point of failure i.e. the Server
- The scalability issue, wherein the increasing number of Clients choked the Server of bandwidth.

P2P system have become increasingly popular making up to about 30 percent of the Internet's traffic.[3] P2P is another paradigm of the distributed computing world. This is a completely decentralized approach with each Client also acting

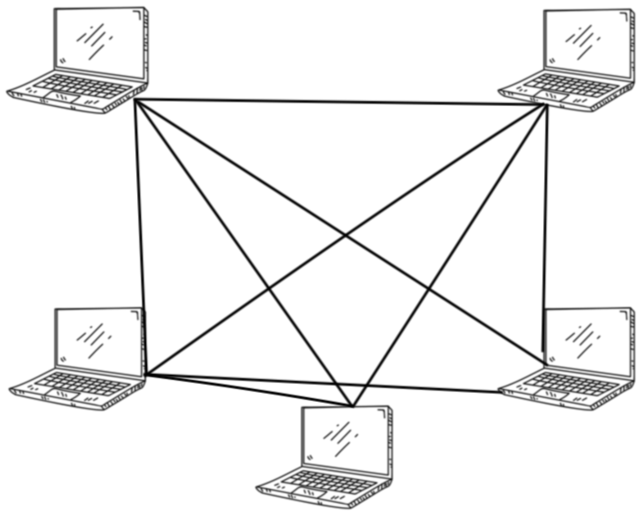

Figure 2.4: A figure showing how the Peer-to-Peer architecture works.

as a Server. All Clients work together to form a swarm and utilize each other's computing resources. In contrast to Cloud Computing, this approach utilizes the upload capacities of Clients. This greatly reduces the operation cost as the Clients use their own infrastructure to distribute content.[18] The lack of centralization also provides better scalability, as a new Client can come in and out of swarm without incurring any cost.[13]

Even though it may seem that the cost benefits of P2P computing should make it the de-facto choice of the IT industry but sadly it lacks a business model[14]. The biggest issue is reliability and availability of services, as P2P works on the best effort philosophy. The cost-effectiveness of the Peer-to-Peer system and the reliability of cloud service, both can be harnessed in a hybrid system. These hybrid systems can be used for many applications: storage and backup systems, music streaming, and potentially online gaming and video streaming.

BitTorrent

BitTorrent is a widely used P2P protocol. It uses the ”tit for tat” exchange scheme. The content that has to be distributed is divided into many different parts called fragments. All the Clients involved in the distribution of a content are called peers. The peers are two types: downloaders and seeders. Downloaders are the peers that do not have the full copy of the content and have some (or none) of the fragments. They download the other fragments from the other peers. Seeders have the complete copy of the content and only participate to distribute the content. Together the downloaders and the seeders form a swarm. A tracker is a file that keeps track of every peer in the swarm. When a downloader or seeder needs to figure out which peers are available in the swarm it can send a request to the tracker file to get the IP and port address of these other members. A text file called the torrent file is distributed on the Internet that contains the following:

- The count of the fragments and a checksum for each fragment. The checksum is an SHA-1 hash and is used to verify the integrity of the downloaded fragment.
- It keeps track of the link at which the tracker file is published.

Each peer downloads/seeds fragments from/to different members of the swarm. They query the tracker file to get the location of other members. Each participating peer that is either downloading, uploading or both, keeps track of the peers it is involved with. Also, the peer keeps track of statistics like download speed and upload speed. In a tit for tat scheme, the peer chooses the best seeders based on their uploading speeds thus increasing its own download speed. Also, as a side effect, the peer would lower its interaction with a slow peer. This phenomenon is called choking. An augmented version of BitTorrent solves the NAT problem in a Peer-to-

Peer system. Modifications in the Clients and trackers help in better distribution of data in a NAT environment. The improvements are:

- If there are seeders and downloaders within the same NAT network then they may contact each other instead of contacting a peer of a different group IP.
- This interaction with the same network is better because NAT systems are connected to LAN which usually has way better bandwidth than a WAN link.
- This leads to a lesser bottleneck in the WAN network.

Minimum Distribution Time

A system that follows the aim of having minimum distribution time has peers in its swarm that collectively work towards reducing the overall time for the content to be distributed completely to each and every Client in the participating swarm.

Bulk Synchronous Systems

As the name suggests, a system in which the all the member peers are synchronized to perform actions at the same time. Such a system is typically required for achieving the object of MDT.

Group Tree Strategy

A coordinated swarming strategy that implements a practical version of the optimal fluid MDT construction in a bulk-synchronous system. In it, any Client does not need to have more than x connections.

The group tree technique has two phases: Initially, the provider breaks down the content to be distributed into x fragments. Here, x is the maximum out-degree of a peer.

- In this phase, each Client is separated into trees. This division is done on the basis of upload speeds. Each Client in a tree would have similar upload capabilities. Different fragments are given to different trees proportional to their upload speeds. A bigger fragment would be given to a tree that has Clients of higher upload speeds. To distribute these fragments, they are sent as chunks to root node of these trees and they are forwarded to other nodes in a pipeline fashion.

- In this phase, a clique is formed by taking one peer from each and every tree. Every member of a clique has a different fragment of the content. Using a P2P strategy the members exchange data to get the complete file.

Overlay Multicast

A multicast is an approach that is somewhere in between a cloud and a P2P approach. In it, the content is distributed completely to another participating node and then it is passed on like a baton. An overlay multicast is fashioned like a network of tree topologies used together in an efficient design to distribute content. This approach uses the Client's upload resources more than than cloud but less than P2P.[3]

2.3 Clustering Algorithms

Clustering is a form of unsupervised machine learning in which there is no training data. The aim is to find an underlying structure in the data-set. Clustering divides a data-set into groups wherein data-points within a group are more similar to each other than to data-points in another group. This similarity can be based on the various different types of distance metric.

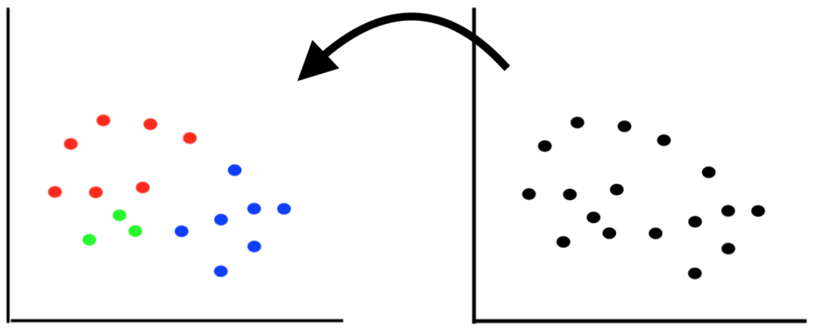

Figure 2.5: A figure showing how clustering works.

In this thesis, we have a large data-set of users participating in Peer-to-Peer file sharing. Our goal is to find an underlying similarity pattern within these users. Clustering is the perfect tool to go about finding this.

There are broadly two overall types of clustering, namely:

Hard Clustering

In hard clustering, each data-point falls completely into one group. Like it is shown in Figure 2.5, every data-point can be completely put into one group and there is a "hard" distinction between the groups. We are interested in this type of clustering for this thesis.

Soft Clustering

In soft clustering, each data-point does not completely fall into one group. Each data-point falls in every group with some probability or likelihood.

2.3.1 Categories

There are several algorithms for clustering. Essentially, these are different ways of defining the distance metric depending upon the situation and preference. Here we discuss relevant algorithms that we considered during our research.

Connectivity Models

This algorithm is based on the approach of distance metric being calculated directly from one data-point to another. Different types of distance metric can be chosen. There are two main ideas that are used in order to solve this. First is to treat every data-point as an individual cluster and then go about forming bigger clusters by merging the closest clusters. The other approach is to form one big cluster including all data-points and then go about diving it into more granular clusters. In this thesis, we ended up finally using an algorithm of this category called agglomerative hierarchical clustering.

Centroid Models

In this algorithm, distance is measured from the centroid of a cluster. This is different from the previous approach where distances were from individual data-points. A popular algorithm of this category is K means. The number of centroids k has to be specified in the beginning.

Distribution Models

In this algorithm, the main idea is that all data-points in a particular cluster belong to the same distribution. This probability distribution could be Gaussian or Normal etc. A common algorithm used in this category is the expectation maximization algorithm. The distribution used in this approach is a multivariate normal distribution.

Density Models

This algorithm creates clusters based on the density of data regions. The higher the density, the higher the chance these data-points fall in the same cluster. A popular example of this algorithm is DBSCAN.

2.3.2 K Means Clustering

In this approach, k clusters are randomly assigned. Next, the centroid of these clusters is calculated and then data-points are re-assigned forming new groups. This is iteratively continued until there is no change in groups. This was the first approach used in our thesis. But this does not work for high dimensional data like ours.

2.3.3 Hierarchical Clustering

In this approach the aim to create a hierarchy of clusters. These clusters can be formed bottom up or top down. As shown in Figure 2.6, the output of clusters can be visualized in a dendrogram. This dendrogram can be cut at different heights to get the required number of clusters.

2.4 Related Works

There have been many studies previously that are about measurement and analysis of Internet user interests in different types of internet traffic. In this thesis we mainly looked into two types of previous research, those that modelled user data from internet

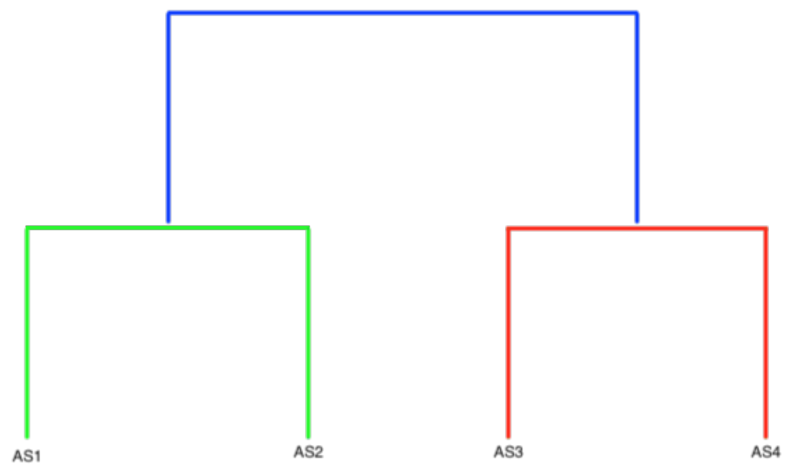

Figure 2.6: A figure showing how hierarchical clustering works.

traffic and those that used user models to implement cache strategies.

Man et al [19] measured the usage patterns in mobile users using large data-sets of user log data. They built a user interest model for personalised recommendation systems. Also, they made use of hierarchical modeling to model their data. Another study on mobile data-sets was by *Xia et al* [24]. They used Geo-tag information from mobile traffic to study mobile user interests. They built *GeoEcho*, a mobile traffic analysis system that clusters mobile data traffic to infer POIs (places of interest). *Qin et al* [21], Used *k mediods* clustering approach for building user interests model to speed up P2P document file sharing.

Guo et al [9], made distributed cache for improving mobile traffic performance. They came up with a novel co-operative cache strategy using users smart devices. This also harnessed user interest model and shared their cached contents using a D2D model. *Lu et al* [16], used user modeling to improve search performance in federated text search in Peer-to-Peer systems. Using user's long term interests based on past queries they were able to reduce the search radius. Also, they had a fallback measure for ad-hoc or new queries. *Chen et al* [4], came up with a way to improve search performance in unstructured P2P networks through exploiting users common interest patterns. These patterns were captured inside a probability theoretic framework called User Interest Model (UIM). This also included a smart way of updating routing tables. *Yu et al* [25], used a modified Poisson distribution to understand the user behaviors in a large scale video on demand systems. *Dernbach et al* [6], built a cache strategy based on user interest using content selection from video streaming services.

Our thesis differs from the previous work because we have for the first time attempted to model user interests in P2P traffic using hierarchical clustering modelling and also for the first time found presence of cloud users in BitTorrent traffic. [12] [7] [2]

3 Measurement and Analysis

The experiment in this thesis is divided into two parts, measurement and observations. In the measurement phase, we collect BitTorrent users and torrent data from around the world using a tool that we developed called *Torrent Scrapper* and in the observation part, we study the collected data.

3.1 Measurement Configuration

Torrent Scrapper was developed using *Python* 2.7. To develop this tool, We used the Python bindings for *Libtorrent* and the library *BeautifulSoup*4. *Libtorrent* is a complete implementation of BitTorrent protocol in C++ with binding available in Python. This thesis makes use of the Python binding to participate in peer swarms. *Libtorrent* enables us to use the BitTorrent protocol programmatically. *BeautifulSoup*4 is used to parse the HTML from web-pages in order to extract torrent magnet links.

3.1.1 Script Algorithm

Torrent Scrapper works as follows:

1. Iterate over every web-page listed in the configuration *JSON*.
2. Parse each web-page and extract every magnet link within those web-pages.
3. Create the configured number of child processes.

4. In parallel, convert all the magnet links into torrent files.
5. Run all these torrent files until a connection is established to the peers sharing the torrent file.
6. Extract the following information of each torrent:
   - *creation_date*: When the torrent was created.
   - *announce*: The request sent to the trackers.
   - *files*: Total number of files in the torrent.
   - *piece_length*: Number of BitTorrent pieces.
   - *name*: Name of the Torrent.
   - *announce_list*: All the trackers to which a request was sent.
   - *type*: The category the torrent belongs to.
   - *source*: Source URL for the torrent.
7. For every peer in that torrent, download the following information:
   - *down_speed*: The download speed for the peer.
   - *ip*: The IP address of the peer.
   - *up_speed*: The upload speed for the peer..
   - *asn*: A unique number called the Autonomous System Number.
   - *asn_cidr*: The assigned ASN CIDR.
   - $asn\_country_code$: The assigned ASN country code.
   - *asn_date*: ASN Allocation date.
   - *asn_description*: Complete name and description of the ASN.

   - *asn_registry*: The assigned ASN registry.

   - *nets*: List of network dictionaries.

8. Close each connection after the data has been gathered and stored in the database or time limit has been exceeded.

9. Delete any residual data of the torrent.

3.1.2 Script Configuration

The tool makes sure that no content data has been uploaded. Only peer meta-data is being downloaded. In this way we avoid any copyright infringement. The websites that were scrapped include: *katcr.co*, *limetorrents.info*, *piratebay.org*. The tool takes in several configuration parameters. Following is the description for the key settings:

- *Process Count*: Number of sub-processes to run. These sub-processes parallelize the workload of converting magnet links to .torrent files.

- *Time out Sec*: Time out in seconds for the sub-process mentioned above.

- *Headers*: HTTP headers used by the request library when making the request for the page to scape.

- *CSV Name*: Name of the ouput csv.

- *Default NA value*: Placeholder for null values in the output csv.

- *URL list*: List of URLs to scrape.

Given below is the configuration *JSON* used in the tool during our experiments.

```

{
 "process_count": 30,
 "timeout_sec": 30,
 "headers": {
   "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
       AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181
       Safari/537.36",
   "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,
       image/webp,image/apng,*/*;q=0.8",
   "accept-charset": "ISO-8859-1,utf-8;q=0.7,*;q=0.3",
   "accept-encoding": "gzip, deflate, br",
   "accept-language": "en-GB,en-US;q=0.9,en;q=0.8",
   "cache-control": "max-age=0",
   "connection": "keep-alive",
   "cookie": "KATSSESS_ID70AT=v9mf684tkqt96en13q4p540m75govqn0",
   "host": "katcr.co",
   "upgrade-insecure-requests": "1"
 },
 "csv_name": "kat_torrent_data.csv",
 "default_na_value": "",
 "url_list": [
   {
     "link": "https://katcr.co/category/tv/page/",
     "type": "tv",
     "pages": 101,
     "source": "kat.ph",
     "multi_page": true
   },
   {
     "link": "https://katcr.co/category/movies/page/",
     "type": "movies",
     "pages": 101,
     "source": "kat.ph",
     "multi_page": true
   },
   {
     "link": "https://katcr.co/category/games/page/",
     "type": "games",
     "pages": 101,
     "source": "kat.ph",
     "multi_page": true
   },
   {
     "link": "https://katcr.co/category/music/page/",
     "type": "music",
     "pages": 101,
     "source": "kat.ph",
     "multi_page": true
   },
    {
      "link": "https://www.limetorrents.info/browse-torrents/Movies/date
          /1/",
      "type": "movies",
      "multi_page": false,
      "source": "limetorrents.cc"
    },
    {
      "link": "https://www.limetorrents.info/browse-torrents/Movies/date
          /1/",
      "type": "movies",
      "multi_page": false,
      "source": "limetorrents.cc"
    },
    {
      "link": "https://www.thepiratebay.org/top/400",
      "type": "games",
      "multi_page": false,
      "source": "piratebay.org"
    },
    {
      "link": "https://www.thepiratebay.org/top/100",
      "type": "music",
      "multi_page": false,
      "source": "piratebay.org"
    },
    {
      "link": "https://www.thepiratebay.org/top/200",
      "type": "movies",
      "multi_page": false,
      "source": "piratebay.org"
    },
    {
      "link": "https://www.thepiratebay.org/top/208",
      "type": "tv",
      "multi_page": false,
      "source": "piratebay.org"
    }
  ]
}
```

3.1.3 Source Code

Main Function

Below you can find the implementation of the main function. It contains the logic for iterating over the URLs and calling the *scrape_page*() function.

```

with open(CONFIG['csv_name'], 'wb') as myfile:
  csv_writer = csv.writer(myfile, quoting=csv.QUOTE_ALL)
  csv_writer.writerow(
    [
      'creation date',
      'announce',
      'files',
      'piece length',
      'name',
      'announce-list',
      'type',
      'magnet link',
      'source'
    ]
  )

  for url in CONFIG['url_list']:
    if url['multi_page']:
      for i in range(1, url['pages']):
        url['link'] = "%s%s" %(url['link'], i) if i > 1 else url['link
          ']
        scrape_page(url, csv_writer)
    else:
      scrape_page(url, csv_writer)
```

Scrape Page

Following is the logic for the *scrape_page*() function which parses the provided URL and then extracts every magnet link in a web-page. Next, it creates multiple child processes. In each child process it converts the parsed magnet link into a torrent file and runs the torrent by calling the *convert_magnet_to_torrent*() function until all the required meta-data is extracted and stored in a database. If it takes too long to get the meta-data, it terminates that torrent and moves to the next one.

```

def scrape_page(url, csv_writer):
  global processes_completed
  response = requests.get(url['link'], headers=CONFIG['headers'])

  if response.status_code != 200:
    print("request denied")
    return

  soup = BeautifulSoup(response.text, 'html.parser')
  links = [link for link in soup.find_all('a') if link['href'].
      startswith('magnet')]

  for link_index in range(0, len(links), CONFIG['process_count']):
    chunk = links[link_index:link_index+CONFIG['process_count']]
    pool = Pool(processes=CONFIG['process_count'])
    chunk_result = pool.map(convert_magnet_to_torrent, chunk)
    chunk_result = [chunk for chunk in chunk_result if chunk != None]

    for idx, result in enumerate(chunk_result):
      if not isinstance(result, dict): continue
      if 'info' not in result: result['info'] = {}
      csv_writer.writerow(
        [
          check_key(result, 'creation date'),
          check_key(result, 'announce'),
          check_key(result['info'], 'files'),
          check_key(result['info'], 'piece length',),
          check_key(result['info'], 'name'),
          check_key(result, 'announce-list'),
          check_key(url, 'type'),
          links[link_index+idx]['href'].encode('ascii', 'ignore'),
          check_key(url, 'source'),
        ]
      )

    processes_completed += CONFIG['process_count']
    print("processes completed: %s" %(processes_completed))
```

Convert Magnet To Torrent

Below is the code for *convert_magnet_to_torrent*() is given. It has the appropriate exception handling in case of timeout or incomplete data while converting magnet to torrent files and extracting peer meta data.

```python

def convert_magnet_to_torrent(link_tag):
  torrent_file_name = "torrent%s" %(os.getpid())
  print("started subprocess: %s" %(torrent_file_name))

  proc = Popen(['python', 'Magnet_To_Torrent2.py', '-m', link_tag['href'
      ].encode('ascii', 'ignore'), '-o', torrent_file_name], stdout=PIPE
      , preexec_fn=os.setsid)
  timer = Timer(CONFIG['timeout_sec'], kill_and_set_flag, [proc,
      torrent_file_name])
  result = []
  try:
    try:
      timer.start()
      stdout, stderr = proc.communicate()
      rawdata = open(torrent_file_name).read()
      print("completed subprocess: %s" %(torrent_file_name))
      result = bencode.bdecode(rawdata)
    finally:
      timer.cancel()
      Popen(['rm', torrent_file_name], stdout=PIPE, stderr=PIPE)
      if torrent_file_name in process_status and process_status[
          torrent_file_name] == 1:
        del process_status[torrent_file_name]
        return None
      else:
        return result
  except Exception as ex:
    template = "An exception of type {0} occurred. Arguments:\n{1!r}"
    message = template.format(type(ex).__name__, ex.args)
    print message
    return None
```

3.1.4 Data Processing Code

The data that is captured by *Torrent Scrapper* contains torrent and peer swarm information. The key insight that we wish to uncover is about ASes. In order to do that we have to process the data and order it by ASes. We developed the following script to do that.

```python

asn_stats = {}
for peer in complete_peer_data_2:
  asn = peer['whois_stats']['asn']
  if asn not in asn_stats:
    asn_stats[asn] = {
      'total_down_speed': peer['down_speed'],
      'total_up_speed': peer['up_speed'],
      'count': 1,
      'desc': peer['whois_stats']['asn_description'],
      'asn_cidr': peer['whois_stats']['asn_cidr'],
      'asn_registry': peer['whois_stats']['asn_registry'],
      'movies': 0,
      'tv': 0,
      'games': 0,
      'music': 0
    }
    asn_stats[asn][peer['type']] += 1
  else:
    asn_stats[asn]['total_down_speed'] += peer['down_speed']
    asn_stats[asn]['total_up_speed'] += peer['up_speed']
    asn_stats[asn]['count'] += 1
    asn_stats[asn][peer['type']] += 1
  for asn, stats in asn_stats.items():
    asn_stats[asn]['avg_down_speed'] = stats['total_down_speed'] / float(stats['count'])
    asn_stats[asn]['avg_up_speed'] = stats['total_up_speed'] / float(stats['count'])

max_avg_down = { 'asn': '', 'speed': 0 }
max_avg_up = { 'asn': '', 'speed': 0 }
max_count = { 'asn': '', 'count': 0 }
count_dist = []
avg_down_speeds = []
avg_up_speeds = []
max_c = -1
max_down = -1
max_up = -1
for asn, stats in asn_stats.items():
  avg_down_speeds.append(stats['avg_down_speed'])
  avg_up_speeds.append(stats['avg_up_speed'])
  count_dist.append(stats['count'])
  if stats['avg_down_speed'] > max_down:
    max_avg_down['asn'] = asn
    max_avg_down['speed'] = stats['avg_down_speed']
    max_down = stats['avg_down_speed']
  if stats['avg_up_speed'] > max_up:
    max_avg_up['asn'] = asn
    max_avg_up['speed'] = stats['avg_up_speed']
    max_up = stats['avg_up_speed']
  if stats['count'] > max_c:
    max_count['asn'] = asn
    max_count['count'] = stats['count']
    max_c = stats['count']
```

Metric	Total Count
Torrents	1721
Peers	16697
AS	1097
PeerswithoutASN	665

Table 3.1: Overall statistics of the collected data.

The grouped data is stored in a dictionary called *asn_stats*. This dictionary contains data for 1097 unique ASes. An example of an entry in *asn_stats* is given below. The key is the unique AS id captured via the *whois* UNIX command. The properties of the object stored as the value are explained in section 3.1.1.

```
{

    u'133384':{
      'asn_registry':u'apnic',
      'asn_cidr':u'103.231.94.0/24',
      'avg_up_speed':4.0,
      'desc':u'GTCL-AS-AP 5BB Broadband, MM',
      'count':1,
      'tv':0,
      'total_down_speed':0,
      'total_up_speed':4,
      'movies':0,
      'games':1,
      'avg_down_speed':0.0,
      'music':0
    },

}
```

3.2 Measurement Observations

In this section we get into the details of the collected data.

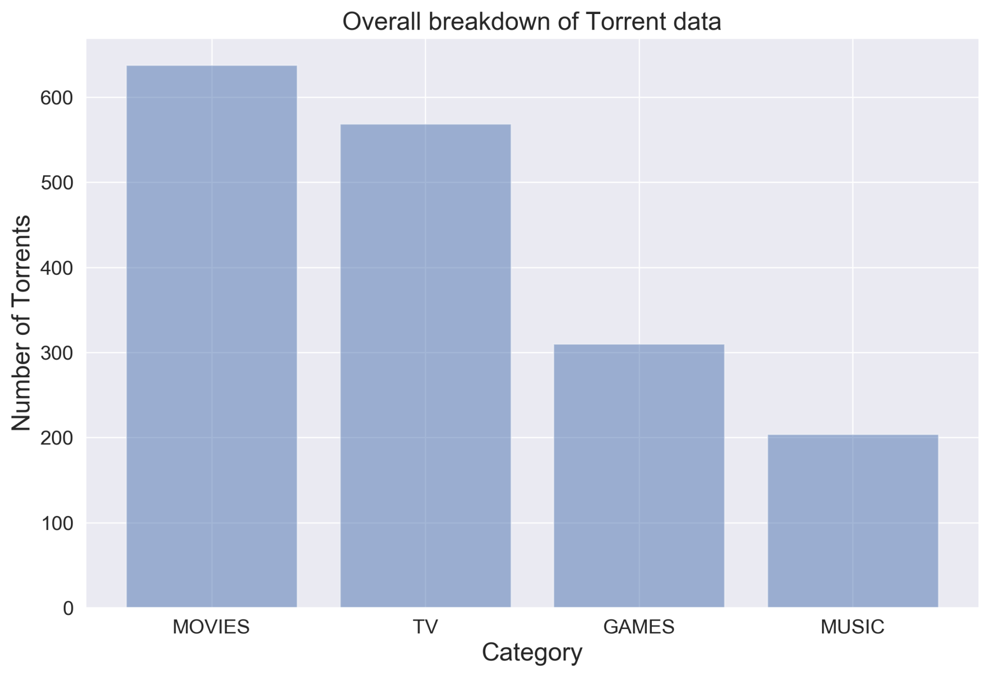

Figure 3.1: Torrent data breakdown according to category.

3.2.1 Overall Metrics

In total around 2800 torrents were downloaded. Total number of torrents downloaded that have peer information are 1721. There are 16697 peers sharing these 1721 torrents. All these peers fall under 1097 unique ASes. There are 665 peers for which AS information could not be gathered. Table 3.1

3.2.2 Torrent Categories

Of all the 1721 total torrents sampled, 638 of them are of the category movies, 569 are TV series, 310 are Games and 204 are music. As shown in Figure 3.1.

Metric	MAX	MIN	MEAN	MEDIAN
Torrent Size (MB)	63397.27	3.13	2782.81	1021.41
Download Speed (KB/s)	16462.64	0	17.738	3
Upload Speed (KB/s)	520.37	0	1.6	0.05
Peer Swarm Size	102	1	7	6
Number of Pieces	10502	11	1334.33	641
Number of Files	769	1	7.095	3
Number of Trackers	5	5	5	5

Table 3.2: Detailed breakdown of peer statistics.

3.2.3 Detailed Metrics

Across all peers statistics collected, the detailed breakdown of download speed, upload speed is given in Table 3.2. The largest torrent is of size 63 GB Game and the smallest torrent is of size 3.13 MB Music file. The highest download speed observed is 16.5 $MBps$. This is because *Torrent Scapper* breaks connection to a peer very quickly in order to only download meta-data and not content. The upload speed measured on average is 1.6 $KBps$. This upload speed is expected to be low compared to download speeds. The large peer swarm encountered contains 102 peers. On average a torrent file contains 7 files and 1334 pieces. Trackers interestingly remained a constant 5 ac-cross statistics.

3.2.4 Heat Map of the world showing Peer Origins

The sampled peers are from 157 countries around the world but mostly concentrated in USA. This is likely because we ran our tool in USA and the BitTorrent protocol tends to connect to peers that have good upload speeds because of the $tit\ for\ tat$ policy. As the tool was run for an extended period, we were able to sample peers from he world over. Netherlands was another country that had high peer count. The heat-map of the geographical location is given in Figure 3.2.

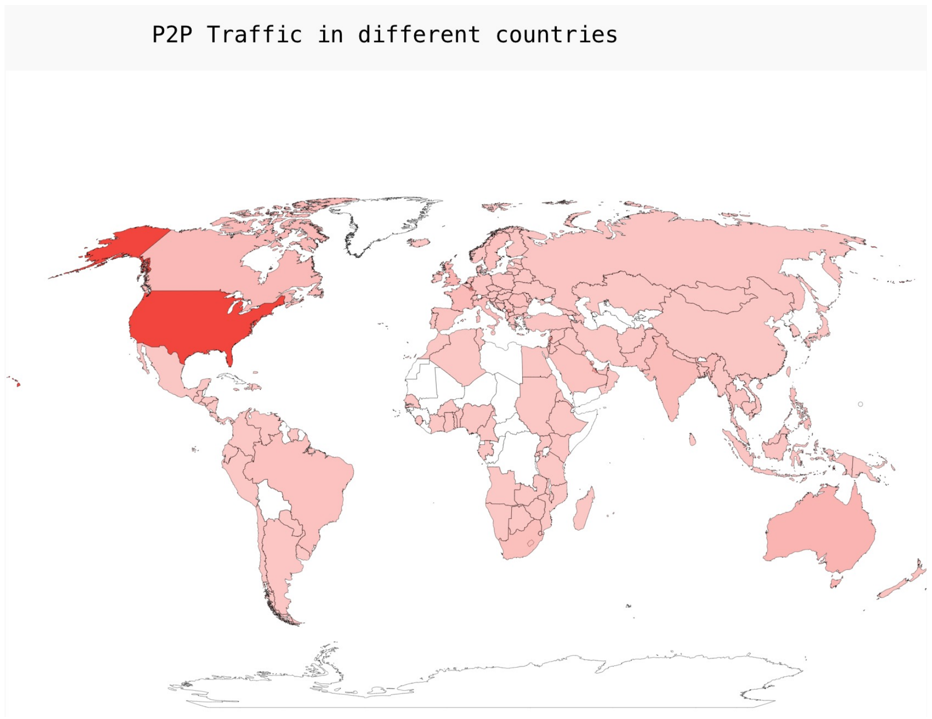

Figure 3.2: Heat Map of the world showing the geographical location of the peers captured.

3.2.5 Ranking Torrents by Content Size

As shown in Figure 3.3, when torrents are ranked by content size, the torrent size exponentially decreases. Most torrents are below 10 GB in size. The average size hovers between 2 to 3 GB because of the majority torrents being movies and TV shows. The larger torrents are usually games.

3.2.6 Category breakdown by Content Size

Figure 3.4 shows the category breakdown when ranking torrents by content size. The largest contents are dominated by games. Then by TV shows, then movies and then the smallest content sizes are usually music. There are exceptions to this rule as demonstrated by the graph.

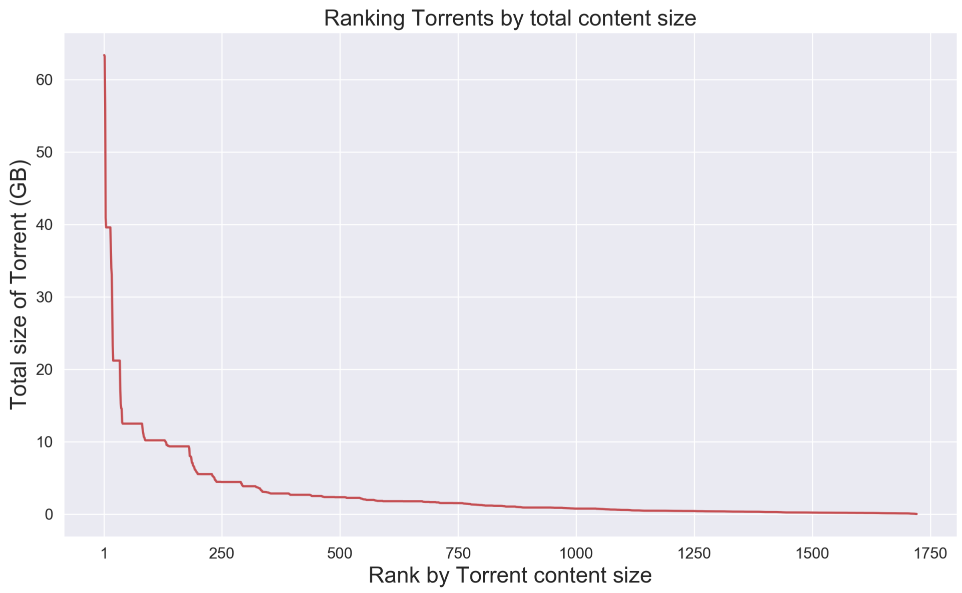

Figure 3.3: Ranking of Torrents based on their content size.

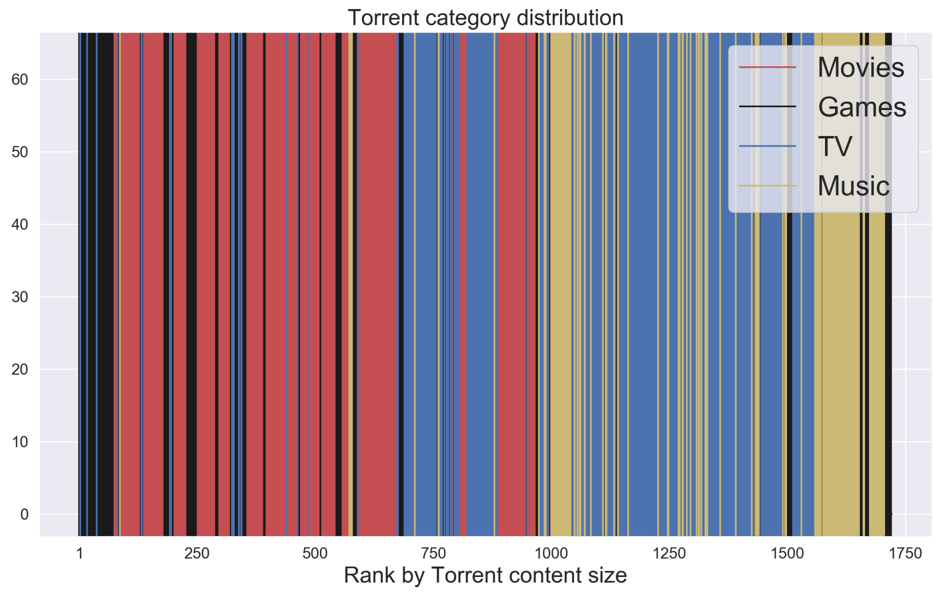

Figure 3.4: Category breakdown when torrents are ranked by content size.

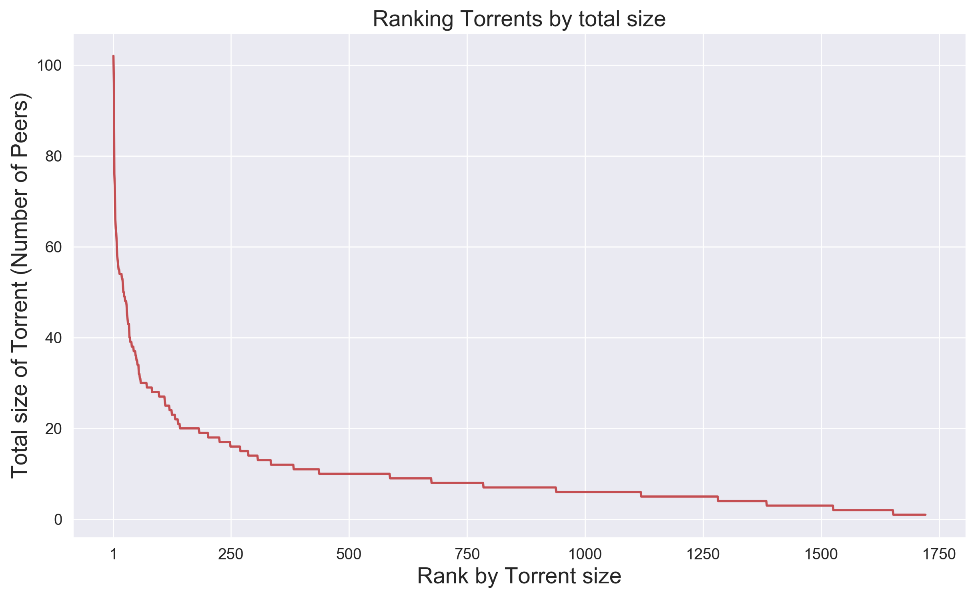

Figure 3.5: Ranking of Torrents based on their peer swarm size (popularity).

3.2.7 Ranking Torrents by Popularity

As shown in Figure 3.5, when torrents are ranked by peer swarm size, the torrent size exponentially decreases. This trend is similar to the pattern seen when ranking torrents by content size. Most torrents have less than 20 peers. The average size hovers around 7, with most having just 6 peers.

3.2.8 Category breakdown by Popularity

Figure 3.6 shows the category breakdown when ranking torrents by popularity. The distribution seems to have a discrete pattern. The most popular categories are TV and games with some music. Movies and TV show an alternating pattern when it comes to popularity outside of the top 200 popular torrents. Among the least popular torrents most are music.

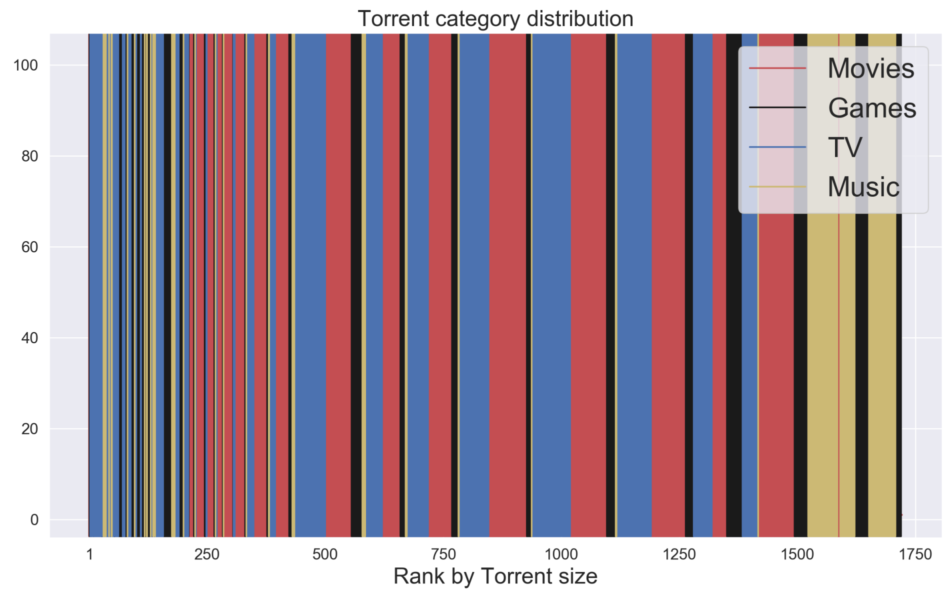

Figure 3.6: Category breakdown when torrents are ranked by popularity.

3.2.9 Comparing Content Size to Popularity

The largest peer swarms seems to somewhere in the middle of torrent rank size. Interestingly, there seems to be no direct correlation Figure 3.7.

3.2.10 Peer activity by Content Size

More activity is seen towards the left half. This could be because the bigger torrents are latest games and newer TV shows titles which have higher interest. Figure 3.8.

3.2.11 Peer activity by Popularity

The bigger peer swarm size exhibit more activity. This is expected in a Peer-to-Peer system. Figure 3.9.

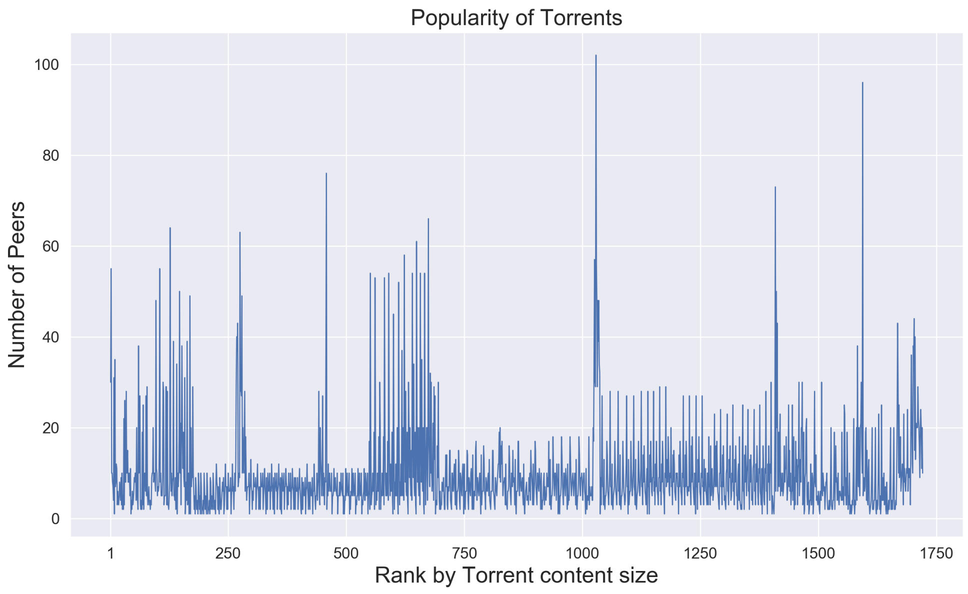

Figure 3.7: Number of peers in different torrents when ranked by content size.

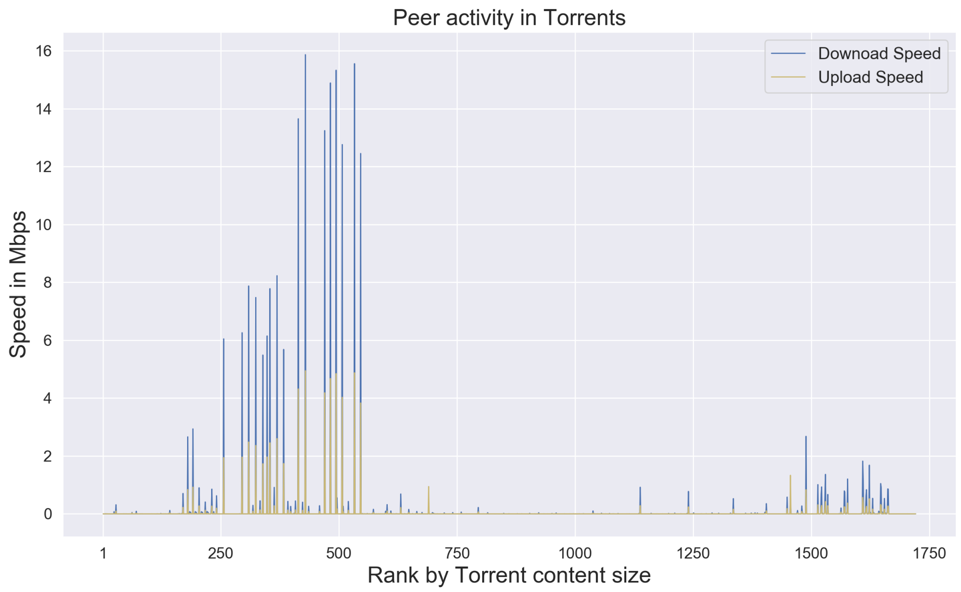

Figure 3.8: Peer activity in torrents when ranked by content size.

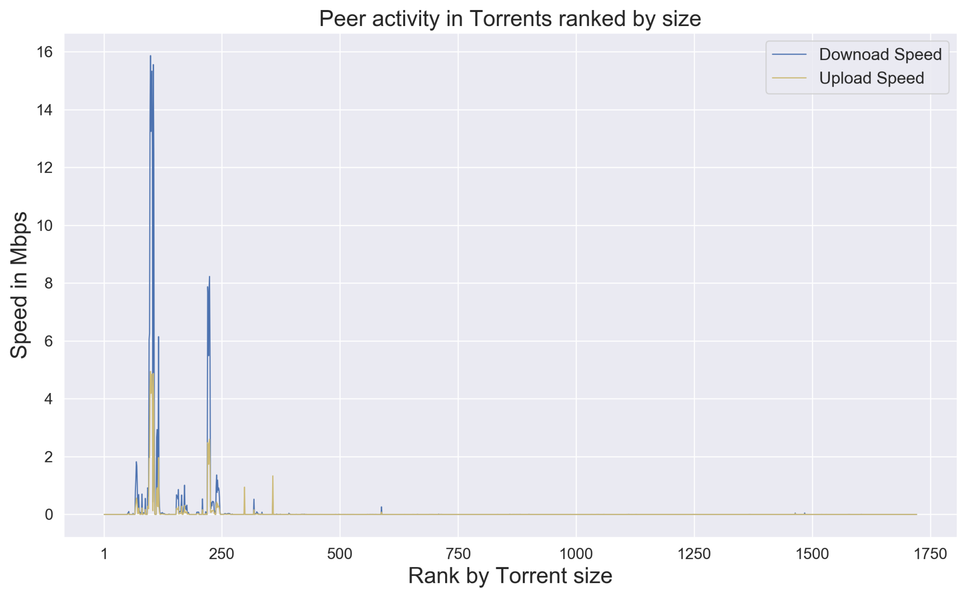

Figure 3.9: Peer activity in different torrents when ranked by popularity.

4 Modeling ISP Similarity

The data collected during the experiments has thrown up lots of interesting observations. Our main goal is to group all of the collected user data by ASes and then investigate if there are any underlying similarities in the grouped data. First, we start by grouping the data, then pre-process and format it, so that it's suitable for statistical analysis. And then finally, run it through a clustering algorithm.

4.1 Grouping Sampled Data by Autonomous Systems

4.1.1 Most Popular Autonomous Systems

After grouping peers by AS, the top 5 most popular ASes are visualized in Figure 4.1. Interestingly, 4 out of 5 most popular ISPs are from the US but only only 1 American ISP is in the top 10 best performers Figure 4.2. Peer interests in those ASes are further broken down. Interestingly, we find that the second most popular AS is Amazon. On further investigation we find that almost all of the users are cloud peers running on virtual machines on the Amazon AWS.

4.1.2 Ranking Autonomous Systems by Performance

Scores of AS performance are measured as a weighted sum of number of peers serviced and average upload and download speeds in the AS. Figure 4.2

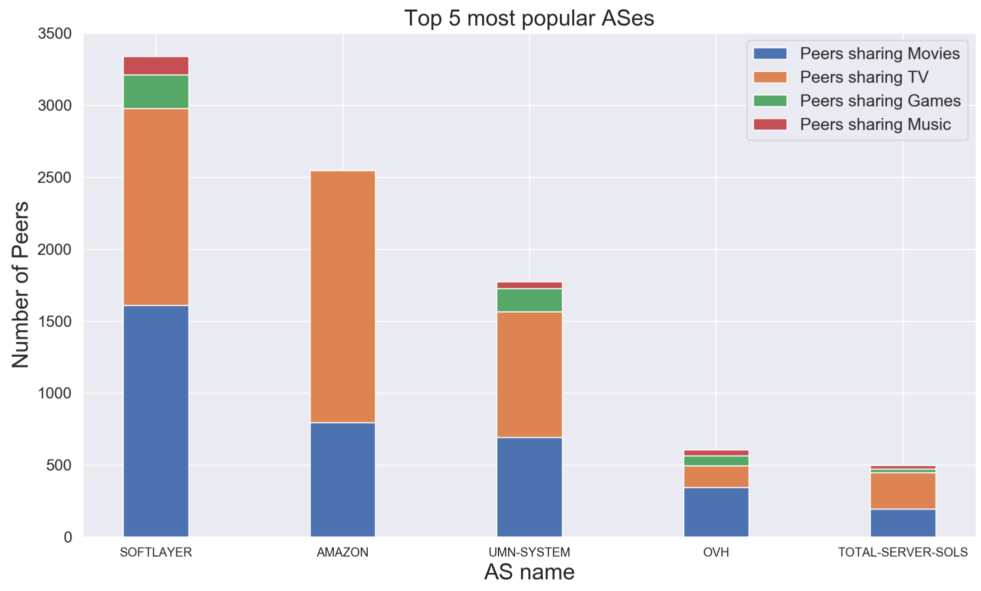

Figure 4.1: The top 5 most popular ASes found in the study.

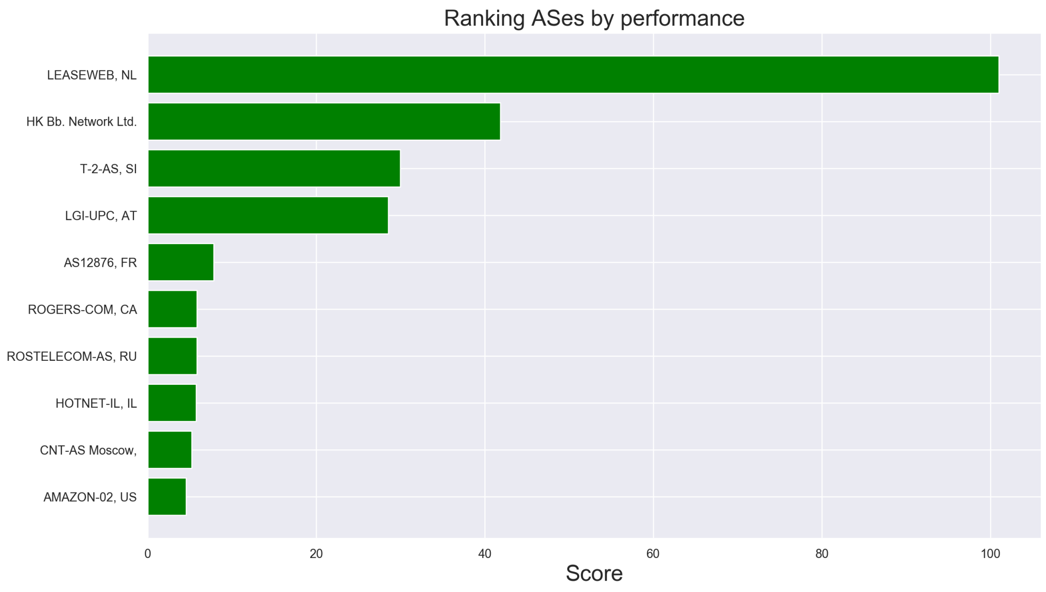

Figure 4.2: Ranking ASes based on performance.

4.2 Statistical Analysis

4.2.1 Pre-Processing Data

The data that we have collected contains the information of different peers sharing various torrents and being serviced by their host AS. This data-set can be used to describe a $i * j$ matrix $\boldsymbol{X}$. The i rows are for i different torrents and j columns are for j different ASes. The entry X_{ij} is the count of peers in AS j that are sharing the torrent i. Each AS can therefore be visualized as a i dimensional vector which we will call it's $profile$. The profile of the kth AS is

$$P_k = (X_1k, X_2k, ..., X_ik) \ where \ k = 1, 2...j.$$

To build this matrix we have ordered the the columns in X according to the descending order of their popularity Figure 4.1. The rows have also been ordered according t the most popular torrents. The second one which is Amazon is of particular interest. When building the matrix with all 1097 ASes and 1721 torrents, we notice that it is a very sparse matrix. This is also supported by our observations in the previous section that popularity seems to be decreasing exponentially when it comes to torrents. The top ten percent of the torrents show around ninety percent of the activity. Due to these observations, we decided to consider only the top 100 torrents and ASes to make the matrix X. Also, considering torrents with minuscule count of peers might not give us accurate insights.

This matrix can also be imagined as a bipartite graph Figure 4.3. In such a graph the vertices can be divided into two sets. There are no edges between vertices of the same set. Matrix X can be thought of as a bipartite graph with one set containing ASes and the other set containing torrents. The edge weight is proportional to the number of peers between a particular AS and torrent.

4.2.2 Projection

As stated earlier, the main goal of this thesis is to find similarity between ASes. Torrents serve as the linkages between these ASes. To visualize similarity, we can project the AS space as a graph. Since every vertex (or AS) is a sequence, projection of columns in matrix X will give us points in a hyper dimensional plane. To make calculations easier we can condense this graph to two dimensions. This gives us a j vertex 2D graph with every vertex connected to every other vertex. The edge weight gives us the level of similarity between two ASes. There are several ways to define similarity between two sequences, we chose the Pearson correlation coefficient because of its effectiveness in similarity measurement of sequences[23].

Consider two profiles P_k and P_l, each being a m sized sequence. The formula for Pearson correlation coefficient for a sequence is:

$$p_{k,1} = \frac{\sum_{i=1}^{m}(X_{ik} - \overline{X_k})(X_{il} - \overline{X_l})}{\sqrt{\sum_{i=1}^{m}(X_{ik} - \overline{X_k})^2}\sqrt{\sum_{i=1}^{m}(X_{il} - \overline{X_l})^2}}$$

Here $\overline{X_k}$ is the mean of kth AS profile. Now we create a j x j matrix called Y that contains the result of p_{kl} i.e. the correlation of P_k and P_l for every AS pair (k, l). Each entry of Y is between -1 and 1. Higher values indicate higher correlation between two ASes.

4.3 Summary Statistics

The matrix Y gives a good idea of the overall relationship between ASes. The matrix is visualized as a heat map in Figure 4.4. To get a more generalised idea, we

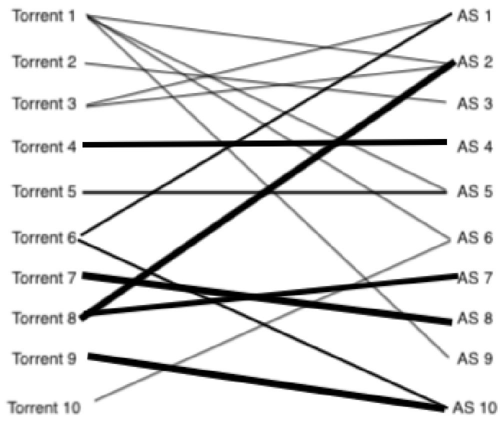

Figure 4.3: A bipartite graph to visualize the data-set. Over here, only the first ten ASes and Torrents are shown. The width of the line connecting the torrent to the AS is commensurate to the total number of peers in that AS downloading that torrent.

calculate a summary statistic of the correlations. If we condense every profile to just its mean value and plot it we should get an overall idea of relationships. Figure 4.5 throws up a very interesting insight. Torrent at index 1 and index 7 stand out from the rest of the ASes as they are the only one with negative correlation. This means that on average these two ASes have traffic that is significantly different from all other ASes. Both these are Amazon ASes. These are basically cloud users on Amazon EC2 instances. This shows us that cloud users have significantly different interest from traditional ASes.

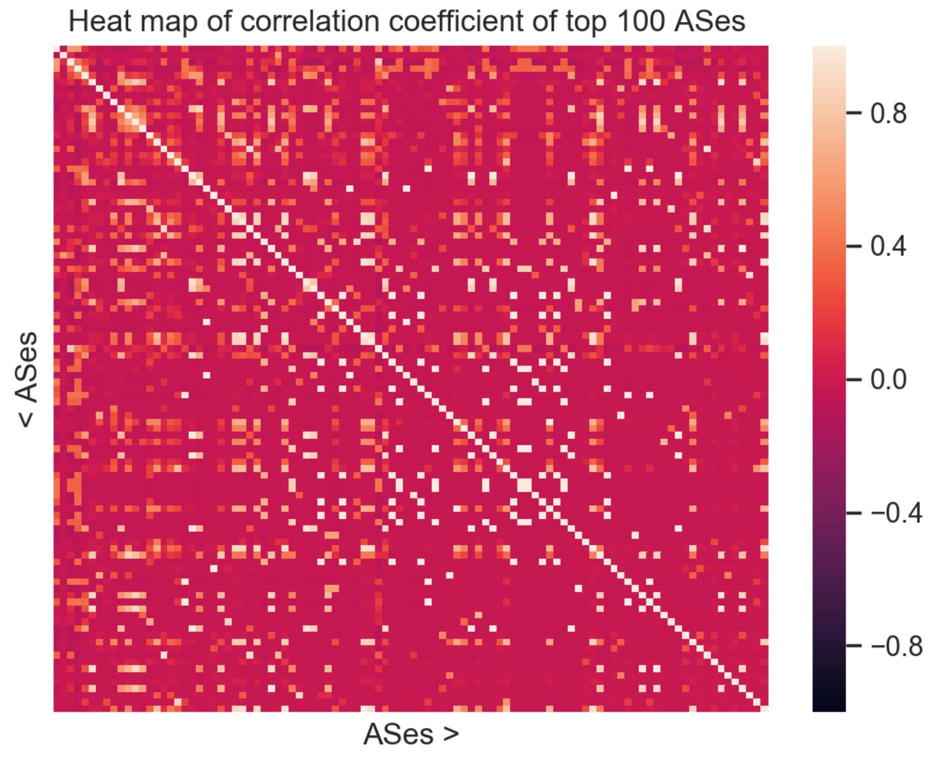

Figure 4.4: Heat map of matrix Y showing the correlation between the profiles of all AS.

4.4 Hierarchical Clustering

Figure 4.5 shows us that users in these cloud ASes have very different interests from other ASes in terms of downloading behavior. But by condensing the profiles to mean values would mean that we are loosing out some details as they do not take into consideration specific relationship between individual ASes.

To have a more granular analysis, we used unsupervised learning to form clusters of ASes based on their similarity. The matrix entry Y_{kl} gives us the correlation between two AS profiles P_k and P_l. This matrix can be considered as a distance matrix. The distance between two ASes would be smaller if the value of the correlation is higher.

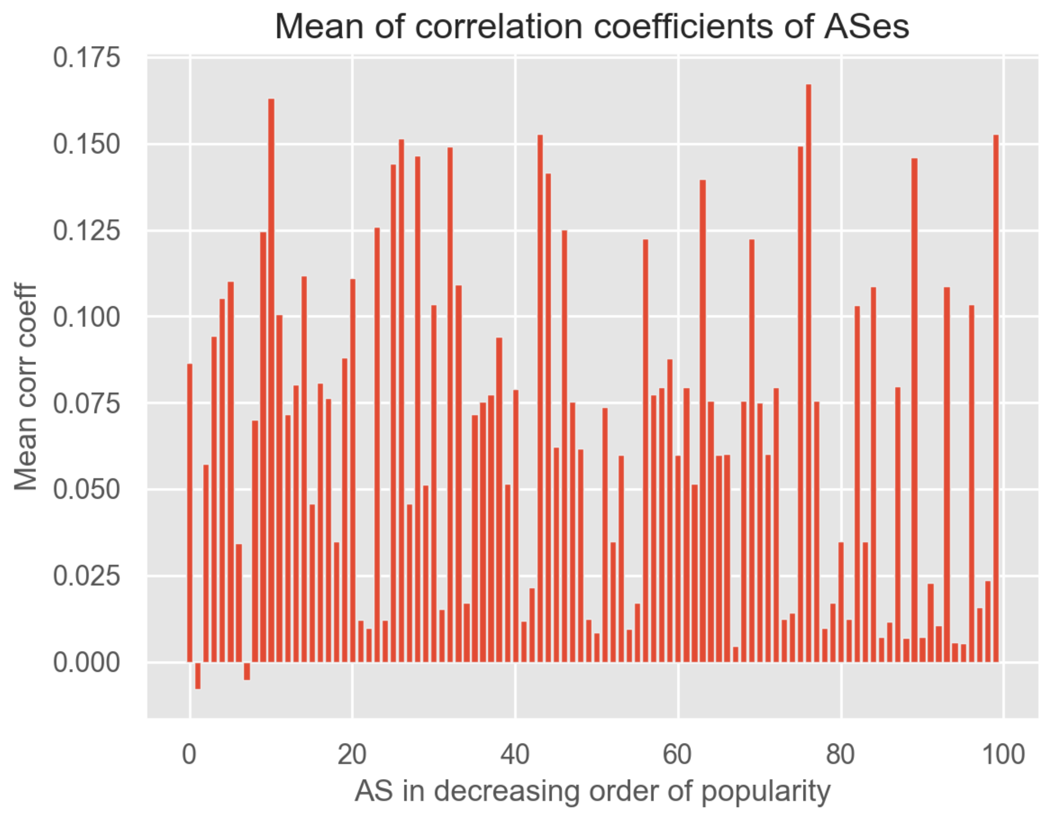

Figure 4.5: Mean of the Pearson correlation coefficient of profiles of an AS to every other AS.

To make calculations easier we consider matrix $Z = 1 - Y$ in place of Y. This will be easier to process as matrix entries are directly proportional to distances. An entry in Z will give us larger values for larger distances and smaller value for smaller distances.

Initially, to develop clusters, we had considered ASes as points on a multi dimensional hyper plane and tried applying a K means clustering approach. But due to highly dimensionality, the approach did give us any useful results. Conversely, using the distance matrix Z we can use it in a hierarchical clustering approach. We use the agglomerative hierarchical clustering method. This approach builds up a hierarchy of clusters in a bottom up fashion and the results can be visualised as a dendrogram.

After running the clustering alogrithm we see that the cloud ASes cluster together as shown in Figure 4.6. This further cements that cloud users have special preferences which are different from rest of the ASes. Also when visualizing the ASes by countries Figure 4.7, we see that ASes that are closer geographically tend to cluster together. This gives us another interesting pattern.

Cluster Dendrogram - Cloud ASes

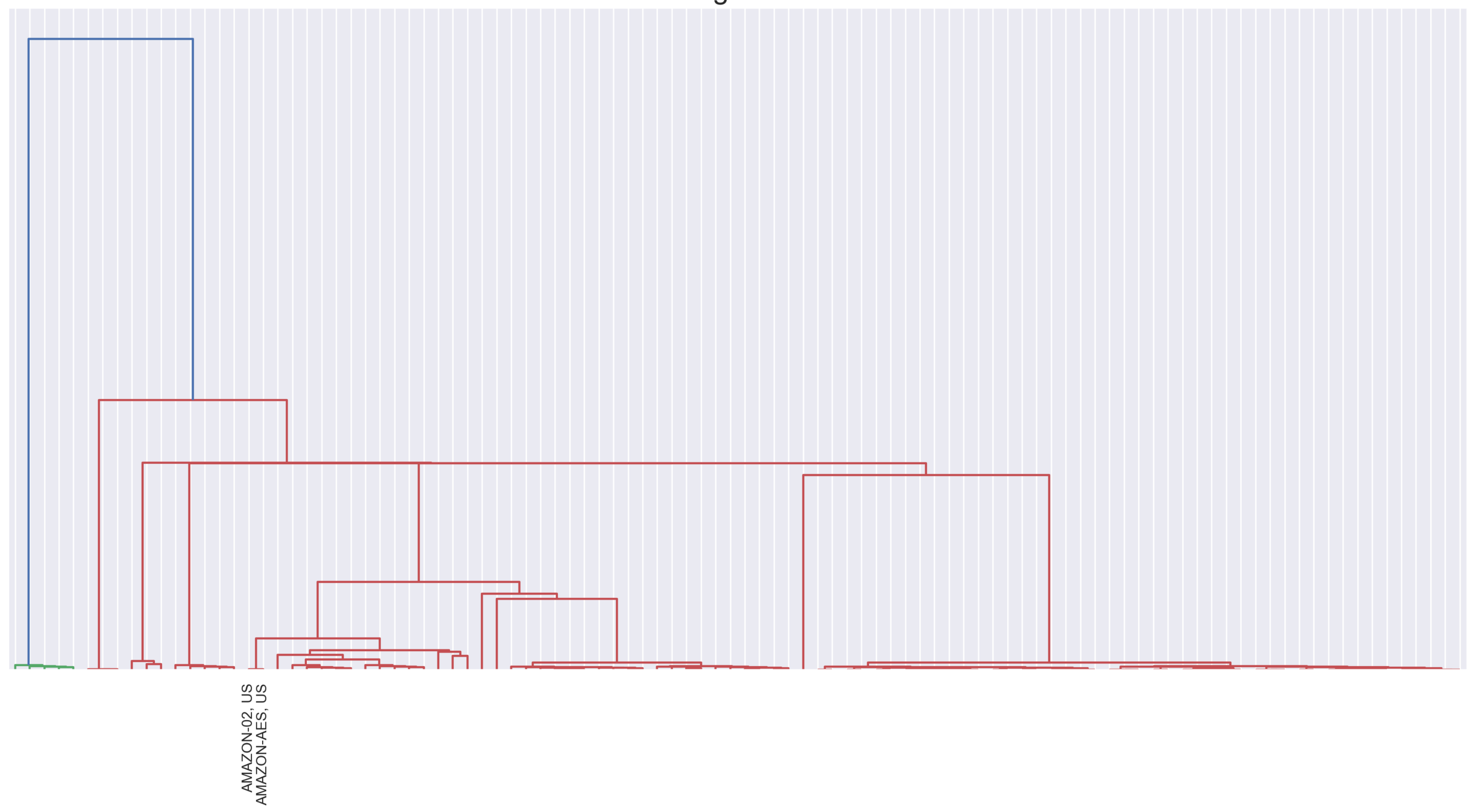

Figure 4.6: Dengrogram showing the clustering of Cloud ASes.

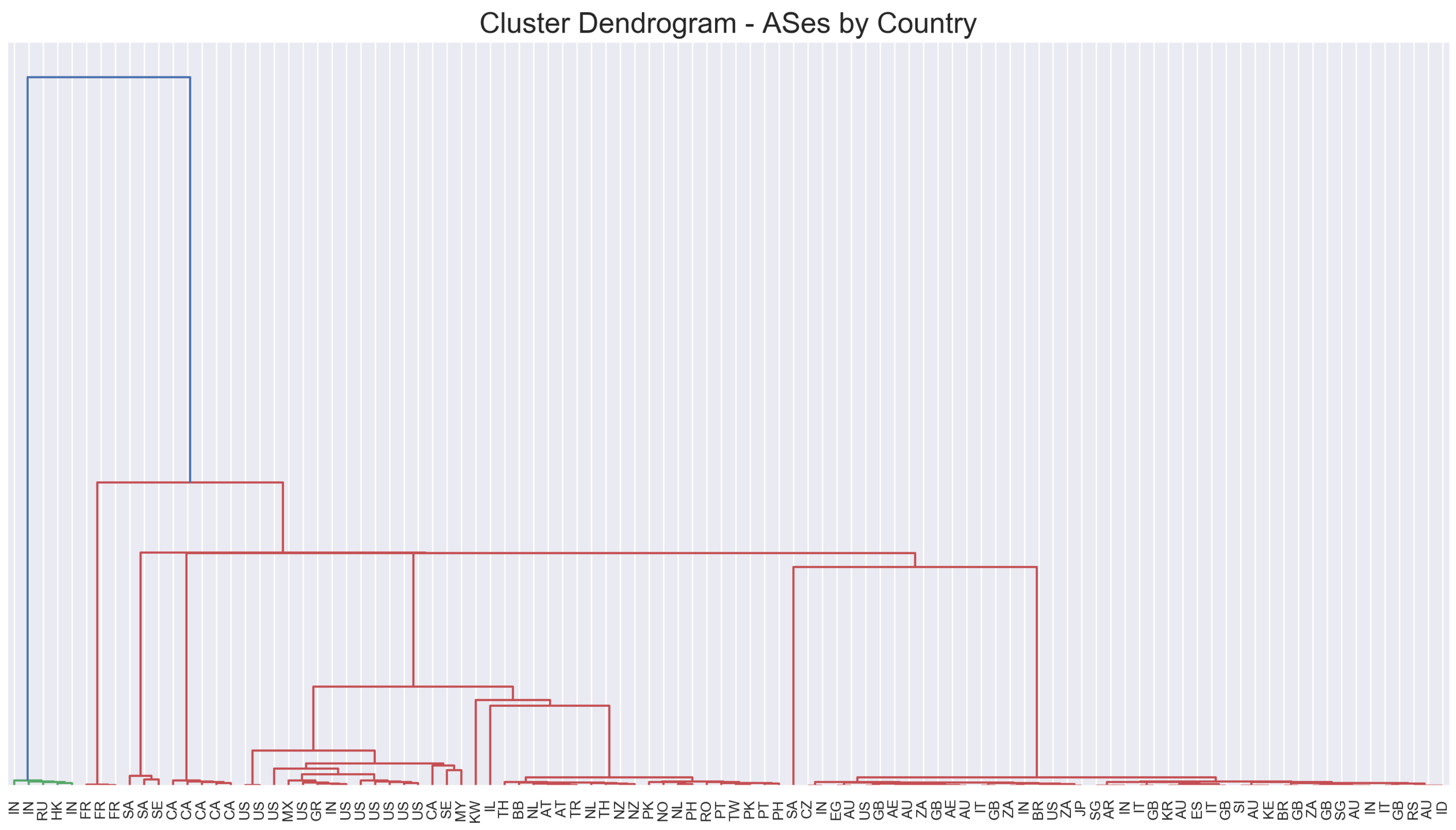

Figure 4.7: Dengrogram illustrating how ASes cluster based on their geographic locations.

5 Conclusions

In this thesis, we have systematically described, discussed and analyzed the peer and torrent information in BitTorrent traffic. After the measurement and subsequently analyzing the data, we obtained several interesting observations. The vast majority of torrents fall into four categories: TV series, Movies, Games and Music. Ranking torrents by content size shows an exponential decrease in size. The bigger content size is usually for Games. Movies and TV are the most popular content in the data-set forming 70 percent of the total torrents. The popularity of Movies and TV shows if further validated when we study how the torrent size relate to content size. We see that most peers are present in average sized torrents. Interestingly though, most activity is seen in larger torrents and much less in smaller torrents. This behavior might be explained by the creation_date of these torrent. Usually, torrents created more recently exhibit higher download/upload activity and most of these active torrents are the latest Games and TV shows. The low speeds for the majority of the torrents can be understood by two explanations. Firstly, most Movies and TV series are copyrighted materials and ISPs usually suppress the sharing of such files. Secondly, our data capturing tool *Torrent Scrapper* limits the content download/upload speed as it closes the connection to the torrent immediately after recording the peer swarms download/upload speed.

After studying the torrent and peer statistics, we processed the collected data and grouped it by Autonomous Systems. After grouping the ASes and their traffic we hit upon an interesting discovery. Our experiments for the first time have shown the presence of cloud peers in BitTorrent traffic. We found the presence of a large number of peers from Amazon ASes. Upon investigation, we found out that these peers are essentially clients from ec2 instances. We find that Amazon AS is the second largest AS among all the ASes and that users in it have a different category breakdown than the other top 5 ASes. To better understand the relationship in among ASes we model these ASes as graphs vertices, with edge weights defining the similarity between two ASes. We use the Pearson correlation coefficient to model this graph. We create a j x j matrix Y, which contains the correlation among j different ASes. We describe a column of this matrix as an AS correlation sequence. Condensing this sequence to its average value threw up a very interesting observation. We found that every AS has an overall positive correlation with every other AS except the two Amazon ASes. This shows us that cloud peers have significantly different interest from peers in traditional ASes. Taking averages gives us an overall picture but it diminishes the granular details. To further investigate the similarity of ASes we choose to use a hierarchical clustering approach. We use the values stored in matrix $Z = 1 - Y$ and use that as a distance matrix for our clustering algorithm. The clustering analysis further solidified our earlier finding as the two cloud ASes clustered together. Another interesting insight that we found via the clustering was that most ASes form groups with other ASes that are closer to their geographical locations. This means that user interests are very similar for geographically closer ASes.

With this thesis, we have taken the inceptive steps towards understanding the preferences of cloud peers. Our experiments have for the first time given insight into the similarity of these cloud users by showing that these users have remarkably different download interests than traditional users. The future work for this thesis would be to focus further into these cloud peers and study them in more detail and try to understand what makes these users so unique. We also hypothesise that these cloud users are increasing and a temporal analysis might give us more insight into it. This should help in the further development of more robust caching implementations thus improving network traffic.

References

[1] O. A. Abdul-Rahman and K. Aida. "Towards Understanding the Usage Behavior of Google Cloud Users: The Mice and Elephants Phenomenon". In: *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. Dec. 2014, pp. 272–277. DOI: 10.1109/CloudCom.2014.75 (cit. on p. 2).

[2] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker. "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport". In: *Proceedings of the IEEE* 99.1 (Jan. 2011), pp. 149–167. ISSN: 0018-9219. DOI: 10.1109/JPROC.2010.2060451 (cit. on p. 17).

[3] R. Bustos, A. Aguilar, K. Makki, and R. K. Ege. "Multicast-P2P content distribution in large-scale enterprise networks". In: *2008 IEEE Symposium on Computers and Communications*. July 2008, pp. 487–494. DOI: 10.1109/ISCC.2008.4625722 (cit. on pp. 8, 12).

[4] G. Chen, C. P. Low, and Z. Yang. "Enhancing Search Performance in Unstructured P2P Networks Based on Users' Common Interest". In: *IEEE Transactions on Parallel and Distributed Systems* 19.6 (June 2008), pp. 821–836. ISSN: 1045-9219. DOI: 10.1109/TPDS.2008.42 (cit. on p. 17).

[5] "Cisco Visual Networking Index: Forecast and Trends, 2017 - 2022 White Paper". In: 1 (June 2019) (cit. on p. 1).

[6] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan. "Cache content-selection policies for streaming video services". In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. Apr. 2016, pp. 1–9. DOI: 10.1109/INFOCOM.2016.7524619 (cit. on p. 17).

[7] I. Foster, Y. Zhao, I. Raicu, and S. Lu. "Cloud Computing and Grid Computing 360-Degree Compared". In: *2008 Grid Computing Environments Workshop*. Nov. 2008, pp. 1–10. DOI: 10.1109/GCE.2008.4738445 (cit. on p. 17).

[8] "Google service level agreement." In: 1 (June 2019). URL: https://cloud.google.com/terms/sla/ (cit. on p. 2).

[9] Z. Guo, H. Jin, C. Zhao, and D. Liang. "User interest based distributed cooperative caching and sharing in wireless networks". In: *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*. Sept. 2016, pp. 466–470. DOI: 10.1109/ICNIDC.2016.7974618 (cit. on p. 16).

[10] "Internet Transit Prices - Historical and Projected". In: 1 (Aug. 2010) (cit. on p. 1).

[11] "Internet World Stats". In: 1 (June 2019) (cit. on p. 1).

[12] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing". In: *IEEE Transactions on Parallel and Distributed Systems* 22.6 (June 2011), pp. 931–945. ISSN: 1045-9219. DOI: 10.1109/TPDS.2011.66 (cit. on p. 17).

[13] B. Kahanwal and T. P. Singh. "The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle". In: *International Journal of Latest Research*

in Science and Technology, Vol. 1, Issue 2. July 2012, pp. 183–187. DOI: 10.1109/CLOUD.2011.84 (cit. on pp. 7, 9).

[14] H. Kavalionak and A. Montresor. *P2P and Cloud: A Marriage of Convenience for Replica Management*. Mar. 2012 (cit. on p. 9).

[15] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville. "Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS". In: *2010 IEEE 3rd International Conference on Cloud Computing*. July 2010, pp. 450–457. DOI: 10.1109/CLOUD.2010.37 (cit. on p. 7).

[16] J. Lu and J. Callan. "User Modeling for Full-text Federated Search in Peer-to-peer Networks". In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '06. Seattle, Washington, USA: ACM, 2006, pp. 332–339. ISBN: 1-59593-369-7. DOI: 10.1145/1148170.1148229. URL: http://doi.acm.org/10.1145/1148170.1148229 (cit. on p. 16).

[17] A. Montresor and L. Abeni. "Cloudy weather for P2P, with a chance of gossip". In: *2011 IEEE International Conference on Peer-to-Peer Computing*. Aug. 2011, pp. 250–259. DOI: 10.1109/P2P.2011.6038743 (cit. on p. 7).

[18] P. Mvelase, H. Sithole, T. Modipa, and S. Mathaba. "The economics of cloud computing: A review". In: *2016 International Conference on Advances in Computing and Communication Engineering (ICACCE)*. Nov. 2016, pp. 159–167. DOI: 10.1109/ICACCE.2016.8073741 (cit. on p. 9).

[19] Ning Man, Chen Xunxun, and Wang Bo. "Hierarchical user interest model based on large log data of mobile internet". In: *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*. June 2016, pp. 1–5. DOI: 10.1109/ICSSSM.2016.7538574 (cit. on p. 16).

[20] W. T. P. Wieder J. Butler and R. Yahyapour. "Service Level Agreements for Cloud Computing". In: 1 (Aug. 2011) (cit. on p. 2).

[21] C. Qin, Z. Yang, and H. Liu. "User Interest Modeling for P2P Document Sharing Systems Based on K-Medoids Clustering Algorithm". In: *2014 Seventh International Joint Conference on Computational Sciences and Optimization*. July 2014, pp. 576–578. DOI: 10.1109/CSO.2014.113 (cit. on p. 16).

[22] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. "Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis". In: *Proceedings of the Third ACM Symposium on Cloud Computing*. SoCC '12. San Jose, California: ACM, 2012, 7:1–7:13. ISBN: 978-1-4503-1761-0. DOI: 10.1145/2391229.2391236. URL: http://doi.acm.org/10.1145/2391229.2391236 (cit. on p. 2).

[23] "Using networks to measure similarity between genes: association index selection". In: 1 (June 2019) (cit. on p. 39).

[24] N. Xia, S. Miskovic, M. Baldi, A. Kuzmanovic, and A. Nucci. "GeoEcho: Inferring User Interests from Geotag Reports in Network Traffic". In: *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. Vol. 2. Aug. 2014, pp. 1–8. DOI: 10.1109/WI-IAT.2014.73 (cit. on p. 16).

[25] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. "Understanding User Behavior in Large-scale Video-on-demand Systems". In: *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*. EuroSys '06. Leuven, Belgium: ACM, 2006, pp. 333–344. ISBN: 1-59593-322-0. DOI: 10.1145/1217935.1217968. URL: http://doi.acm.org/10.1145/1217935.1217968 (cit. on p. 17).