Study of Peer-to-Peer Network Based Cybercrime Investigation: Application on Botnet Technologies

by

Mark Scanlon, B.A. (Hons.), M.Sc.

A thesis submitted to University College Dublin
for the degree of Ph.D. in the College of Science

October 2013

School of Computer Science and Informatics
Mr. John Dunnion, M.Sc. (Head of School)
Under the supervision of
Prof. M-Tahar Kechadi, Ph.D.

DEDICATION

This thesis is dedicated to my wife, Joanne, who has supported, encouraged and motivated me throughout the last nine years and has been especially patient and thoughtful throughout my research. This thesis is also dedicated to my parents, Philomena and Larry Scanlon.

CONTENTS

ACKNOWLEDGEMENTS

With no doubt, the work on this thesis has been the most challenging endeavour I have undertaken so far. I am thankful to my supervisor, Prof. M-Tahar Kechadi, for his guidance and encouragement. I would like to thank the staff and students in the School of Computer Science and Informatics, University College Dublin for providing me with the opportunity to learn, facilities to perform my research, and a motivating environment that carried me forward through my course work. My gratitude goes to my friends Alan Hannaway, Cormac Phelan, John-Michael Harkness, Michael Whelan, Alex Cronin, Pat Tobin, Jason Farina and Dr. Pavel Gladyshev for many interesting and developing discussions, presentations and collaborations. Many thanks to all my immediate friends for their constant encouragement and support.

This work was co-funded by the Irish Research Council (formally the Irish Research Council for Science, Engineering and Technology) and Intel Ireland Ltd., through the Enterprise Partnership Scheme. Amazon Web Services also generously contributed to this research with grants funding the costs involved in experimentation conducted on their cloud infrastructure, including Elastic Compute Cloud (EC2) and Relational Database Service (RDS).

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACTA	Anti-Counterfeiting Trade Agreement
API	Application Programming Interface
BEP	BitTorrent Enhancement Proposal
C&C	Command and Control
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CFTT	Computer Forensics Tool Testing
DDNS	Dynamic Domain Name Server
DDoS	Distributed Denial of Service
DEB	Digital Evidence Bag
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DNS	Domain Name System
DPI	Deep Packet Inspection
EC	European Commission
EC2	Elastic Compute Cloud
ENISA	European Network and Information Security Agency

FAT	File Allocation Table
FRED	Forensic Recovery of Evidence Device
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IDE	Integrated Drive Electronics
IDS	Intrusion Detection System
IFPI	International Federation of the Phonographic Industry
IP	Internet Protocol
IRC	Internet Relay Chat
ISP	Internet Service Provider
LOIC	Low Orbit Ion Cannon
MD	Message Digest
MPAA	Motion Picture Association of America
NAT	Network Address Translation
NFAT	Network Forensic Analysis Tool
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NTFS	New Technology File System
P2P	Peer-to-Peer
PEX	Peer Exchange
PIPA	Preventing Real Online Threats to Economic Creativity and Theft (PROTECT) of Intellectual Property Act

RPC	Remote Procedure Call
SATA	Serial Advance Technology Attachment
SHA	Secure Hashing Algorithm
SIEM	Security Incident and Event Management
SOPA	Stop Online Piracy Act
SSH	Secure Socket Handling
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TPP	Trans-Pacific Partnership
UP2PNIF	Universal P2P Network Investigation Framework
URI	Uniform Resource Identifier

ABSTRACT

The scalable, low overhead attributes of Peer-to-Peer (P2P) Internet protocols and networks lend themselves well to being exploited by criminals to execute a large range of cybercrimes. The types of crimes aided by P2P technology include copyright infringement, sharing of illicit images of children, fraud, hacking/cracking, denial of service attacks and virus/malware propagation through the use of a variety of worms, botnets, malware, viruses and P2P file sharing. This project is focused on study of active P2P nodes along with the analysis of the undocumented communication methods employed in many of these large unstructured networks. This is achieved through the design and implementation of an efficient P2P monitoring and crawling toolset.

The requirement for investigating P2P based systems is not limited to the more obvious cybercrimes listed above, as many legitimate P2P based applications may also be pertinent to a digital forensic investigation, e.g, voice over IP, instant messaging, etc. Investigating these networks has become increasingly difficult due to the broad range of network topologies and the ever increasing and evolving range of P2P based applications. In this work we introduce the Universal P2P Network Investigation Framework (UP2PNIF), a framework which enables significantly faster and less labour intensive investigation of newly discovered P2P networks through the exploitation of the commonalities in P2P network functionality. In combination with a reference database of known network characteristics, it is envisioned that any known P2P network can be instantly investigated using the framework, which can intelligently determine the best investigation methodology and greatly expedite the evidence gathering process. A proof of concept tool

was developed for conducting investigations on the BitTorrent network. A Number of investigations conducted using this tool are outlined in Chapter 6.

LIST OF PUBLICATIONS

- Peer-Reviewed International Journal Publications

  M. Scanlon, A. Hannaway, and M-T. Kechadi. Investigating Cybercrimes that Occur on Documented P2P Networks. *International Journal of Ambient Computing and Intelligence (IJACI)*, 3(2):56–63, 2011.

- Peer-Reviewed International Conference Publications

  M. Scanlon and M-T Kechadi. The Case for a Universal P2P Botnet Investigation Framework. In *9th International Conference on Cyber Warfare and Security ICCWS-2014*, West Lafayette, IN, USA, March 2014. ACPI, [Extended Abstract Accepted, Paper Pending Final Decision].

  M. Scanlon and M-T. Kechadi. Universal Peer-to-Peer Network Investigation Framework. In *International Workshop on Emerging Cyberthreats and Countermeasures (ECTCM 2013), part of the Eight International Conference on Availability, Reliability and Security (ARES2013)*, Regensburg, Germany, September 2013. IEEE.

  M. Scanlon and M-T. Kechadi. Digital Evidence Bag Selection for P2P Network Investigation. In James J. (Jong Hyuk) Park, Victor C.M. Leung, Cho-Li Wang, and Taeshik Shon, editors, *Future Information Technology, Application, and Service; The 8th International Symposium on Digital Forensics and Information Security (DFIS-2013)*, Lecture Notes in Electrical Engineering. Springer Netherlands, 2013.

M. Scanlon and M-T. Kechadi. Peer-to-Peer Botnet Investigation: A Review. In James J. (Jong Hyuk) Park, Victor C.M. Leung, Cho-Li Wang, and Taeshik Shon, editors, *Future Information Technology, Application, and Service; The 7th International Symposium on Digital Forensics and Information Security (DFIS-12)*, volume 179 of *Lecture Notes in Electrical Engineering*, pages 231–238. Springer Netherlands, 2012.

M. Scanlon, A. Hannaway, and M-T. Kechadi. A Week in the Life of the Most Popular BitTorrent Swarms. *5th Annual Symposium on Information Assurance (ASIA'10)*, 2010.

M. Scanlon and M-T. Kechadi. Online Acquisition of Digital Forensic Evidence. In *Proceedings International Conference on Digital Forensics and Cyber Crime (ICDF2C 2009)*, Albany, New York, USA, September 2009. Elsevier Limited.

- International Conference Posters

M. Scanlon. Investigating Cybercrimes that Utilise Peer-to-Peer Internet Communications. In *Intel European Research and Innovation Conference (ERIC '10)*, 2010.

CHAPTER

ONE

INTRODUCTION

1.1 Background

In June 1999, the control that the content producing industry (composed of movie producers, TV show producers, musicians, writers, etc.) had over their traditional distribution model was permanently changed due to the release, and subsequent rise in popularity, of Napster by Shawn Fanning [9]. Napster brought the relatively new concept of Internet file sharing into the mainstream. It facilitated its users in sharing music with millions of other users around the world. The ease of use, vast library of available content, perceived anonymity and zero cost model enabled Napster to grow rapidly. It's rise in popularity also coincided with the release of new portable devices capable of playing digital audio files, MP3 players [10]. The difference in user difficulty between converting store bought CDs into a suitable format when compared to performing a search for the song's title and double clicking the version you wanted was significant. At its peak, it enabled over 25 million users to share more than 80 millions digital songs with each other [11]. This was not the first implementation of Peer-to-Peer (P2P) technology, but it certainly was the first to gather attention. It enabled regular computer users with Internet connections to perform copyright infringement on a scale incomparable to physical copying of tapes and CDs.

P2P technologies are most known for unauthorised distribution of copyrighted

content but the merits of P2P have been exploited by other criminals with more sinister intentions. The ever increasing proliferation of computers has resulted in a new breed of high-tech, highly skilled, computer savvy criminals emerging. For the lesser skilled criminal, a large underground market creating and selling software packages to enable the online execution of a range of crimes has emerged. As this phenomenon continues, an increasing number of "offline" crimes are being aided by computers, e.g., fraud, identity theft, phishing, terrorism, child sexual exploitation, etc. As a result, digital forensic investigators and law enforcement in general are playing catchup in an attempt to gain the necessary expertise to combat these crimes. Looking to always be one step ahead of the law, criminals are continually looking for more advanced methods of conducting their crimes. With the advent of "botnets", i.e., large distributed networks of compromised machines, criminals are now able to take advantage of far superior distributed processing power, bandwidth and other resources than a single machine could ever afford them. These botnets also award the criminal a relative degree of anonymity if the botnet itself is entirely decentralised, i.e., no central server or single point of penetration, such as a P2P botnet. Each compromised node in a P2P botnets is obliged to forward on received commands and queries to other known active nodes in the network. The scalable and minimal investment attributes of P2P and similar distributed Internet protocols lend themselves well to being exploited by criminals to execute a range of cybercrimes. These crimes not only include those offline examples previously mentioned, but also new computer targeted crimes, such as distributed denial of service (DDoS) attacks, virus/malware propagation, etc.

1.2 Research Problem

Much of the existing research into P2P cybercrimes relies on packet sniffing as the primary method for collecting information. This method involves setting up a honeypot, as outlined in greater detail in Section 4.6.2, and deliberately

infecting the machine with the required malware. The downside of this type of investigation is that the system is reliant on recording typical network communication to find out information about the system being investigated. Any single node on a P2P network may never communicate with every other node, as each node generally maintains a list in the order of 5–10 other known active nodes. The motivation for the research detailed in this thesis is to design and test a new methodology for investigating P2P networks. This methodology involves emulating and multiplying regular client usage resulting in the distributed capability of crawling an entire network.

The objectives of this research are as follows:

1. Provide an insight into the technical requirements of the design and implementation of a forensically sound P2P crawling and investigation tool; collecting of digital evidence and the counter-detection measures that may need to be employed.
2. Demonstrate the application of a P2P network crawling system as a plausible option for forensic investigation.
3. Design an architecture for such a system. It should be forensically verifiable, cost effective, expandable, reliable and widely compatible with current computer hardware and network capacities.
4. Prototype the system and perform experimental analysis to measure the viability of the system for both documented and undocumented networks.
5. Draw some recommendations about future use of these technologies.

1.3 Contribution of this Work

Many of the tools available in the field of digital network forensic investigations are based upon the deployment of packet sniffing or deep packet inspection devices and software, which are outlined further in Sections 2.3 and 2.4 . These methods can result in a huge volume of data to be analysed by the forensic investigator. "Typically, only a small fraction of the examined data is of interest in an investigation" [12]. The existing techniques are concentrated around the procedures that should be implemented after the physical confiscation of the computer equipment. The research outlined as part of this thesis results in a system capable of quickly implementing the communication protocol of any given P2P network, resulting in more focused data collection. The data collected can be partially processed at the point of collection, eliminating the need to store, index and analyse irrelevant information.

The contribution of this research can be summarised as follows:

- Design of a forensically sound P2P network investigation system, which can be used for the collection of court-admissible evidence or used for system monitoring. The system also enables the user to conduct a cloud based investigation. This results in the forensic investigators being able to spend more time analysing evidence, as opposed to being in the field collecting it. The design approach can be extended to defining how to best deal with the issues of cost, speed, compatibility and redundancy of the data while ensuring that the process is reproducible and reliable.

- Proof of the viability of the system through experimentation of all the necessary components. Each component of the system was individually tested to ensure the forensic integrity of the data collected.

- Performance results from testing "real-world" scenarios where such a system may be used, i.e., collecting evidence from a live P2P network investigation.

- Outline a new forensically sound method for storing remote network captured P2P evidence.

1.4 Limitations of this Work

With such a large variety of P2P networks and P2P based cybercrimes, a number of limitations for the scope of this research were introduced:

1. To conduct comprehensive testing across every known P2P network was deemed too large a task for the purposes of this work due to time and resource constraints.

2. As a proof of concept for the viability of the system designed as part of this work, it was deemed acceptable to perform testing and investigation of unauthorised file-sharing occurring on P2P networks. The methodology and techniques outlined are equally applicable to the investigation of any P2P based cybercrimes.

1.5 Structure of the Thesis

This thesis is organised as follows:

- After introducing the context and highlighting the main goals of the project in this Chapter, in Chapters 2, 3, and 4, we present literature reviews of related research work and software tools relevant to the areas of Digital Forensics, P2P File-sharing and Botnet Investigation respectively. These chapters outline some of the tools, systems, architectures, and best practices associated with the corresponding fields from a technical, and legal perspective.

- Chapter 5 presents the architecture and design of the universal P2P network investigation framework capable of expansion to deal with any

P2P network investigation. We also outline the design considerations which should be incorporated into a framework of this nature. Chapter 6 presents the results from a proof of concept investigation tool developed for the investigation of the BitTorrent file-sharing P2P network. The results of comprehensive experiments carried out to prove the viability of such a framework. This testing phase incorporated the testing of each individual component of the system to ensure forensic integrity and ultimately, court admissible evidence.

- Chapter 7 summarises and concludes this research. This chapter also outlines scenarios where the technology developed can be adapted and reused for additional purposes. Guidelines for further developments to the presented work are also outlined and discussed.

CHAPTER

TWO

DIGITAL FORENSIC INVESTIGATION; STATE OF THE ART

2.1 Introduction

"A forensics expert must have the investigative skills of a detective, the legal skills of a lawyer, and the computing skills of the criminal." [13]. This chapter outlines some of the digital network evidence acquisition, investigation software, and hardware tools commonly used by forensic investigators in law enforcement and private investigations such as ForNet, Wireshark, Security Incident and Event Management Software (SIEM), Network Forensic Analysis Tools (NFAT), and Deep-Packet Inspection (DPI). Current commercial, research and open-source tools are discussed specifying their benefits and designs. Common digital evidence storage formats are also discussed, outlining the cross-compatibility between the tools available and the associated formats. Best practices associated with the field of digital forensics from a technical, cryptographical and legal perspective are also discussed.

2.2 Computer Forensic Investigation

Generally speaking, the goal of a digital forensic investigation is to identify digital evidence relative to a specific cybercrime. Investigations rarely rely entirely on digital evidence to prosecute the offender, instead relying on a case built from physical evidence, digital evidence, witness testimony and cross-examination. However, when dealing solely with digital evidence, there are three major phases [14]:

1. *Acquisition Phase* – The acquisition phase is concerned with capturing the state of a digital system for later analysis. This is similar to the collection of physical evidence from a crime scene, e.g., taking photographs, collecting fingerprints, fibres, blood samples, tire patterns, etc. During this phase in a digital investigation, it is typically very difficult to tell which evidence is relevant to the case, so the goal of this phase is to collect all possible digital evidence (including any data on removable storage devices, network traffic, logs, etc.).

2. *Analysis Phase* – After a successful and complete acquisition of the system state from a suspect computer, the data acquired needs to be analysed to identify pieces of evidence. The analysis of evidence is carried out on an exact copy of the original evidence. This copy is verified against the original through the use of a hashing algorithm, as outlined in more detail in Section 2.7. Carrier [14] defines three major categories of evidence a digital investigator needs to discover when conducting his analysis:

    - Inculpatory Evidence – This is any evidence which supports a given theory.
    - Exculpatory Evidence – This is any evidence which contradicts a given theory.
    - Evidence of Tampering – This is any evidence which cannot be related to any theory currently under investigation, but shows that

the system was tampered with to avoid identification.

The procedure followed during this phase includes examining file and directory contents (including recovered deleted content) to draw verifiable conclusions based on any evidence that is collected.

3. *Presentation Phase* – The steps performed in the previous two phases are the same regardless of the type of investigation being conducted, e.g., corporate, law enforcement or military. However, the presentation phase will be different depending on corporate policy or local law. This phase presents the conclusions and their corresponding evidence that the digital investigator has deduced. In a court settings, the lawyers must first evaluate the evidence to confirm that it is court admissible.

2.2.1 Network Forensic Investigation

The 2006 National Institute of Standards and Technology's (NIST) special publication "Guide to Integrating Forensic Techniques Into Incident Response" [15] outlines a number of best practices and legal considerations for forensic investigators working with network data. The NIST guide outlines the typical sources of network evidence and tools that should be used during the evidence collection phase of an investigation:

- Firewall and router logs – These devices are normally configured to record suspicious activity.
- Packet Sniffing – This allows the investigator to monitor, in real-time, the activity on the network.
- Intrusion Detection Systems (IDS) – Some larger networks may employ IDS to capture packets related to suspect activity.
- Remote Access Servers – this includes devices such as VPN gateways and modem servers that facilitate connections between networks.

- Security Event Management Software – These tools aid in analysis of logs files, typically produced by IDS tools, firewalls, and routers.
- Network Forensic Analysis Tools – These tools allow a reconstruction of events by visualising and replaying network traffic within a specified period.
- Other Sources – These include Internet Service Provider (ISP) records, client/server applications, hosts' network configureration and connections, and Dynamic Host Configuration Protocol (DHCP) records.

A number of tools capable of collecting and analysing some of the above evidence are outlined in Section 2.3.

2.3 Network Investigation Tools

While the area of Computer Forensics and Cybercrime Investigation is relatively new among the more traditional computer security models, there is a small number of companies and open-source tools dedicated to forensic investigations. There are numerous free packet sniffing software tools available. A number of these tools are discussed in the following subsections:

2.3.1 TCPDump/WinDump

TCPDump and WinDump are the Unix and Windows equivalent command line network software analysers developed in the 1990s. The tools run on a local machine and are capable of capturing all the network traffic over ethernet or wireless connections. They have the ability to display in a semi-coherent fashion the captured traffic frame by frame and allow the analysis of the data. As its name might suggest, TCPDump focuses mainly on the TCP/IP protocol [16]. An example capture of an SSH session using WinDump can be seen in Figure 2.1.

```
WinDump.exe: listening on \Device\NPF_{479E2967-ACC3-4CC2-BF14-E3F1410671B4}
11:05:27.454643 IP (tos 0x0, ttl 128, id 22254, offset 0, flags [DF], proto: TCP
 (6), length: 52) Mark-PC.ucd.ie.53821 > scanlon.ucd.ie.22: S, cksum 0x9468 (cor
rect), 714757140:714757140(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
11:05:27.455417 IP (tos 0x0, ttl  63, id 0, offset 0, flags [DF], proto: TCP (6)
, length: 52) scanlon.ucd.ie.22 > Mark-PC.ucd.ie.53821: S, cksum 0x9c70 (correct
), 3719570269:3719570269(0) ack 714757141 win 5840 <mss 1460,nop,nop,sackOK,nop,
wscale 7>
11:05:27.455450 IP (tos 0x0, ttl 128, id 22255, offset 0, flags [DF], proto: TCP
 (6), length: 40) Mark-PC.ucd.ie.53821 > scanlon.ucd.ie.22: ., cksum 0xb3e9 (cor
rect), ack 1 win 16425
```

Figure 2.1: Example Frame Capture of SSH Session Using WinDump.

2.3.2 Ethereal

Ethereal is another free tool available for both Unix and Windows. It is more user friendly than TCPDump as it has a graphical user interface (GUI) to assist its users. Ethereal also provides a large number of protocol decoding options; more than 400 in total [16]. It allows the forensic investigator to analyse data collected on a packet basis or protocol basis. An example capture of an SSH session using Ethereal and its presentation in the GUI can be seen in Figure 2.2.

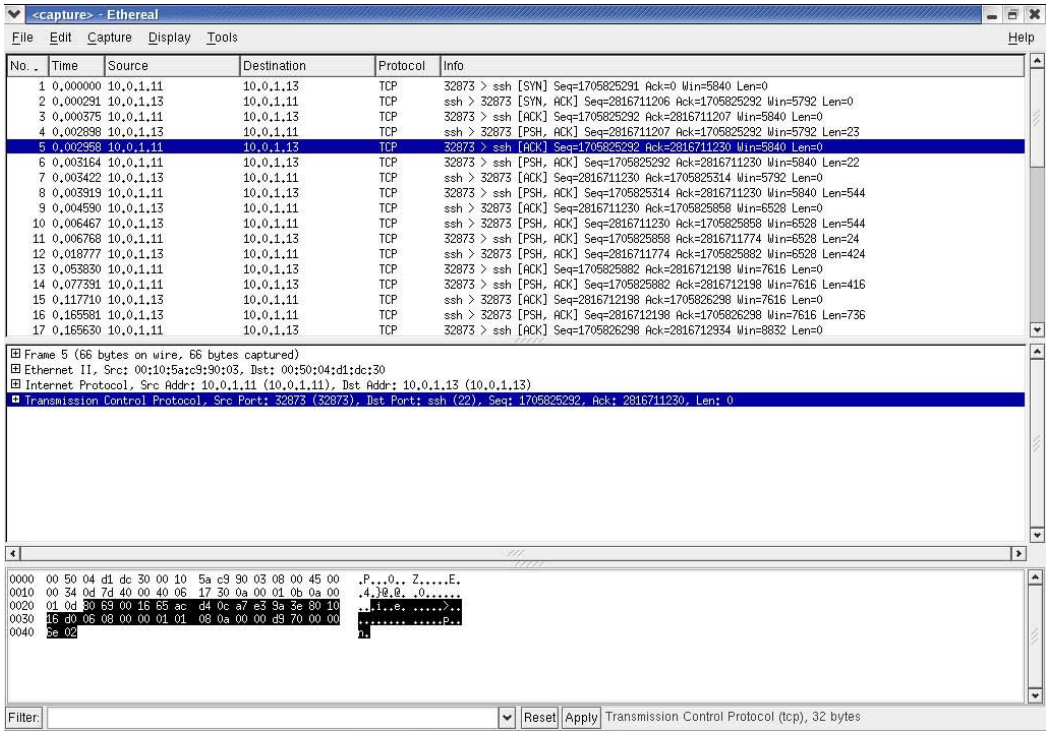

Figure 2.2: Example Frame Capture of SSH Session Using Ethereal.

2.3.3 Network Forensic Analysis Tools

NFATs are intelligent packet analysis tools capable of identifying firewall circumvention [17]. For example, corporate firewalls may block access to their staff from using instant messaging at work. Yahoo Messenger normally operates on port 5050, but when this port is blocked it will automatically switch to port 23 (usually reserved for telnet) [18]. While this port change might bypass a firewall rule in place, an NFAT would still be able to identify the network usage as being Yahoo Messenger due to packet analysis. NFATs are not designed as a replacement for firewalls or IDS software, but are designed to work in conjunction with them. Typically NFATs will rely on another piece of software to capture the traffic, e.g., TCPDump.

2.3.4 Security Incident and Event Manager Software

SIEM software is a combination of the formally different software categories of Security Incident Management Software and Security Event Management Software and takes a different investigative approach to the "on-the-fly" analysis tools outlined above. SIEM software is focused on importing security event information for a number of network traffic related sources, e.g., IDS logs, firewall logs, etc. [15]. It operates on an "after the fact" basis whereby it analyses copies of the logs attempting to identify suspicious network activity events by matching IP addresses, timestamps and other network traffic characteristics. An open source example of this software is called OSSIM [19].

2.4 Packet Inspection Hardware

In the regular operation of Network Interface Cards (NICs), the devices only accept incoming packets that are specifically addressed to its IP address. However when a NIC is placed in promiscuous mode, it will accept all packets that it sees, regardless of their intended destinations. Packet sniffing

hardware generally operates on this principle, with configuration available to capture all packets or only those with specific characteristics, e.g., certain TCP ports, certain source or destination IP addresses, etc. [15]. This style of network traffic capture can be used in combination with software sampling optimisation techniques in order to reduce the overall size of the data to be investigated [20].

The current standard hardware device used for digital evidence acquisition in the forensic laboratory is the Forensic Recovery of Evidence Device (FRED). This machine incorporates a selection of equipment tailored for digital investigations available from Digital Intelligence [21]. Each FRED workstation contains a collection of write-blocked (read-only) ports including Serial Advance Technology Attachment (SATA), Integrated Drive Electronics (IDE), Small Computer System Interface (SCSI), Universal Serial Bus (USB) and FireWire. However in order to perform network evidence capture, the workstation incorporates a standard 10/100/1000Mb ethernet card due to the requirement for any NIC to both send and receive packets. This NIC is capable of collecting network evidence when used in conjunction with one of the software tools outlined above.

2.5 Evidence Storage Formats

There is currently no universal standard for the format that digital evidence and any case related information is stored. This is due to the fact that there are no state or international governmental policies to outline a universal format. Many of the vendors developing forensic tools have their own proprietary evidence storage format. With such a small target market (mainly law enforcement), it sometimes makes business sense for them to try to lock their customers into a proprietary format. This results in their users being more likely to buy only their software in the future as it will be compatible with their existing evidence. There have been a number of attempts at creating open formats to store evidence and its related metadata. The following

subsections describe the most common evidence storage formats.

2.5.1 Common Digital Evidence Storage Format

The Common Digital Evidence Storage Format (CDESF) Working Group was created as part of the Digital Forensic Research Workshop (DRFWS) in 2006. The goal of this group was to create an open data format for storing digital forensic evidence and its associated metadata from multiple sources, e.g., computer hard drives, mobile Internet devices, etc. [22]. The format which the CDESF working group were attempting to create would have specified metadata capable of storing case-specific information such as case number, digital photographs of any physical evidence and the name of the digital investigator conducting the investigation. In 2006, the working group produced a paper outlining the advantages and disadvantages of various evidence storage formats [23].

Due to resource restrictions, the CDESF working group was disbanded in 2007 before accomplishing their initial goal.

2.5.2 Raw Format

According to the CDESF Working Group, "the current de facto standard for storing information copied from a disk drive or memory stick is the so-called "raw" format: a sector-by-sector copy of the data on the device to a file" [24]. The raw format is so-called due to the fact that it is simply a file containing the exact sector-by-sector copy of the original evidence, e.g., files, hard disk/flash memory sectors, network packets, etc. Raw files are not compressed in any manner and as a result, any deleted or partially overwritten evidence that may lay in the slackspace of a hard disk is maintained. All of the commercial digital evidence capturing tools available today have the capability of creating raw files.

2.5.3 Advanced Forensic Format

The Advanced Forensic Format (AFF) is an open source, extensible format created by S. Garfinkel in Basis Technology in 2006 [25]. The AFF format has a major emphasis on efficiency and as a result it is partitioned into two layers; the disk representation layer which defines segment name used for storing all data associated with an image and the data storage layer which defines how the image is stored (binary or XML) [26]. The format specifies three file extensions; *.aff, *.afd and *.afm. *.aff files store all data and metadata in a single file, *.afd files store the data and metadata in multiple small files, and *.afm files store the data in a raw format and the metadata is stored in a separate XML file [26].

2.5.4 Generic Forensic Zip

Generic Forensic Zip (gfzip) is an open source project. Its goal is to create a forensically sound compressed digital evidence format based on AFF 2.5.3 [27]. Due to the fact that it is based upon the AFF format, there is limited compatibility between the two in terms of segment based layout. One key advantage that gfzip has over the AFF format is that gfzip seeks to maintain compatibility with the raw format, as described in Section 2.5.2. It achieves this by allowing the raw data to be placed first in the compressed image [26].

2.5.5 Digital Evidence Bag (QinetiQ)

The method for traditional evidence acquisition involves a law enforcement officer collecting any relevant items at the crimescene and storing the evidence in bags and seals. These evidence bags may then be tagged with any relevant case specific information, such as [28]:

- Investigating Agency / Police Force
- Exhibit reference number

- Property reference number
- Case/Suspect name
- Brief description of the item
- Date and time the item was seized/produced
- Location of where the item was seized/produced
- Name of the person that is producing the item as evidence
- Signature of the person that is producing the item
- Incident/Crime reference number
- Laboratory reference number

Physical evidence containers, such as evidence bags, are trusted due to the well understood and practised process called "chain of custody" [29].

Digital Evidence Bag (DEB) is a digital version of the traditional evidence bag, created by Philip Turner in 2005 [28]. DEB is based on an adaptation of existing storage formats, with potentially infinite capacity. The data stored in a DEB is stored in multiple files, along with metadata containing the information that would traditionally be written outside on an evidence bag. There are currently no tools released that are compatible with the QinetiQ DEB format.

2.5.6 Digital Evidence Bag (WetStone Technologies)

In 2006, C. Hosmer, from WetStone Technologies Inc. [30], published a paper outlining the design of a Digital Evidence Bag (DEB) format for storing digital evidence [29]. This format is independent from the Digital Evidence Bag outlined in Section 2.5.5. The format emerged from a research project funded by the U.S. Air Force Research Laboratory. The motivation for this format was similar to that described in Section 2.5.5, i.e., to metaphorically mimic the plastic evidence bag used by crime scene investigators to collect physical

evidence such as fibres, hairs, etc. This format will be released publicly when complete.

2.5.7 EnCase Format

The EnCase format for storing digital forensic is proprietary to the evidence analysis tool of the same name [31]. It is by far the most common evidence storage option used by law enforcement and private digital investigation companies [26]. Because of the proprietary nature of the format, along with the lack of any open formal specification from Guidance Software [32], much remains unknown about the format itself. Some competitors to Guidance Software have attempted to reverse engineer the format to provide an element of cross-compatibility with their tools [25]. EnCase stores a disk image as a series of unique compressed pages. Each page can be individually retrieved and decompressed in the investigative computer's memory as needed, allowing a somewhat random access to the contents of the image file. The EnCase format also has the ability to store metadata such as a case number and an investigator [25].

2.6 Evidence Handling

When analysing physical evidence, the commonly used procedure is known as the "chain of custody" [28]. The chain of custody commences at the crime scene where the evidence is collected, when the investigating officer collects any evidence s/he finds and places it into an evidence bag. This evidence bag will be sealed to avoid any contamination from external sources and signed by the officer and will detail some facts about the evidence, e.g., description of evidence, location, date and time it was found etc. The chain of custody will then be updated again when the evidence is checked into the evidence store. When it comes to analysing the evidence, it will be checked out to the analysts' custody and any modification to the evidence required to facilitate

the investigation, e.g., taking a sample from a collected fibre to determine its origin or unique properties. Each interaction with the evidence will be logged and documented.

The procedures outlined above for physical evidence need to be slightly modified for evidence acquisition and analysis. Due to the fact that digital evidence is analysed on forensic workstations, most of the above sequences can be automated into concise logging of all interactions. During a digital investigation, there is no requirement to modify the existing evidence in any way. This is because all analysis is conducted on an image of the original source and any discovered evidence can be extracted from this image, documented and stored separately to both the original source and the copied image. It is imperative when dealing with all types of evidence that all procedures used are reliable, reproducible and verifiable. In order for evidence to be court admissible, it must pass the legal criteria for the locality that the court case is being heard, as outlined in greater detail in Section 2.8.

2.6.1 What does "Forensically Sound" really mean?

Many of the specifications for digital forensic acquisition and analysis tools, storage formats and hash functions state that the product in question is "forensically sound" or that the product works with the digital evidence in a "forensically sound manner", without specifying exactly what the term means. In 2007, E. Casey published a paper in the Digital Investigation Journal entitled "What does "forensically sound" really mean ?" [33].

In the paper, Casey outlined some of the common views of forensic professionals with regard to dealing with digital forensic evidence. Purists state that any digital forensic tools should not alter the original evidence in any way. Others point out that the act of preserving certain types of evidence necessarily alters the original, e.g., a live memory evidence acquisition tool must be loaded into memory (altering the state of the volatile memory and possibly overwriting some latent evidence) in order to run the tool and capture

any evidence contained in the memory. Casey then goes onto to explain how some traditional forensic process require the alteration of some of the evidence in order to collect the required information. For example, collecting DNA evidence requires taking a sample from some collected evidence, e.g., a hair. Subsequently, the forensic analysis of this evidentiary sample (DNA profiling) is destructive in its nature which further alters the original evidence.

Casey summarises that from a forensic standpoint, evidence acquisition and handling should modify the evidence as little as possible and when modification is unavoidable, it should be well documented and considered in the final analytical results. "Provided the acquisition process preserves a complete and accurate representation of the original data, and its authenticity and integrity can be validated, it is generally considered forensically sound" [33].

2.7 Cryptographic Hash Functions

Cryptographic hash functions are deterministic procedures which operate by taking a block of data or a file as input and output a fixed length digital fingerprint or cryptographic hash value/sum. The data input to a hash function is commonly referred to as the "message", while the hash sum produced is referred to as the digest.

The ideal collision resistant cryptographic hash function (h) has four main properties, defined by B. Preneel as part of his Ph.D. thesis in 1993 [34]:

1. The description of h must be publicly known and should not require any secret information for its operation.

2. The argument/message X can be of arbitrary length and the result h(X) has a fixed length of n bits (with $n \geq 128$).

3. Given h and X, the computation of h(X) must be "easy".

4. The hash function must be "one-way" in the sense that given a Y, it

is infeasible to find a message X such that h(X) = Y , i.e., it should be impractical to modify a message without changing its hash. It should also be infeasible given X and h(X) to find a message X' $\neq$ X such that h(X') = h(X), i.e., it should not be possible to have two different messages with the same hash.

5. The hash function must be collision resistant: this means that one should not find two distinct messages that hash to the same result. It also should not be feasible to find a message X that has a given hash sum h(X).

2.7.1 Collision Resistance

The measure of the unlikelihood of two different inputs to a hashing function returning the same hash sum is known as the collision resistance of the hash function. Generally speaking, the larger the internal state size that the hashing function has to operate with, the better the collision resistance of that function.

In 2005, Wang and Yu published a paper outlining their attempts to break a number of specified hash functions, entitled "How to Break MD5 and Other Hash Functions" [35]. In this paper they described a method for engineering two files which, when hashed using MD5, would result in having the same hash sum. In their experiments, they created two different files, F1 and F2, by reverse engineering them to have the specific bits in the specific file locations required for the hashing function to produce an identical hash sum so far. It is important to note that there is no documented evidence that, if given a specific file F1, that anyone is capable of engineering a second file F2 that has the same hash sum. As a result of this paper, the United States Computer Emergency Readiness Team (US-CERT), part of the United States' Department of Homeland Security, published a vulnerability note stating that MD5 should be considered cryptographically broken and unsuitable for further use and that most United States governmental applications will be required to move to the SHA-2 family of hashing functions by 2010 [36].

To date, no collisions have been found in any of the SHA-2 family of hashing

functions.

2.7.2 Avalanche Effect

The avalanche effect of a cryptographic hashing function refers to a desirable property whereby should the input file be modified slightly [37], e.g., changing a single bit of the file, the resultant hash sum produced changes significantly. The term "avalanche effect" used to describe this property was created by H. Feistel in 1975 [38]. Table 2.1 shows a sample set of common hashing functions along with sample hash sums they produce for two slightly different input files showing the influence the avalanche effect has on each function.

2.7.3 Overview of Common Hashing Algorithms

While there are hundreds, if not thousands, of hashing functions in existence, the list of commonly used functions is significantly shorter. This is due to the fact that NIST and the National Security Agency (NSA) in the United States have prioritised the standardisation of hashing functions. The most popular hashing functions, outlined below, are all based on the message digest principle. The message digest principle was designed by Ronald Rivest [39] and constitutes a hash function taking in a message of arbitrary length and producing a fixed length message digest (hash value/sum) based on that input.

2.7.3.1 MD Family

The Message Digest (MD) algorithm family of hash functions were all created by Ronald Rivest, a professor in Massachusetts Institute of Technology, along with some collaboration from others. The family contains six iterations of the algorithms; MD, MD2 (1988), MD3 (1989), MD4(1990), MD5 (1991) and MD6 (2008.) From the original iteration up as far as MD5, the algorithms all produced 128-bit message digests. These MD hash values are expressed as 32

Hash Algorithm	**Length in bits**	***Sentence 1:* The quick brown fox jumps over the lazy dog**	***Sentence 2:* The quick brown fox jumps over the lazy cog**	**Diff %**
Adler32	32	5BDC0FDA	5BD90FD9	25.0%
CRC32	32	414FA339	4400B5BC	87.5%
Haval	128	713502673D67E5FA 557629A71D331945	4C9409BE8321D982 72D9252F610FBB5B	93.8%
MD2	128	03D85A0D629D2C44 2E987525319FC471	6B890C9292668CDB BFDA00A4EBF31F05	93.8%
MD4	128	1BEE69A46BA81118 5C194762ABAEAE90	B86E130CE7028DA5 9E672D56AD0113DF	93.8%
MD5	128	9E107D9D372BB682 6BD81D3542A419D6	1055D3E698D289F2 AF8663725127BD4B	100%
RipeMD128	128	3FA9B57F053C053F BE2735B2380DB596	3807AAAEC58FE336 733FA55ED13259D9	93.8%
RipeMD160	160	37F332F68DB77BD9 D7EDD4969571AD67 1CF9DD3B	132072DF69093383 5EB8B6AD0B77E7B6 F14ACAD7	95.0%
SHA-1	160	2FD4E1C67A2D28FC ED849EE1BB76E739 1B93EB12	DE9F2C7FD25E1B3A FAD3E85A0BD17D9B 100DB4B3	95.0%
SHA-256	256	D7A8FBB307D78094 69CA9ABCB0082E4F 8D5651E46D3CDB76 2D02D0BF37C9E592	E4C4D8F3BF76B692 DE791A173E053211 50F7A345B46484FE 427F6ACC7ECC81BE	95.3%
SHA-384	384	CA737F1014A48F4C 0B6DD43CB177B0AF D9E5169367544C49 4011E3317DBF9A50 9CB1E5DC1E85A941 BBEE3D7F2AFBC9B1	098CEA620B0978CA A5F0BEFBA6DDCF22 764BEA977E1C70B3 483EDFDF1DE25F4B 40D6CEA3CADF00F8 09D422FEB1F0161B	95.8%
SHA-512	512	07E547D9586F6A73 F73FBAC0435ED769 51218FB7D0C8D788 A309D785436BBB64 2E93A252A954F239 12547D1E8A3B5ED6 E1BFD7097821233F A0538F3DB854FEE6	3EEEE1D0E11733EF 152A6C29503B3AE2 0C4F1F3CDA4CB26F 1BC1A41F91C7FE4A B3BD86494049E201 C4BD5155F31ECB7A 3C8606843C4CC8DF CAB7DA11C8AE5045	96.1%

Table 2.1: Example hash sums for a small file containing the sentences outlined. The percentage difference shows the difference in the hash sums produced. While each character of a hash is hexadecimal, i.e., 1 of 16 possible values, it is notable that some hashing functions have differences greater than the expected maximum difference, i.e., >93.8%. This is due to a more pronounced avalanche effect in the hashing function.

hexadecimal digits, as can be seen in Table 2.1. MD6 is based on a variable length message digest size to improve performance for smaller inputs, and as a result the message digest can be anywhere in the range from 0 - 512 bits in length.

MD5 is a popular hash function used in numerous applications. Most of the tools available to the digital investigator rely on a combination of the CRC32 and the MD5 hash functions for maintaining data integrity [23].

MD6 was entered into the competition for the SHA-3 Family of hash functions. However, in July 2009, the algorithm was withdrawn from the competition because in order for it to be fast enough to compete, the design would have had to compromise its resistance to differential attacks.

2.7.3.2 SHA-0 and SHA-1 Family

The first specification of the Secure Hashing Algorithm (SHA) family of hashing functions was published in 1993 by the US National Institute for Standards and Technology. This early specification is now known as the SHA-0 function. SHA-0 was withdrawn from use by the US National Security Agency in 1995 and was replaced by a modified version of the function; SHA-1. Both SHA-0 and SHA-1 produce 160-bit hash sums and they have a maximum input message size of $2^{64} - 1$ bits (or 2048 petabytes).

X. Wang, Y.L. Yin and H. Yu produced a paper entitled "Finding Collisions in the Full SHA-1" in 2005 [40]. This paper outlined the first attack on the SHA-1 hash function. The authors successfully found collisions on the SHA-1 function. They achieved this by first finding near-collisions. They then were able to discover full collisions based on the analysis of the near collisions. They conclude that although the SHA-1 family of hash functions has message expansion, it does not offer enough avalanche effect in terms of differing inputs.

2.7.3.3 SHA-2 Family

The SHA-2 Family consists of the following hash functions: SHA-224, SHA-256, SHA-384, and SHA-512. The number in the name of the hash function represents the output message digest size in bits. H. Gilbert and H. Handschuh produced a journal paper entitled "Security Analysis of SHA-256 and Sisters" in 2004 [41] which published their results from the analysis of the SHA-2 family of hash functions. They found that the attacks that have broken the SHA-1 family no longer are applicable to the SHA-2 family.

The SHA-224 and SHA-256 have the same maximum input file size of $2^{64} - 1$ bits (or 2048 petabytes) as with the SHA-1 Family, while the SHA-384 and SHA-512 have a maximum of $2^{128} - 1$ bits (or 3.78 x 10^{22} petabytes).

2.7.3.4 SHA-3 Family

NIST, part of the Department of Commerce, held a five year development competition to decide on which hashing function to choose for the third iteration of the SHA Family. As part of the competition, NIST accepted over 60 entries into the first round of testing. This number was reduced down to 14 accepted into the second round which was announced in August 2009 [42]. The remaining candidates in the second round are BLAKE [43], Blue Midnight Wish [44], CubeHash [45], ECHO [46], Fugue [47], Grøstl [48], Hamsi [49], JH [50], Keccak [51], Luffa [52], Shabal [53], SHAvite-3 [54], SIMD [55] and Skein [56]. The winner of the hashing function, Keccak, was announced in November 2012 after evaluation of the final round entries [57]. Keccak uses a "sponge construction" with no explicit maximum limit for file size and for produces a variable length hash.

2.8 Court Admissible Evidence

Since the United States leads the way with the implementation of many standards in relation to evidence handling and the court admissibility of evidence, many other countries look to the procedures outlined by the United States in this area when attempting to create their own legal procedures [58]. As a result, much of the information available regarding the admissibility of digital forensic evidence into court cases is specifically tailored to the Unites States, but will influence law makers across the globe. Carrier [14] states that in order for evidence to be admissible into a United States legal proceeding, the scientific evidence (a category which digital forensic evidence falls under in the U.S.) must pass the so-called "Daubert Test" (see Section 2.8.1 below). The reliability of the evidence is determined by the judge in a pre-trail "Daubert Hearing". The judge's responsibility in the Daubert Hearing is to determine whether the methodologies and techniques used to identify the evidence was sound, and as a result, whether the evidence is reliable.

2.8.1 Daubert Test

The "Daubert Test" stems from the United States Supreme Court's ruling in the case of Daubert vs. Merrell Dow Pharmaceuticals (1993) [59]. The Daubert process outlines four general categories that are used as guidelines by the judge when assessing the procedure(s) followed when handling the evidence during the acquisition, analysis and reporting phases of the investigation, [14] and [59]:

1. *Testing* – Has the procedure been tested? Testing of any procedure should include testing of the number of false negatives, e.g., if the tool displays filenames in a given directory, then all file names must be shown. It should also incorporate testing of the number of false positives, e.g. if the tool was designed to capture digital evidence, and it reports that it was successful, then all forensic evidence must be exactly copied to

the destination. NIST have a dedicated group working on Computer Forensic Tool Testing (CFTT) [60].

2. *Error Rate* – Is there a known error rate of the procedure? For example, accessing data on a disk formatted in a documented file format, e.g., FAT32 or ext2, should have a very low error rate, with the only errors involved being programming errors on behalf of the developer. Acquiring evidence from an officially undocumented file format, e.g., NTFS, may result in unknown file access errors occurring, in addition to the potential programming error rate.

3. *Publication* – Has the procedure been published and subject to peer review? The main condition for evidence admission under the predecessor to the Daubert Test, the Frye Test, was that the procedure was documented in a public place and undergone a peer review process. This condition has been maintained in the Daubert Test [14]. In the area of digital forensics, there is only one major peer-reviewed journal, the International Journal of Digital Evidence.

4. *Acceptance* – Is the procedure generally accepted in the relevant scientific community? For this guideline to be assessed, published guidelines are required. Closed source tools have claimed their acceptance by citing the large number of users they have. The developers of these tools do not cite how many of their users are from the scientific community, or how many have the ability to scientifically assess the tool. However, having a tool with a large user base can only prove acceptance of the tool; it cannot prove the acceptance of the undocumented procedure followed when using the tool.

In 2005, The House of Commons Science and Technology Committee in the United Kingdom published a report entitled "Forensic Science on Trial" [58]. In this report they outline numerous standards to be used across the field of forensics. As part of this report, the admissibility of expert evidence is discussed. As it stood in the UK when the report was written, the judge

of any given case had the role of the "gate-keeper" for any evidence s/he would admit into his/her court. It was determined that judges are not well-placed to determine the scientific validity without input from scientists, especially due to the absence of an agreed protocol for assessment. The main recommendation to come from the report is that the Forensic Science Advisory Council should develop a "gate-keeping" test for expert evidence, built in partnership with judges, scientists and other key players from the criminal justice system and that it should be built upon the US Daubert Test [58].

2.9 Legal Considerations of Network Forensics

Collecting network traffic can pose legal issues. Deploying a packet sniffing or deep packet inspection device, such as those outlined above, can result in the (intentional or incidental) capture of information with privacy and security implications, such as passwords or e-mail content, etc. As privacy has become a greater concern for regular computer users and organisations, many have become less willing to cooperate or share any information with law enforcement. For example, most ISPs will now require a court order before providing any information related to suspicious activity on their networks [15]. In Europe, continental legal systems operate on the principle of free introduction and free evaluation of evidence and provide that all means of evidence, irrespective of the form they assume, can be admitted into legal proceedings [61].

One aspect of the use of search and seizure warrants in an Internet environment concerns the geographical scope of the warrant issued by a judge or a court authorising the access to the digital data. In the past, the use of computer-generated evidence in court has posed legal difficulties in common law countries, and especially in Australia, Canada, the United Kingdom and the USA. The countries are characterised by an oral and adversarial procedure. Knowledge from secondary sources is regarded as "hearsay evidence", such as other persons, books, records, etc., and in

principle is inadmissible. However, digital evidence has become widely admissible due to several exceptions to this hearsay rule [61].

2.10 Summary

This chapter describes some foundations behind the system described in Chapter 5. It outlined some of the tools, formats, tests and procedures used for the acquisition and analysis of digital forensic evidence. This chapter also outlined some network focused forensic tools and systems developed for aiding digital forensic investigations. Traditionally, in order for a digital forensic investigation to begin, the investigator must physically visit the crime scene and collect any suspect computer equipment. This equipment will then be brought back to the forensic laboratory. When investigating network crimes, the procedure is somewhat different. The forensic investigator may need to install a physical deep packet inspection device onto the suspect's Internet connection (assuming a warrant is granted to do so). This will then typically be left in situ for a predetermined amount of time and then taken away for analysis. This DPI device will generally contain all of the suspects network traffic for the investigation duration. Analysis, and subsequent detection of any incriminating evidence, can only begin at this stage. As a result of this offline analysis, it may be some time before an arrest can be made.

CHAPTER

THREE

PEER-TO-PEER FILE-SHARING

3.1 Introduction

P2P networks can be used in a number of ways including distributed computing, collaboration and communication, but perhaps they are best known for their use in file-sharing [62]. In 1999, three influential P2P systems were launched attracting significant interest in the Internet technology; the Napster music sharing system, the Freenet anonymous data store and the SETI@home distributed volunteer-based scientific computing project [63].

In 2008, Cisco estimated that P2P file sharing accounted for 3,384 petabytes per month of global Internet traffic, or 55.6% of the total usage. Cisco forecast that P2P traffic will account for 9,629 petabytes per month globally in 2013 (approximately 30% of total global usage) [64]. While the volume of P2P traffic is set to almost triple from 2008-2013, its proportion of total Internet traffic is set to decrease due to the rising popularity of media streaming sites and one-click file hosting sites (often referred to as "cyberlockers") such as Rapidshare, Mega, Mediafire, etc. Cisco estimate that P2P file transfer will decline over the next few years to 5,755 petabytes per month by 2017 [65]. The decline is accounted for due to the rise in streaming services and traditional server based file-sharing. BitTorrent is the most popular P2P protocol used worldwide and accounts for the biggest proportion of Internet traffic when compared to other P2P protocols. The most recent measurement data from

Ipoque GmbH. has measured BitTorrent traffic to account for anything from 20-70% of total Internet usage in 2009, depending on the specific geographical area concerned [66]. With the evolution towards employing encrypted traffic, these measurement statistics have been upwardly estimated over the measured traffic.

3.1.1 Financial Impact on Content Producing Industry

The content producing industries report that revenue figures are steadily declining as a result of online piracy. The International Federation of the Phonographic Industry's (IFPI) Digital Music Report 2011 states that legitimate digital music distribution is up 1000% from 2004 to 2010, although total global recorded music revenues are down 31% over the same period [67]. The report cites Internet piracy as having a significant impact on their sales. The report cites a study from 2010 entitled "Piracy, Music and Movies: A Natural Experiment" which estimates that physical sales would be up 72% with the abolishment of piracy in Sweden [68].

The 2012 Digital Music report states that 28% of Internet users are accessing at least one unlicensed site monthly and that approximately half of those users are using P2P networks [69]. In 2006, Zentner [70] summarises that downloading MP3 files online reduces the probability of buying music by 30%. In 2008, the Motion Picture Association of America (MPAA) reported that Internet piracy cost the film industry $7 billion that year [71].

While the figures outlined above are provided by the content producing industry, the figures of total physical and digital sales in comparison to illegal downloads are not available or provided by the industry for independent verification. However, unauthorised distribution of copyrighted content must have an impact on the profits of the industry as a whole. As a result of these financial losses incurred by the content producing industry, there has been a significant push for technological and legislative measures to deter users from choosing the pirated option. A number of these measures are outlined in

Sections 3.2 and 3.6.

Research has been conducted into the decision of an average consumer to legally purchase digital content versus the decision to illegally download the content. In his 2005 paper, Fetscherin evaluated the choices made by consumers in legally or illegally downloading movies on the Kazaa P2P network [72]. He found that the majority of users prefer to download movies legally, but that a significant number will always opt for the free option. The factors affecting consumer behaviour in this legal/illegal choice are, in order, the risk of being caught, the price, the perceived value of the original and the availability of high quality copies.

In 2013, the European Commission's Institute for Prospective Technological Studies published a working paper analysing the impact of illegal downloading and legal streaming on the legal purchases of digital music [73]. The results were based on over 16,000 European consumers from all European Commission (EC) countries. It was found that Internet users do not view illegal downloading as a substitute for legal digital music. An increase of 10% of clicks on illegal downloading websites was found to lead to a 0.2% increase in clicks on legal purchase websites.

3.2 Legislative Response to Online Piracy

While not merely limited to P2P file-sharing, there has been significant effort globally to create new legislative measures to combat online copyright infringement. A number of these provisions are outlined below [74]:

1. Stop Online Piracy Act (SOPA) – This United States act states that if a website is deemed to be dedicated to the theft of U.S. property then it should be blocked by various Internet companies. These include ISPs, search engines, payment providers and advertising services. Each must prevent access to the site for their customers and cease operation with the site and its owners. This act has facilitated the significant seizure of

Internet domain names by the U.S. government since 2012.

2. PROTECT IP Act (PIPA) – In the United States, the "Preventing Real Online Threats to Economic Creativity and Theft of Intellectual Property Act of 2011" allows the Attorney General to sue operators of Internet sites dedicated to infringing activities.

3. Anti-Counterfeiting Trade Agreement (ACTA) – This is a treaty signed between the United States, Australia, Canada, Korea, Japan, New Zealand, Morocco, and Singapore. The treaty specifies that laws should be created to make those responsible for copyright infringement on a commercial scale criminally liable.

4. Trans-Pacific Partnership Agreement (TPP) – This is a trade agreement negotiated between the United States, Australia, Brunei, Chile, Malaysia, New Zealand, Peru, Singapore, and Vietnam. This broadly defined agreement states that its member countries must legislate for significant wilful copyright infringement for financial gain.

5. Graduated Response – France, New Zealand, South Korea, Taiwan and the United Kingdom have implemented various "three strikes" laws, with Ireland and the United States having a voluntary system put in place [71]. The model requires the rights holders to monitor for unauthorised online infringing activity and reporting the corresponding IP addresses to the ISP involved. The ISP can identify the customer and send them a notification. Repeat infringers risk bandwidth throttling, protocol blocking or account suspension.

As deduced by Carrier [74], the language used in the existing treaties, acts and agreements outlined above is quite vague. The resulting lack of clarity enables biased interpretations, which in turn promotes a litigation based business model. Carrier claims this will ultimately stem innovation. While it's clear that there is a need for legislation to protect content producers, the current iterations leave much open to various interpretation.

3.3 Peer-to-Peer File-sharing System Design

When designing and implementing a P2P file-sharing system, developers must make a number of key decisions regarding the network and its purposes:

1. Centralised/Decentralised/Hybrid – A centralised network can offer a simpler design whereas a decentralised design can offer a more robust network. Hybrid networks are much more complex to implement but can offer many of the advantages of both centralised and decentralised systems.

2. Open Source/Proprietary – Making the network design and client open source promotes an active development community but may result in compatibility issues across numerous clients with different update cycles resulting in newer features taking some time to roll out across all clients.

3. Encrypted/Unencrypted – Encrypting all network communication can help eliminate some of the packet sniffing investigation methods deployed by IT administrators and investigators but can decrease the performance of the overall system.

3.3.1 Centralised Design

In a centralised P2P network design, there are one or more central servers which puts users in contact with each other. When a new user wishes to join the network, s/he registers with a known server which, in turn, is able to supply the user with a list of other known active peers currently on the network. Depending on the specific centralised design, the server itself may index the entire system, i.e., maintain an active list of users and the content they are sharing, or help contribute to a distributed hash table (DHT) maintained by the server. The latter option passes much of the querying load onto the connected peers. Hashing is used to prevent the accidental

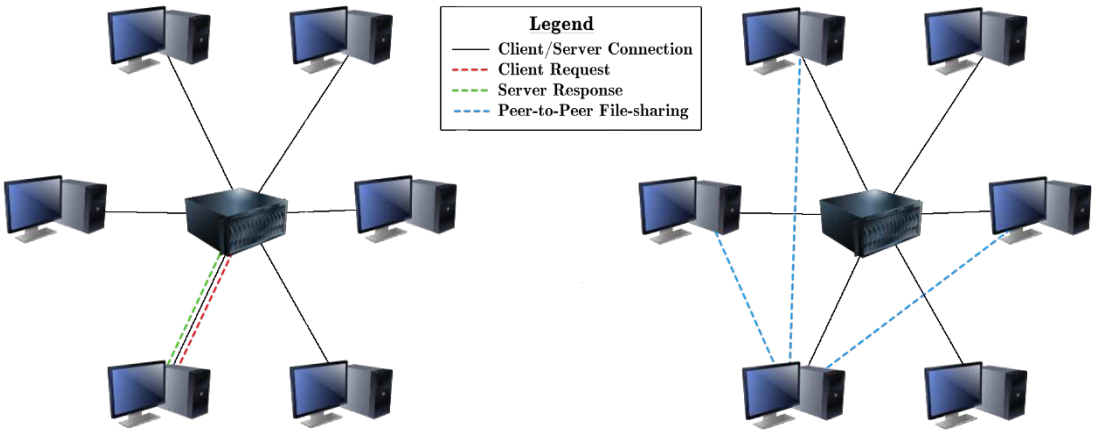

Figure 3.1: Centralised P2P system overview.

downloading of incorrect content, e.g., two files on the network with the same name but with different content.

A sample centralised usage scenario is shown in Figure 3.1, whereby the server records each user's shared files. On the left hand side of the figure, the P2P client issues a request to the server for any users sharing a specific piece of content. The server responds with a list (IP addresses and port numbers) of active nodes on the network sharing that content. Once the user chooses one of these files to download, the server no longer has any further part in the interaction. The user's client software will connect directly to the remote peer and download the file directly. In most modern systems built using this design, the user may download part of the desired file from multiple other peers simultaneously. This is shown on the right hand side of the figure and results in a faster throughput of the download by distributing the workload. Once all the required parts are complete, the file is combined into the original content and is immediately available to the user. By default, many P2P systems automatically make any newly downloaded files available on the network for other peers.

The advantage of a centralised design lies in its efficiency of conducting queries and the resultant small traffic footprint devoted to querying. However, the most significant downside to a centralised design is that there is a single point of failure. If the central server is disrupted or removed from the

network, the entire network ceases to function and is rendered useless.

3.3.2 Decentralised Design

Decentralised P2P network design removes the single point of failure from the centralised design outlined above. It achieves this by enforcing that each node in the network simultaneously plays the role of client and server. When a node receives a query request, that request is passed on to all known nodes and the results are passed back to the source of the query. Each node in the network maintains a small active list of current connections and helps to contribute to a DHT. To remove any loss to the DHT as a result of a node going offline, each distributed part will exist on multiple nodes.

The primary advantage of a decentralised system is that there is no single point of failure. Removing any single node from the network will have no significant impact on the entire system. However, due to its decentralised design, each query will take much longer to complete as the query needs to be passed directly from node to node before any response comes back to the source. As a result, the larger the network becomes, the longer the time required to conduct a complete search of the entire network will become. This results in a much larger querying traffic footprint in order to keep the network functional. There are a number of solutions to this problem, i.e., limiting the number of hops a query can be passed along, including a specified query timeout, etc. However, each of these solutions result in a partial search of the network.

A sample usage scenario is outlined in Figure 3.2 showing how each node acts as both a client and a server. The query can be seen passing from node to node until a "hit" is found. Depending on the design, this query hit might get passed back through the same sequence of steps the query took or alternatively may be relayed directly back to the querying node. The file transfer occurs in a similar direct fashion as exists in a centralised P2P network. When a new node wishes to join the network, it needs to bootstrap onto the DHT in some manner. Depending on the implementation of the network, there may be a

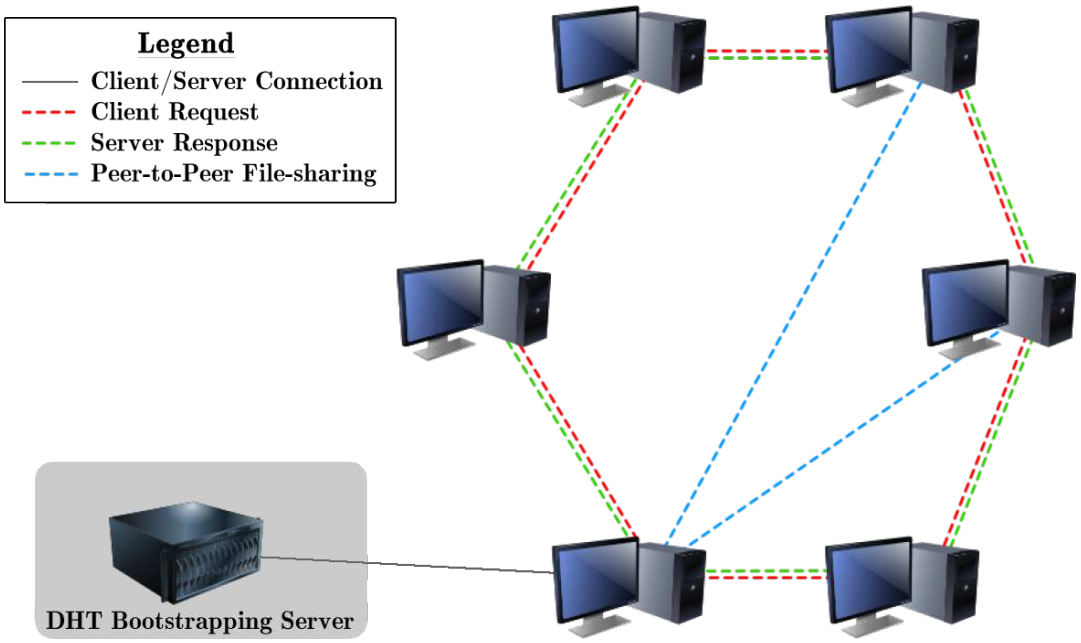

Figure 3.2: Decentralised P2P system overview.

hardcoded list of always-on DHT servers or a list of always-on nodes. These servers/nodes can provide a new client with a list of active nodes to bootstrap onto the network.

3.3.3 Hybrid Design

Hybrid P2P networks take on a number of the features from both centralised and decentralised networks in an attempt to overcome the limitations of each. This type of network will employ a large number of centralised servers to prevent the network becoming dysfunctional if an individual server should be taken offline. In practice, these servers are actually regular peers on the network. Static hybrid networks allow a peer to specify that they would like to become a server, or "supernode" in the configuration of the client software, e.g., in eDonkey. Dynamic hybrid network clients, e.g., Limewire, FastTrack and Gnutella, can automatically promote any peer to become a supernode dependant on specified criteria such as uptime, bandwidth capability, latency etc.

Querying a hybrid network can involve varying query distribution options.

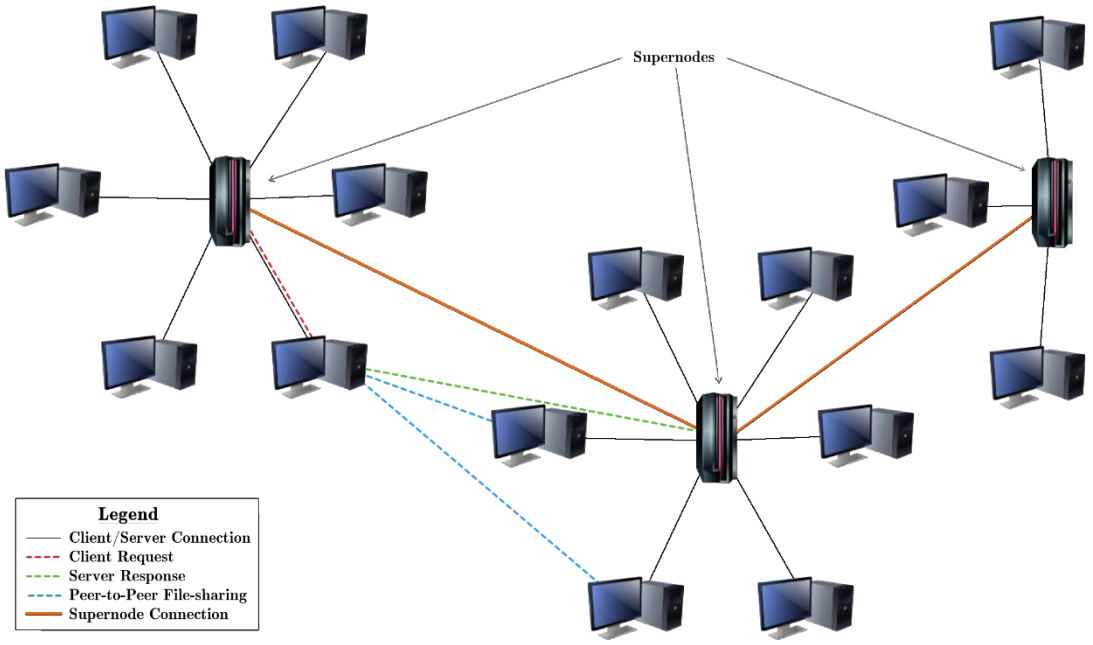

Figure 3.3: Hybrid P2P system overview.

There are two main query propagation mechanisms employed by hybrid P2P networks [75]:

1. Random Walk – Employing this mechanism, a file search query is sent randomly to known neighbours in the network. If this node can resolve the query, i.e., has a file matching the request, a query hit is sent back to the source of the request. If it does not resolve the query, it randomly passes this query onto another neighbour from its list and the process iterates until either a hit is returned or a timeout kills the query.

2. Expanding Ring – This query propagation option can be thought of as a sequence of flooding searches in which the time-to-live (TTL) is increased at each iteration. A simple flooding search is conducted whereby a query is sent to all known neighbours. Each neighbour will propagate the received query onto all of its active neighbours (it will only forward on each specific query once). Again, if any node can resolve the query, it will directly reply to the origin of the query with a hit. The query dies when the TTL reaches zero. This propagation mechanism is outlined in Figure 3.3, where a query is sent from a node to

a supernode. If this supernode cannot resolve the query, it is forwarded onto other supernodes and a query hit is returned directly to the source node. File transfer then commences directly between the nodes.

3.4 Peer-to-Peer File-sharing Networks

3.4.1 Napster

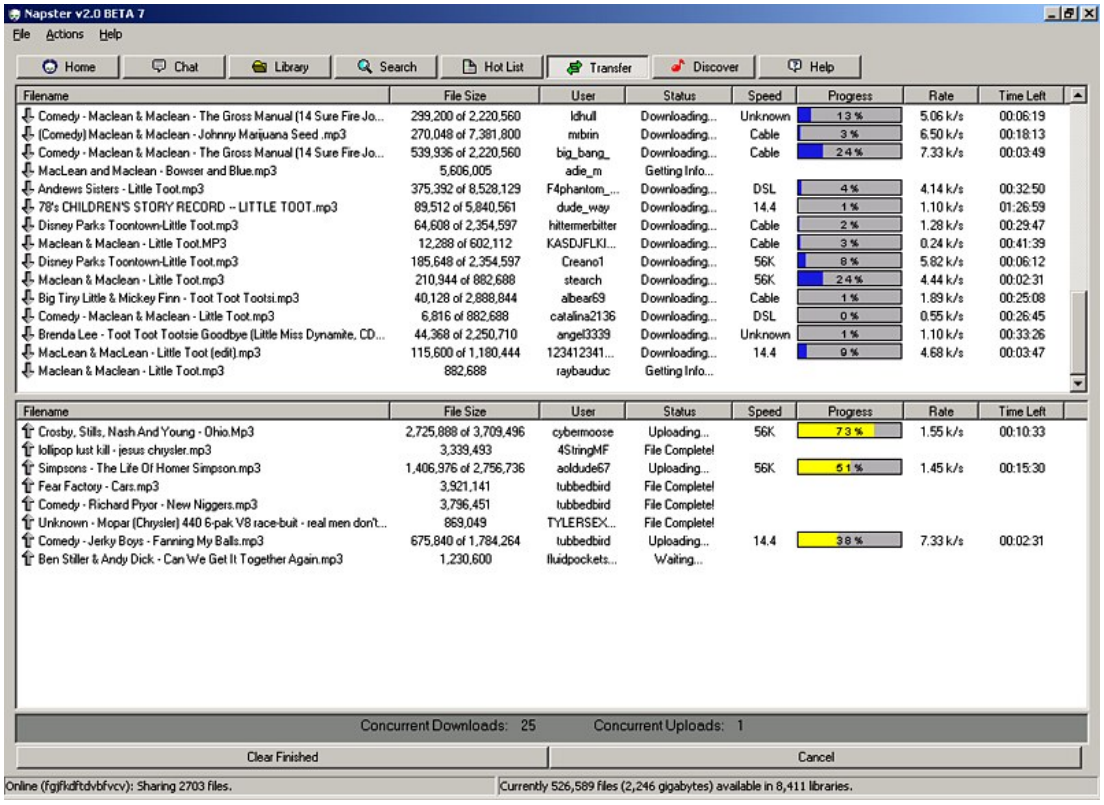

Figure 3.4: Screenshot of Napster. Downloads can be seen at the top, with uploads at the bottom.

In 1999, Napster pioneered the idea of global P2P file sharing. This early stage MP3 sharing network was supported by a centralised file search facility. Users could query this centralised database for desired content and their clients would be advised of other users currently sharing that piece of content. The client would then proceed to download this content directly off the user hosting the content. Due to its "free" usage model, Napster quickly grew to become a large global P2P network. By the end of 2000, Napster grew to

over 75 million users sharing over 10,000 MP3 files every second [62]. The centralised index ultimately was the downfall of Napster's design. In July 2001, after losing a court case with the Recording Industry Association of America, Napster became the first P2P system to be ordered to shutdown in what has become known as the "Napster Decision" [74].

3.4.2 Gnutella

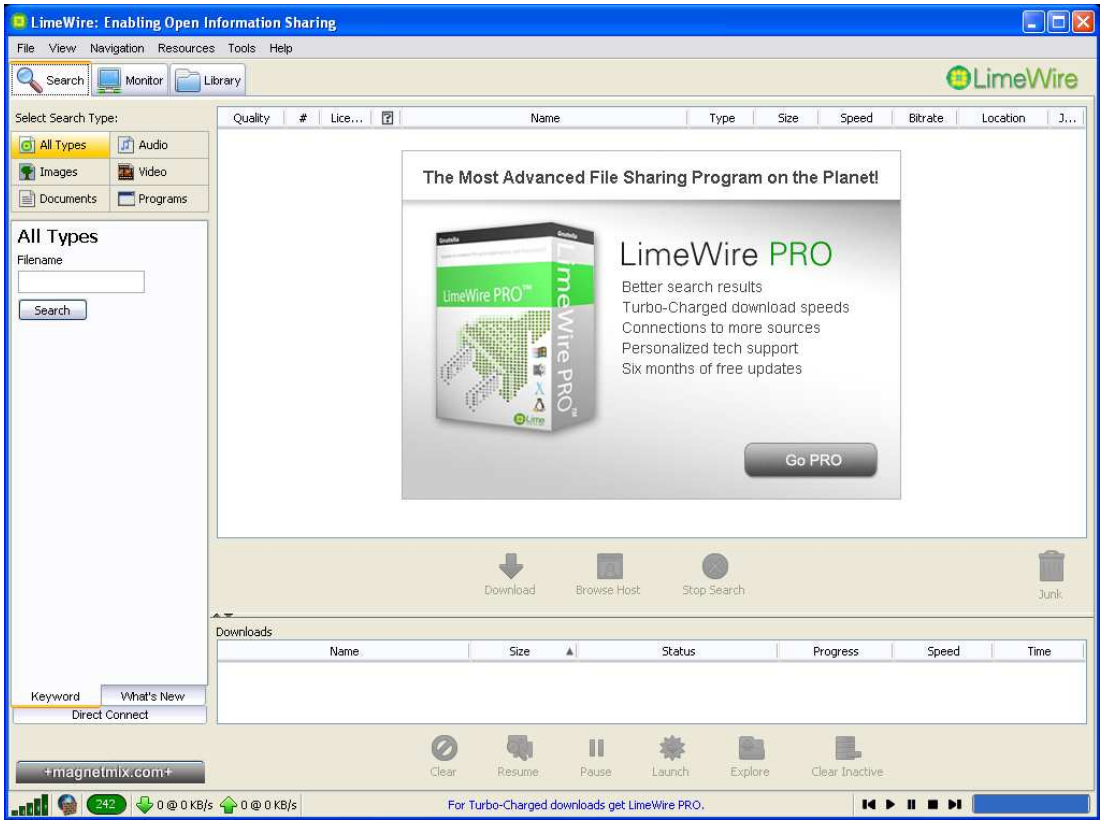

Figure 3.5: Limewire Screenshot.

Gnutella was released in 2000 as an open source file-sharing protocol by Nullsoft. When the system was released online, Nullsoft's owners, America Online Inc., first learned of its existence and the company quickly ordered the removal of the release from the Internet. However, the protocol was already downloaded by numerous other developers who were able to publish the specification [76]. This enabled numerous popular clients to be built on the protocol including BearShare, FrostWire, Morpheus, Shareaza and LimeWire

(shown in Figure 3.5). Gnutella clients are generally capable of resuming any partially completed downloads by reestablishing connections to previously known or new peers sharing the same content [77].

The Gnutella protocol adopts decentralised search algorithms which effectively eliminate the single point of failure of the centralised approach creating a much more robust network. This results in clients searching for content shared on Gnutella bouncing the query from node to node, with any hits being reported back through a reversed sequence of these network bounces [78]. A sample Gnutella node map is shown in Figure 3.6 with supernodes, represented as solid dots, acting as servers for many leaf nodes (represented as hollow dots) [76].

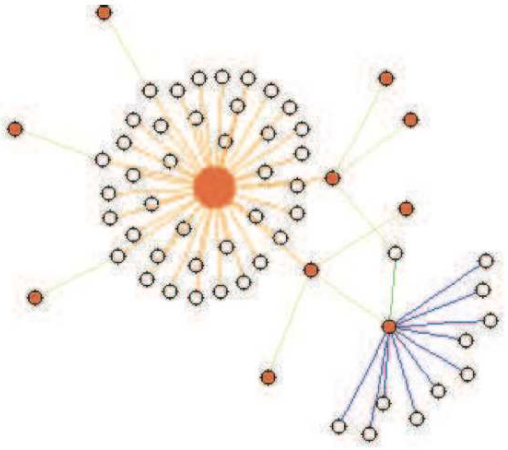

Figure 3.6: Gnutella Node Map.

The decentralisation of the network also left the network open and vulnerable to exploitation. While "legitimate" sharing of copyrighted content is the primary focus of the network, it was quickly exploited for malicious purposes by cybercriminals, e.g., the spread of viruses, worms and botnets. In 2008, Kalafut et al. found that 68% of all downloadable responses in Limewire (Gnutella's largest client at the time) containing archives and executables contained malware [79].

3.4.3 eDonkey

eDonkey is one of the most successful P2P applications and operates on a hybrid P2P network of the same name. Alternative clients built on the protocol include eMule, Morpheus, Shareaza and MLDonkey. The systems uses a distributed network of servers running a specific server application. The servers do not share any files and only aid in the management of the distributed information through indexing which peers are sharing which files. The network gained significant popularity in Europe; with Germany, France and Austria topping the geographical overview accounting for 66.21%, 6% and 1% respectively in 2004 [80]. Files are divided into chunks of 9,500kb which an MD4 checksum associated with each chunk.

3.4.4 BitTorrent

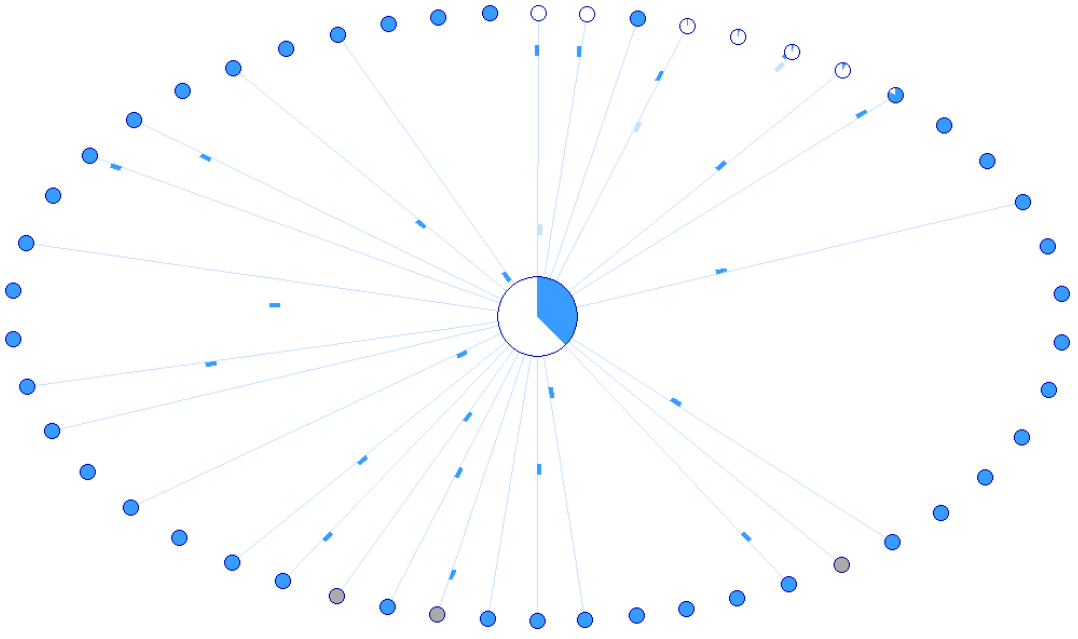

Figure 3.7: Visualisation of a Typical BitTorrent Swarm

In July 2001, the first implementation of the BitTorrent protocol was released. The BitTorrent protocol is designed to easily facilitate the distribution of files to a very large number of downloaders with minimal load on the original file source [81]. This is achieved through the downloaders uploading their

completed parts of the entire file to other downloaders. A BitTorrent swarm is made up of seeders, i.e., peers with complete copies of the content shared in the swarm, and leechers, i.e., peers who are downloading the content. Due to BitTorrent's ease of use and minimal bandwidth requirements, it lends itself as an ideal platform for the unauthorised distribution of copyrighted material. This typically commences with a single source sharing large sized files to many downloaders.

Based on global bandwidth usage, BitTorrent is the most popular P2P network in use today. In 2005, D. Erman measured BitTorrent traffic to account for over 60% of the world's bandwidth usage [82]. The BitTorrent protocol is designed to easily facilitate the distribution of files to a potentially large number of interested parties, i.e., other peers, with a minimal load on the original file source, as outlined in the BitTorrent protocol specification. This is achieved through the following steps:

1. The file is split up into a number of uniformly sized pieces or chunks with typical chunk sizes generally ranging from 128kB to 4MB.

2. The initial source of the file creates a UTF-8 encoded ".torrent" metadata file, which includes unique SHA-1 hash values for the entire file and each of the file chunks, along with other required file information, e.g., filenames, chunk size, total file size, path information, client information, comments etc.

3. This metadata file is then shared by the creator with other users interested in acquiring the original content either through direct distribution, e.g., email, instant messaging etc., or through the much more common method of uploading onto a torrent indexing website, such as ThePirateBay.org. Following the recent trend of maximising decentralisation of the BitTorrent eco-system, many indexing websites now only serve "magnet" URIs. These URIs enable the user to connect to a distributed hash table, as outlined below, and acquire the metadata file from other users.

4. Users interested in downloading the available content must then download this metadata file and open it using a BitTorrent client, such as Azureus/Vuze or µTorrent.

5. The BitTorrent client is then tasked with identifying other peers who are sharing the file uniquely identified in the metadata file, i.e., other peers in the swarm. This includes identifying seeders, i.e., peers with complete copies of the content shared in the swarm, and other leechers, i.e., peers who are currently downloading the content, but are sharing the completed chunks with others. This peer discovery is achieved through a variety of methods including tracker communication, distributed hash tables and peer exchange.

The success of the BitTorrent protocol can be attributed to uploaders incurring no additional cost (besides their Internet connectivity costs) to share files with many users. In practice, the original uploader needs only to stay connected to the swarm until a sufficient number of leechers have one full copy of the file between them. This is made possible through the leechers uploading their completed chunks of the entire file to other downloaders. Due to BitTorrent's ease of use, minimal bandwidth requirements and perceived Internet anonymity, it lends itself well as an ideal platform for the unauthorised distribution of copyrighted material. Initially this distribution consists of a single original source for sharing a large sized file between many peers.

Each BitTorrent client must be able to identify a list of active peers in the same swarm who have at least one piece of the content and is willing to share it, i.e., that has an available open connection and has enough bandwidth available to upload. By the nature of the protocol, any peer that wishes to partake in a swarm, must be able to communicate and share files with other active peers. There are a number of methods that a client can attempt to discover new peers which are in the swarm:

1. Tracker Communication – BitTorrent trackers maintain a list of seeders

and leechers for each BitTorrent swarm they are currently tracking. Each BitTorrent client will contact the tracker intermittently throughout the download of a particular piece of content to report that they are still alive on the network and to download a short list of new peers on the network.

2. Peer Exchange (PEX) – Peer Exchange is a BitTorrent Enhancement Proposal (BEP) whereby when two peers are communicating, a subset of their respective peer lists are shared during the communication.

3. Distributed Hash Tables – Within the confounds of the standard BitTorrent specification, there is no intercommunication between peers of different BitTorrent swarms. Azureus/Vuze and µTorrent contain mutually exclusive implementations of distributed hash tables as part of their standard client features. These DHTs maintain a list of all active peers using each client and enables cross-swarm communication between peers. Each peer in the DHT is associated with the swarm(s) in which it is currently an active participant.

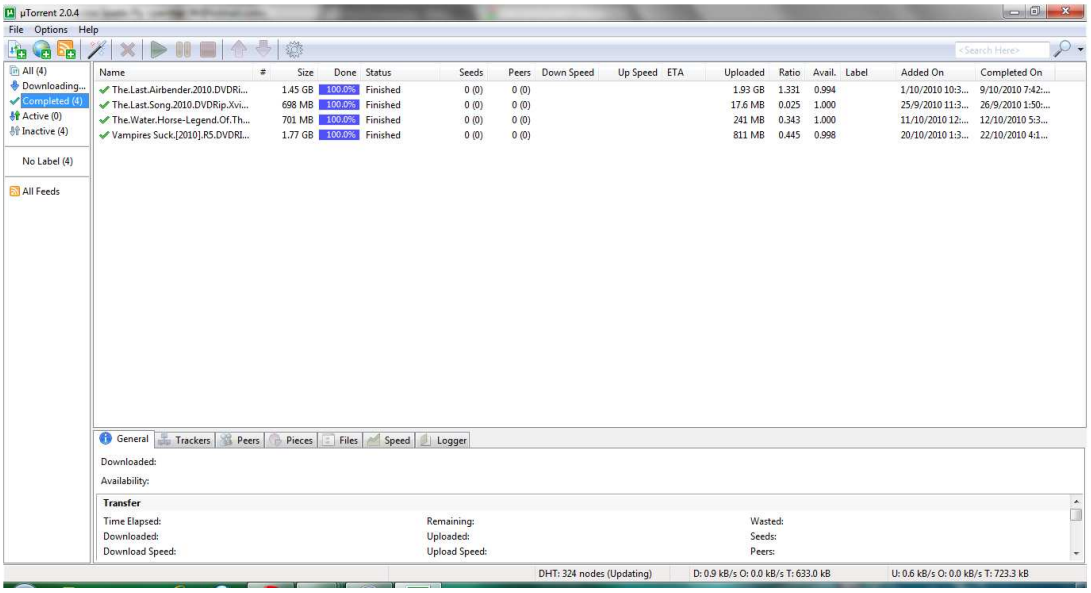

Figure 3.8: µTorrent Screenshot.

Due to the fact that the protocol is openly documented, numerous BitTorrent clients are available besides the official BitTorrent clients produced by BitTorrent Inc. [83], such as Azureus/Vuze, µTorrent, Shareaza, BitLord,

BitComet. Each client maintains its list of known active peers in a different manner. Different client parameters such as when the client decides to close an inactive connection and how the upload capacity is distributed results in some clients being more performant than others. In 2010 Iliofotou et al. conducted a large scale measurement study involving more than 11 million clients across over six thousand ISPs comparing the real world performance of the two most popular BitTorrent clients, Vuze and µTorrent [84]. It was found that µTorrent is on average 16% faster than Vuze.

3.5 Anti-Infringement Measures

With the popularity of acquiring copyrighted content illegally, many content producers are employing a number of technical solutions in an attempt to combat online piracy. In recent years, a number of anti-P2P companies have started to offer their services to the content producing industries.

3.5.1 Attacks on Leechers

In 2008, Dhungel et al. identified some of the techniques employed to interfere with unauthorised downloading and measured how successful attacks on BitTorrent leechers were [85]. Two different attack vectors were identified:

1. Fake-Block Attack – In this attack, the goal is to prolong the download time for a particular leecher. This is achieved by offering fake blocks of the desired content. While this is easily identified client-side through the hashing of the completed block, nonetheless time is wasted as the hashing can only occur once the 128kB to 4MB block is completely downloaded. The download could be further delayed if the peer decides to redownload this block, or any other block, from an attacker.

2. Uncooperative Peer Attack – In this scenario, the attacker joins a swarm and establishes connections with as many peers as possible without ever

sharing any blocks of the content. With each peer generally maintaining in the order of ten active connections at a time, taking up one or more of these valuable connections can have a significant impact on the performance of the user's download.

3.5.2 Pollution

In 2005, Liang et al. identified that one way employed to combat unauthorised file-sharing of copyrighted content is to spam the network with large volumes of bogus or polluted files [86]. With many P2P networks relying on a simple metadata (movie title, artist, song title, etc.) keyword search to locate desired content, polluting the P2P ecosystem with bogus files might be seen as a useful copyright infringement countermeasure to copyright holders. Due to each piece of content having as many as 50,000 different variations available, polluting the system with fake versions of the content can be a simple process merely requiring a user to rename a bogus file with the desired popular title.

3.6 Forensic Process/State of the Art

3.6.1 Network Crawling

Crawling a P2P network attempts to discover each node participating in a given network. Depending on the network design, total peer enumeration may be possible, with decentralised networks generally proving easier to crawl. For example, Napster can only be crawled through the responses returned by querying the system for specified content whereas Gnutella can be crawled through the exploitation of the ping/pong peer discovery messages built into the protocol [87]. Network crawling may attempt to find all the nodes sharing a specified piece of content, or attempt to enumerate the size and geolocation of the entire network.

3.6.2 Deep Packet Inspection

Deep packet inspection attempts to classify packets as belonging to a given network. Traditionally this task was deemed relatively simple to implement as specific applications generally had a specific port number assigned to it. P2P traffic is becoming harder to identify due to port obfuscation, encryption and tunnelling. DPI has evolved to take packet flows into consideration when attempting to identify traffic. A combination of per-packet sampling and per-flow sampling is generally used to aid in identification. In 2013, Khalife et al., deployed an OpenDPI testbed in an attempt to identify encrypted P2P traffic [20]. Incorporating packet flow analysis greatly improved the accuracy of the detection of P2P traffic, as can be seen in Figure 3.9.

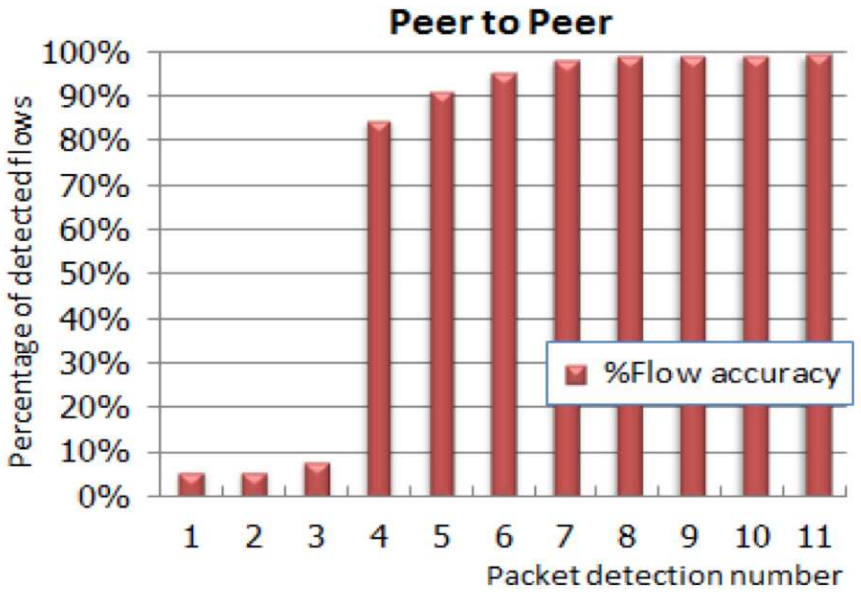

Figure 3.9: Flow accuracy results for P2P traffic as a function of the packet detection number

3.6.3 Identifying Copyrighted Content

In 2008, Nasraoui et al. proposed a system to identify copyrighted content (movies, TV shows, ebooks, audio files, etc.) [88]. It was proposed that a database of known file hashes and metadata could be maintained by law enforcement agencies facilitating the identification of any content. In order

for such a system to be complete, it would require cooperation from content producers and most likely a large number of volunteers in a wiki-style approach to maintenance. In this model, each audio or video file would have to be downloaded and manually verified to be a copy of the original. The hash of the verified file would then be added to the shared database.

An alternative to the per-file hashing approach has been developed by Audible Magic Corp. The company has patented an audio and video identification methodology for identifying any piece of content through an heuristic approach [89]. This system creates a "fingerprint" of the content not based on the digital signature, but instead based on the recognition of the audio wave patterns produced by playing the file. This facilitates the identification of known content, irrespective of the codec, bitrate or metadata of the file.

3.7 Forensic Counter-measures

Due to privacy concerns from the monitoring of P2P file-sharing, some users attempt to circumvent detection through the use of forensic countermeasures. Many ISPs monitor their network traffic and perform throttling or "traffic shaping" in order to curtail bandwidth hogging services, such as P2P file-sharing.

3.7.1 Anonymous Proxies

Some users of P2P file-sharing employ anonymous proxy services, such as Tor (The Onion Router) [90] or I2P (Invisible Internet Project) [91]. These services are distributed overlay networks designed to anonymise TCP-based applications. Each packet sent from a client operating on the network is encrypted and subsequently bounced through a random number of nodes before reaching its destination server. The response is then sent back through another random path [92]. There are currently no methods available to reverse

engineer a users actual IP address from the traffic originating from a Tor exit node. However, in 2012, Gilad et al. outlined a number a methods for detecting whether a given client is using Tor or not [93].

3.7.2 Encrypted Traffic

The use of encryption helps to overcome some of the network forensic investigations utilising packet sniffing or deep packet inspection. An number of P2P networks employ encryption methods for all communication, e.g, SSL. Many of the tools deployed by ISPs and network monitoring companies rely on DPI and payload heuristics to analyse network traffic and this encryption renders these approaches ineffective [94].

3.7.3 IP Blocking

To avoid detection, some users employ custom firewall tools, such as PeerBlock, PeerGuardian, Moblock, etc., to ban any incoming or outgoing communication with specific other users [95]. These tools accept a list of defined "bad" IP addresses. Users can create their own lists or acquire lists of known bad IP addresses from numerous online services. One such service is iBlocklists [96], which allows users to download lists of known IP address ranges for a number of content producing companies, governmental organisations, educational institutions, anti-piracy companies, etc. Such lists can contain over 222,000 IP address ranges and in total can cover over 796,128,149 IP addresses [97].

3.8 Malware Risks on P2P Networks

For almost as long as P2P networks have been used for file-sharing, they have been exploited for the propagation of viruses and malware. Keyword based metadata searching systems are most vulnerable to attack due to their

simplified searching method. For example, a user may easily download and attempt to play an executable file as opposed to the music file he was attempting to download. Users can be easily tricked into downloading these files, by renaming the malware file with a desirable popular artist name, e.g., "Katy Perry.exe". In 2006, Shin et al. found that over 12% of Kazaa client hosts were infected by over 40 different viruses, with 15-22% of the total crawled data in their investigation containing viral files [98]. The Kazaa installation file also came bundled with malware. During the install process, and in order to complete the installation, the user has to agree to installing some third-party software alongside the Kazaa installation, as can be seen in Figure 3.10.

Figure 3.10: Kazaa end user licence agreement

In an attempt to propagate itself, many malwares will connect to a popular P2P network and attempt to get other users to download and infect their machines. Many of these self-propagating malwares will cleverly respond with a dynamic filename based on whatever keywords the incoming request query contains [99].

3.9 Summary and Discussion

In this chapter, P2P file-sharing systems and their differing design characteristics were introduced. Many of these popular networks are used mainly for the unauthorised distribution of copyrighted material which costs the content producing industry billions of dollars every year. Some of the methods used for investigating these networks were also introduced. These generally involve either building a bespoke network crawling tool or deploying a hardware/software network packet analysis system.

3.9.1 Weaknesses of Current Investigative Approaches

If different investigative bodies, e.g., law enforcement from different countries, wish to investigate the same network, each body needs to start from scratch in the development of their own tool. As a result, from a P2P cybercrime investigation perspective, a significant amount of time is wasted globally in the duplication of developmental, investigate and analysis efforts in an attempt to reach a common goal. The universal P2P network investigation framework described in Chapter 5 introduces solutions to some of these issues.

CHAPTER

FOUR

BOTNET INVESTIGATION

4.1 Introduction

In the past, cyberattackers required high-end computer equipment coupled with high bandwidth Internet connections to accomplish their goals. In recent years, high bandwidth at home and workplace broadband Internet connections have become common-place. This has resulted in these computers being targeted by criminals to create large, global distributed systems, i.e., botnets, to perform their bidding. The software robots, or bots, which form these distributed systems are controlled remotely by the criminal attacker, or botmaster. The paradigm of modern botnet cybercrime involves enslaving compromised computers as a strategic criminal asset. Traditionally viruses were created with the intention to attack and destroy infected systems, but now malware has evolved to gain control of infected machines and use these machines to build a global network to perpetrate cybercrimes [100].

Botnets have become the tool of choice to conduct a number of online attacks, e.g., DDoS, malware distribution, email spamming, phishing, advertisement click fraud, brute-force password attacks, etc. Criminals involved in conducting their craft online all share one common goal; not to get caught. Botnet design, as a result, has moved away from the traditional, more traceable and easily blocked client/server paradigm towards a decentralised P2P based communication system. P2P Internet communication technologies

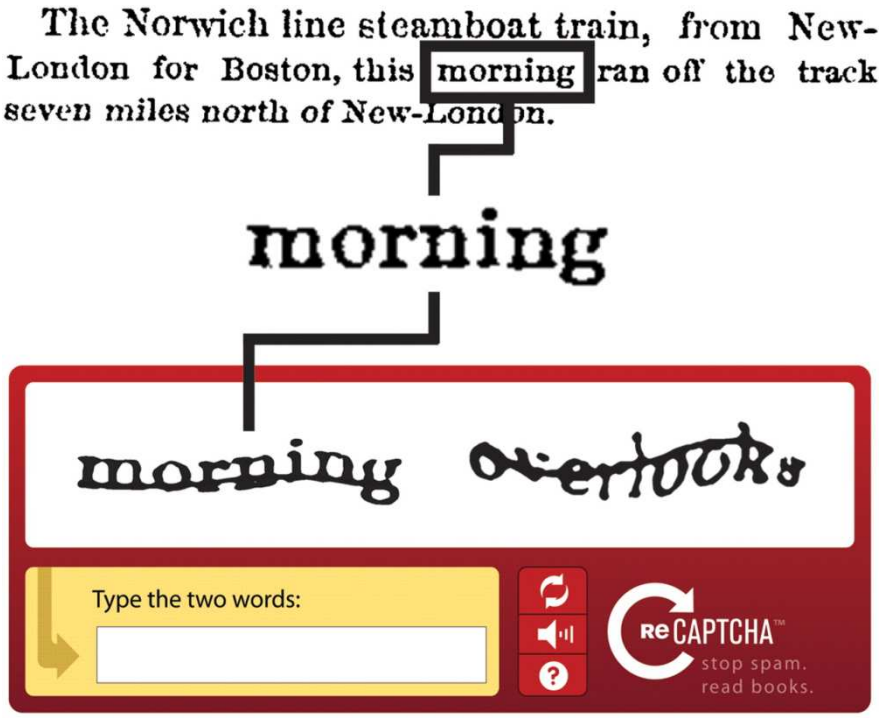

Figure 4.1: Sample CAPTCHA from the reCAPTCHA online service and its automated book scanning source text

lend themselves well to use in the world of botnet propagation and control due to the level of anonymity they award the botmaster. For the cybercrime investigator, identifying the perpetrator of these P2P controlled crimes has become significantly more difficult. This chapter outlines the state-of-the-art in P2P botnet investigation.

The prevalence of large, global botnets has resulted in many online services deploying human authentication systems to reduce or eliminate automated registration for forums, email accounts, social networks, etc. These systems aim to prevent automated botnet login attempts resulting in password cracking or spam. The most common test for telling whether any given visitor to a website is human or machine is to employ "Completely Automated Public Turing test to tell Computers and Humans Apart" (CAPTCHA) verification. The test involves presenting the user with a string of obfuscated characters and requires the user to identify the often scrambled words in order to proceed. Figure 4.1 shows a sample word taken from a text with unreliable results from regular optical character recognition algorithms [101]. It has now

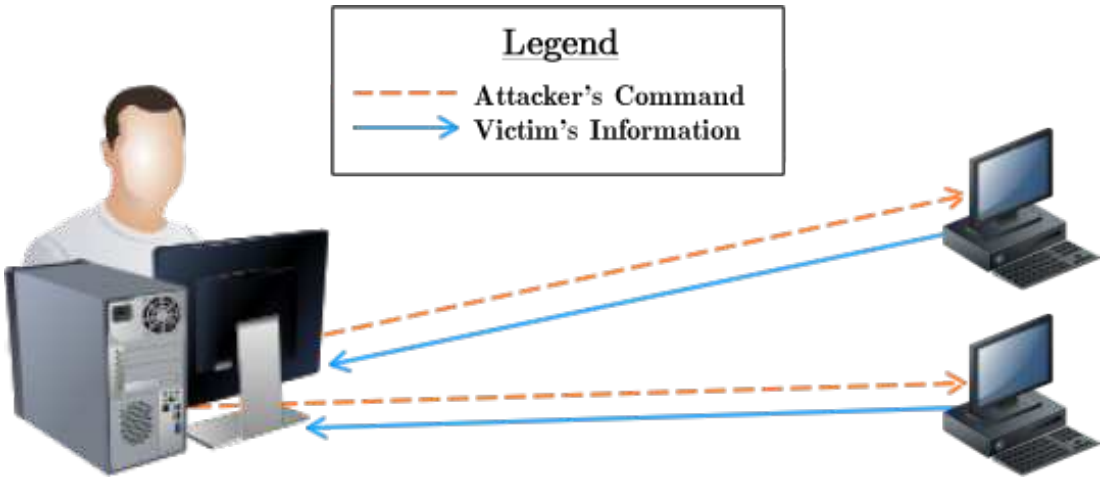

Figure 4.2: Simple Trojan Horse Architecture Controlling Multiple Computers

become common to use these scanned words which are difficult automated recognition to verify human website usage. The human delivered results from these CAPTCHA systems can feed directly into text scanning and recognition projects, such as Google Books.

4.2 Botnet Architectures

Arguably, the simplest implementation of botnet technology involves an attacker manually taking control of each victim's computer using a remote trojan horse based attack, as shown in Figure 4.2 [102]. In this model, the attacker had one-to-one direct control of the victim's machine. The target of the attack would generally be the user of the infected machine as the attacker is capable of capturing keystrokes, executing any applications, intercepting print jobs, accessing local or network files and destroying the victim's operating system or system configuration. From an anonymity standpoint, this design has significant privacy issues for the attacker. Any investigation of the trojan's network traffic would easily reveal the attacker's IP address, which could subsequently be resolved back to reveal his identity.

Subseven was one of the most popular trojan horse systems (alternatively referred to as Sub7 or Sub7Server). It consisted of a client trojan horse virus

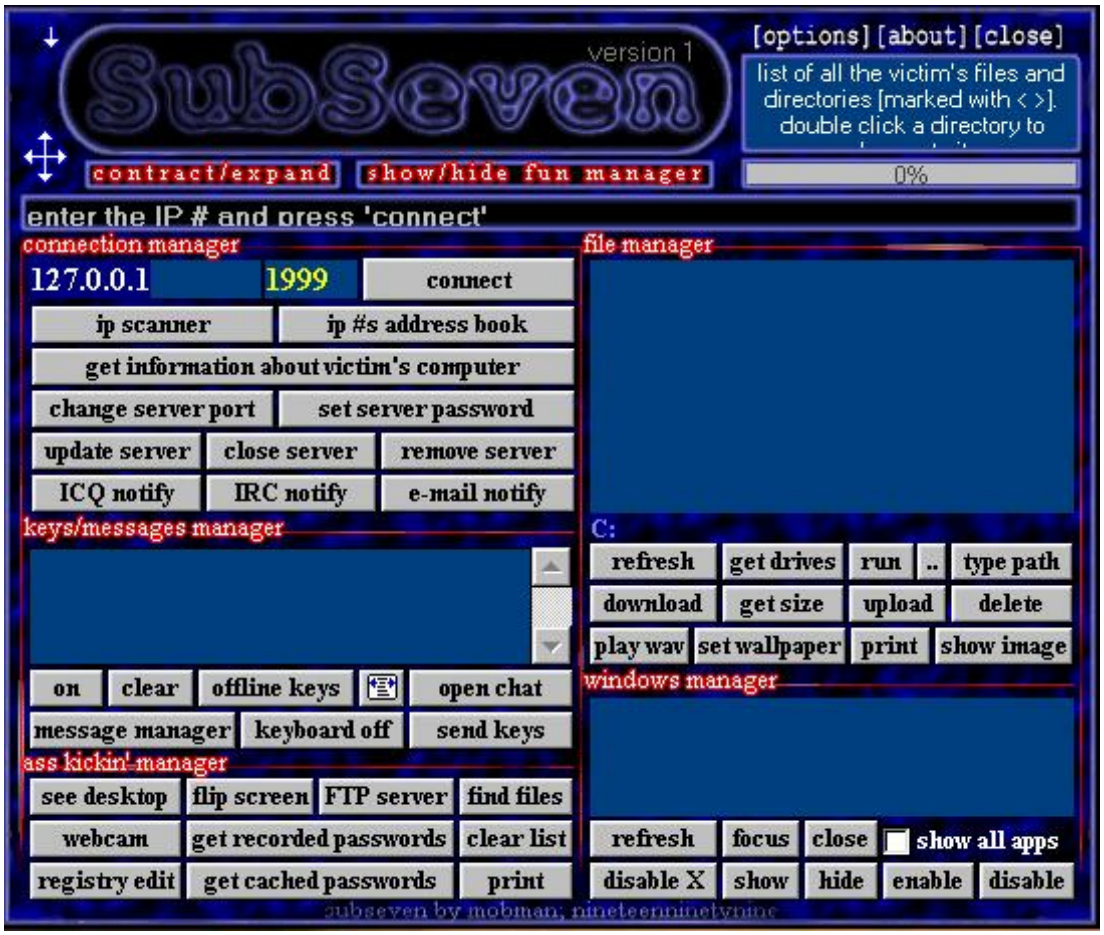

Figure 4.3: Subseven Control Panel

and a server or control panel to facilitate the attack. Released in 1999, it targeted vulnerable Windows machines [103]. The types of commands the attacker could execute can be seen in Figure 4.3.

4.2.1 Client/Server Botnet Design

The simplest botnet architecture relies on a central server to control all nodes on the network, often referred to as a centralised design. When a bot comes online, it registers its availability with the server, which in turn issues the bot with commands for the work it must complete. These Command and Control (C&C) servers are directly controlled by the botmaster. The architecture of this C&C based system contains a single controlling server with multiple compromised machines communicating with it, as outlined in Figure 4.4.

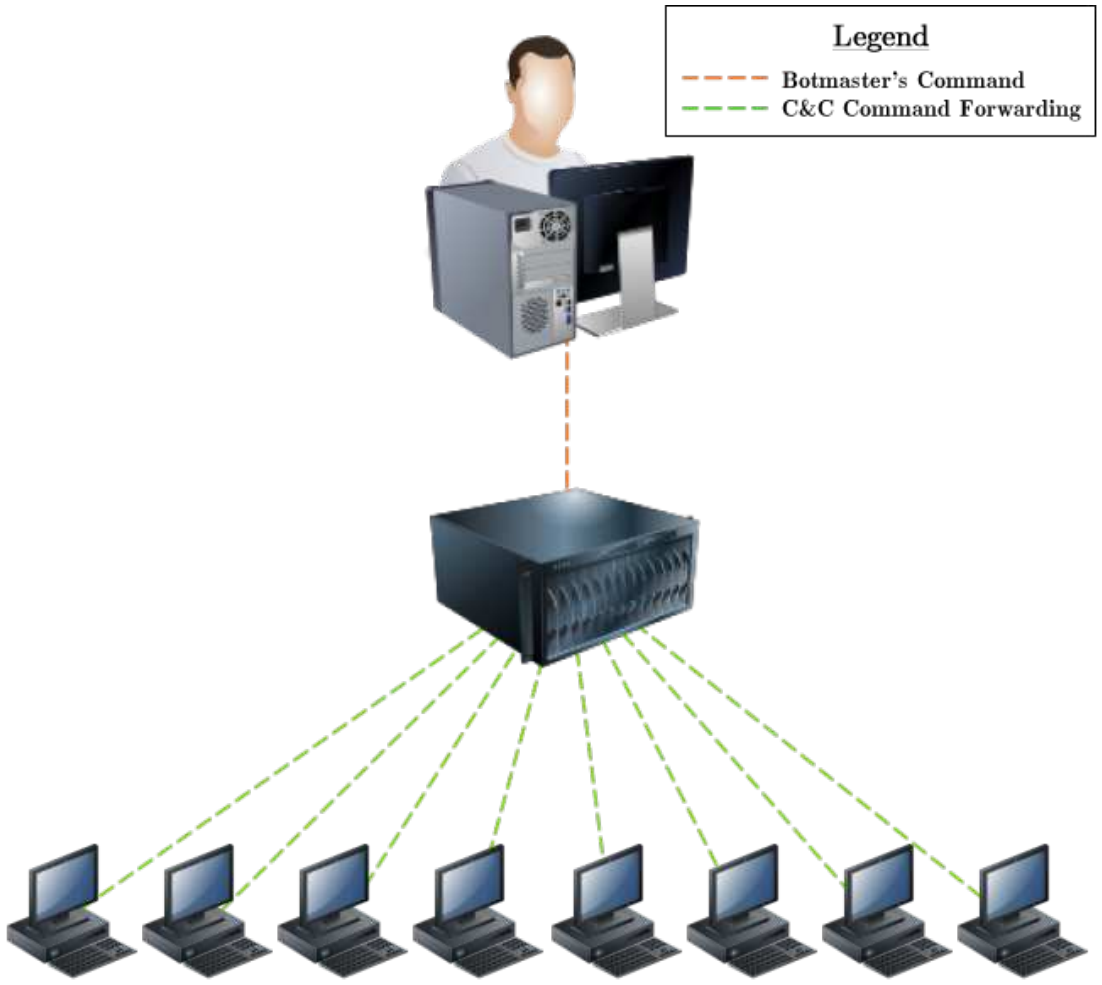

Figure 4.4: Command and Control Server Botnet Network Architecture

Traditional botnet design was centred on a client/server paradigm (see Figure 4.4). Using this model, the botmaster issues requests to the HTTP (regular website based communication) or Internet Relay Chat (IRC) based C&C server. In the case of IRC, each connecting bot will pick a randomised nickname for the chatroom and as a result, if necessary, the server has the ability to issue unique commands to each bot. The use of a C&C server eliminates the need for the botmaster's computer to remain online in order to distribute the latest orders to the entire botnet. C&C servers also award the botmaster an added level of anonymity from detection.

An advantage to using the HTTP based design over the IRC design, is that the bots themselves do not need to be continuously connected to the server [104]. Instead, the client-side HTTP bot software, which runs on the infected

nodes, is programmed to periodically "check-in" with the C&C server in order to get its latest commands. In the IRC based model, the bots remain connected to the IRC channel while online. This leaves the system vulnerable to IRC based investigation and manipulation by law enforcement or other forensic investigators.

The main concern with this simple client/server model is that it leaves the botnet vulnerable to a single point of failure. To counteract this, multiple C&C servers may be used optionally in conjunction with a dynamic Domain Name System (DNS) service, such as DynDNS [105] or No IP [106]. The dynamic hostnames required are hard-coded into the bot software, enabling the botmaster to quickly and easily swap in a new command and control server when needed. This is achieved by simply updating the IP addresses associated with the dynamic DNS provider ensuring no disruption of the botnet's regular operation. Cloud services lend themselves well to being exploited for running C&C servers and offer the botmaster the ability to quickly and easily change not only the IP address, but the geographic location of the C&C servers frequently. In 2009, Amazon discovered that its Elastic Compute Cloud (EC2) was being used by a new version of the Zeus botnet for its C&C functionalities [107].

The weak point of the original HTTP/IRC based command and control centre botnet design, as it can be seen in Figure 4.4, is that there is a single point of failure. If law enforcement or any other third party wished to destroy the botnet, the command and control server can be targeted and the botnet can be effectively destroyed, i.e., left without any commands or work to complete. The next evolution of botnet design incorporated multiple C&C servers to attempt to alleviate the strain and weaknesses of a single server design, as it can be seen in Figure 4.5. In this model, each bot in the system will register and check-in with multiple C&C servers.

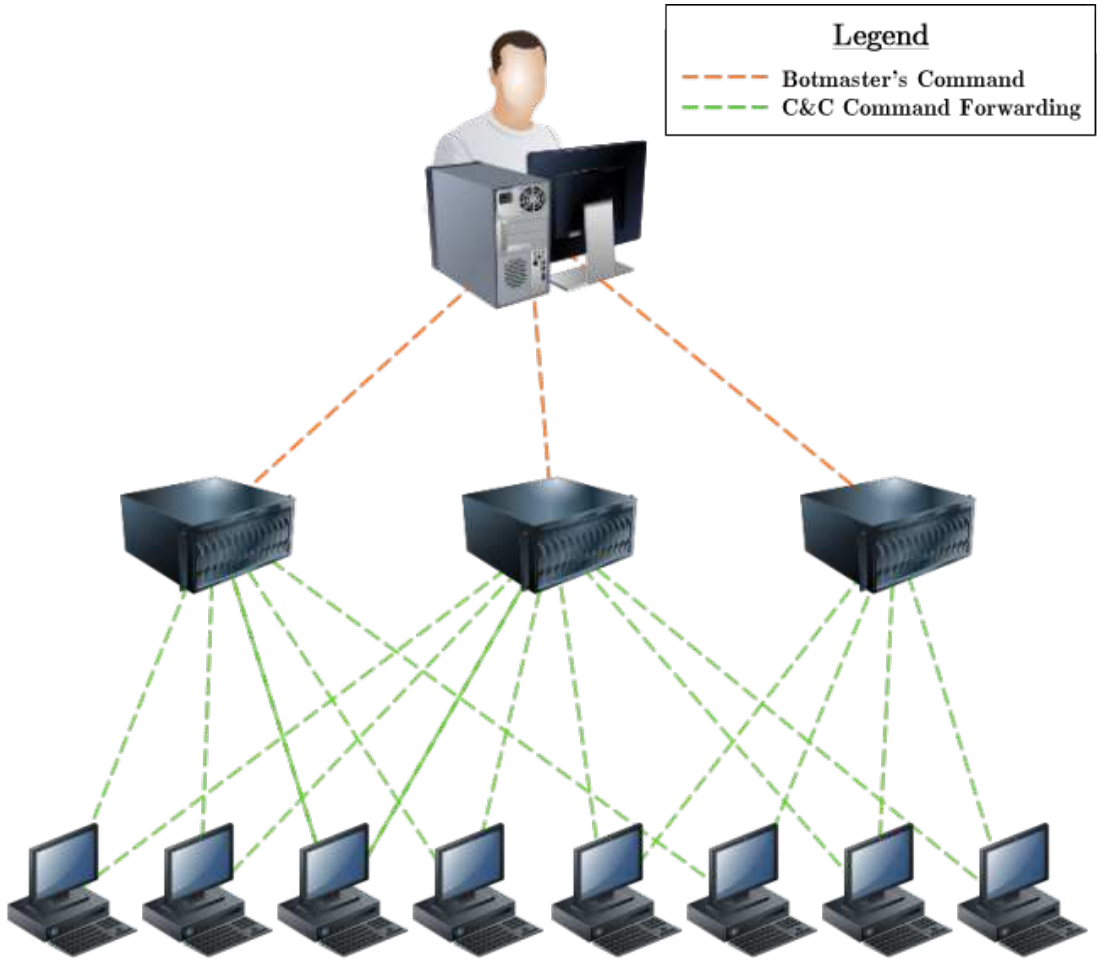

Figure 4.5: Evolution of botnet architecture to eliminate single point of failure

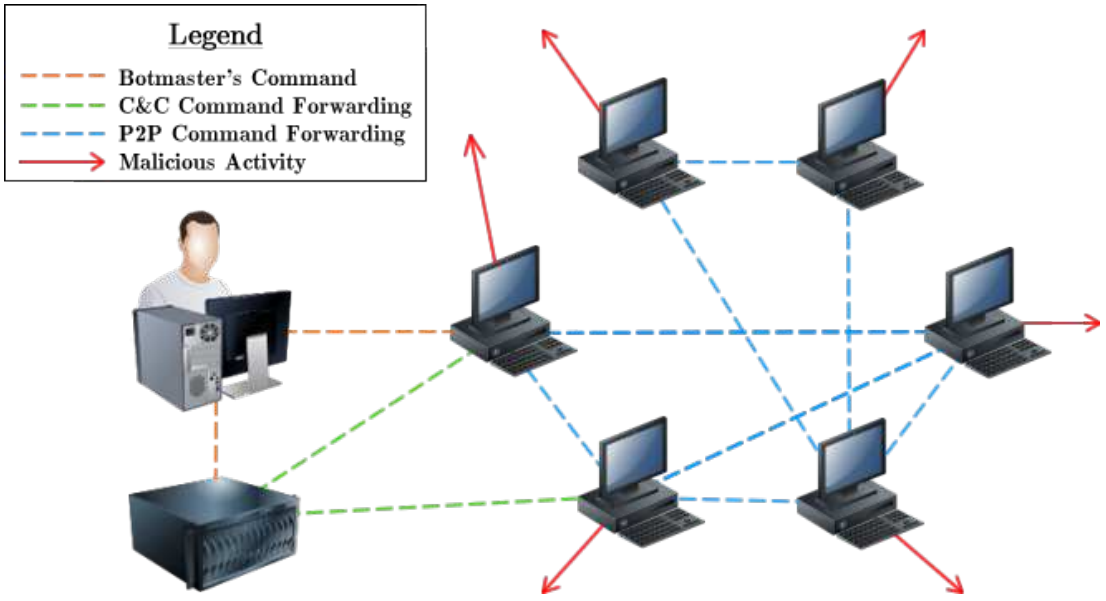

Figure 4.6: Typical botnet topology with commands optionally routed through a C&C server.

4.2.2 P2P Design

The design of botnet architecture has continuously evolved over the last number of years to improve the performance and attack resilience of the system, while awarding a greater degree of anonymity to the botmaster. A natural progression of the client/server design involved expansion to incorporate as many C&C servers as possible. Further extension of this model involves utilising P2P technologies in the architecture of the systems. This effectively turns each bot in the network into a C&C server. This results in every active node in the network having the ability to communicate with a subset of all the other nodes in the same botnet. Commands and updates are passed from peer to peer in a manner that eliminates the need for controlling servers. As a result, the single point of failure of the client/server botnet design is effectively eliminated [108].

Decentralised botnet design relies on a DHT to record all the active nodes on the network [109]. In order for any new bot to participate in the network, it must have an avenue available to it to initially connect to this DHT. This is generally done through the hardcoding of an initial "seed" list or a list of bootstrapping servers. The seed list is a local cache of IP addresses which are more likely active nodes in the network. A node attempting to join the network can contact either a seed or one of the bootstrapping servers to connect to the DHT and begin regular operations. Without some initial hardcoded bootstrapping method to connect to the DHT, the only other completely decentralised option available to a P2P system is to employ random address probing . While this may initially appear an unlikely method, recall that discovering a single node connected to the DHT is sufficient to join the system and some of the DHTs involved can contain millions of active nodes. In practice, Dinger et al. [110] found that limiting the randomised scan rate to 100 packets per second resulted in locating a BitTorrent DHT peer within ten minutes with a probability of $\geq$94%.

Botnet developers are continuously updating and improving botnet design.

In 2013, Memon et al. proposed a new botnet system, named Tsunami, which attempts to eliminate the bespoke bootstrapping server weakness of decentralised P2P botnet design by parasitically transmitting all botnet communications on the existing global Kad network [111]. The Kad network is a DHT based P2P network with over 4 million users most commonly used by numerous P2P file-sharing applications, such as eMule and MLDonkey. Commands are sent from the botmaster to any active bots using the "lookup" messages in Kad. This facilitated a hidden payload instruction up to 106 bits in length. The lookup command in Kad is automatically passed from node-to-node in a similar method to that of querying the Gnutella P2P file-sharing network, as described in Section 3.4.2. It was found that Tsunami could reach 75% of its bots within 4 minutes and receive responses back from 99% of these bots.

4.2.3 Hybrid Design

A hybrid botnet topology builds upon the P2P design by promoting particular high uptime and high bandwidth nodes to "supernode" status, in a similar fashion to Gnutella outlined in Section 3.4.2. This results in a two tiered model with client bots and servant bots [112]. The servant bots are the only ones receiving the commands directly from the botmaster and are responsible both for spreading those orders throughout the network and are responsible for the maintenance of the network itself [113].

4.3 Botnet Lifecycle

For most botmasters, the botnet lifecycle starts with the configuration of the botnet client (infecting malware to run on victims' machines) and the botnet controlling server (responsible for the dissemination of the latest updates and commands). There are numerous software solutions available to criminals wishing to create their own botnet, requiring varying degrees of technical

knowledge and costs:

1. Buy or rent an existing botnet – There are numerous avenues available to the criminal to buy or rent partial or entire botnets. Botnets are rented or sold for differing prices depending on the associated "quality", i.e., size, node uptime, bandwidth, latency, geolocation, etc. This option requires minimal technical knowledge and the perpetration of the crime can commence almost instantaneously. This model shares many of the characteristics as renting cloud computing resources from legitimate providers. Minimal upfront time and money is required on behalf of the botmaster to get started.

2. Buy pre-developed botnet software – This option will supply the criminal with developed software. This software is configured to spread the client malware via P2P networks, email attachments, instant messaging, etc. The cost of purchasing such a system increases with the decreased likelihood that the client executable will be discovered by anti-virus software and the broader the operating system compatibility. Choosing this option will allow the criminal to create numerous different botnets if desired, based on the purchased technology. Some technical knowledge and time is required in order to spread and infect the desired number of nodes. A significant downside to this option is that the same software will likely be sold to numerous criminals who may each create several botnets. The more prevalent the software is, the higher the likelihood that the malware will be detected or reported to anti-virus/malware detection providers. In this instance, all botnets based on this software may be rendered useless. However, in order for this to occur, all victims must update their virus definitions and remove the infection.

3. Develop customised bespoke botnet software – This is the most resilient option against detection, but obviously requires the highest amount of technical ability. Before any development can commence, a vulnerability

must first be discovered in a popular operating system, browser, e-mail client or other common piece of software or hardware. This vulnerability facilitates the infection of the victims' machines, as outlined in greater detail in Section 4.3.1. Assuming a functional botnet client can be developed, the botmaster still has the same task as described above in spreading the infection to as many victims as required.

Once the botmaster has the required software (often a PHP/MySQL driven back-end installed on a command and control server), the configuration of the infecting malware must take place. Each newly infected machine must accept new orders, update the bot client software, send information back to the botmaster and potentially distribute orders with other known nodes in the P2P system. This configuration file will contain settings and required operational information, such as trusted C&C servers, update servers, DHT information, communication frequencies, resource usage limits, etc., [114]. In a purchased or rented system, this configuration step will likely have been completed by the seller.

To ensure maximum flexibility, any hardcoded host information, such as the list of trusted C&C servers, will generally be included using a dynamic hostname supplied by dynamic DNS providers such as DynDNS [105]. This allows the botmaster to regularly shutdown or move the C&C servers without needing to update any of the bots on the network. To achieve this move, the botmaster merely changes the IP address associated with each dynamic domain name supplied in the configuration file.

A typical botnet deployment lifecycle from a new vulnerable host's point-of-view can be seen in Figure 4.7 [115]. Each of the indicated steps are outlined in greater detail in Sections 4.3.1 to 4.3.5. Most botnets aim to achieve, from the point-of-view of the users of the effected machines, no discernible performance reduction to the regular expected operational speed. The bot's client can be configured to intelligently use only available resources and as a result, potentially never get discovered. A user with what appears

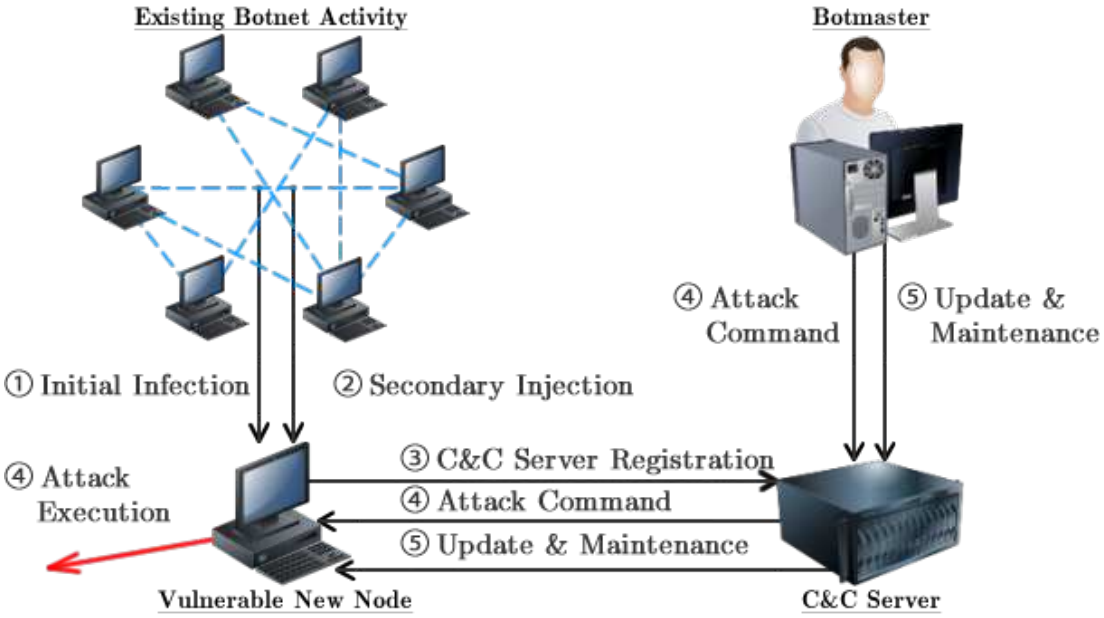

Figure 4.7: Typical Botnet Lifecycle from a Victim's Point-of-View

to be a fully functional computer may never decide to install or run anti-virus software.

4.3.1 Spreading and Infection Phase

In order for a botmaster to execute his desired acts, the bot malware client must be running on a significant number of infected machines. In order for a machine to get infected with the malware, either some user interaction is required or a software vulnerability is exploited in order for the binary executable to run [116].

The infection or "recruitment" phase is referred to as the phase in which a clean host is infected by a bot binary and, as a result, becomes a member of the botnet [117]. This generally involves some malicious executable compromising a host through any available means, e.g., taking advantage of a software or hardware exploit through social engineering, instant messaging, unprotected network shares, malicious email attachments or the mimicking of desirable content on download sites or on P2P file-sharing networks [118]. In 2007, the European Network and Information Security Agency (ENISA) conducted

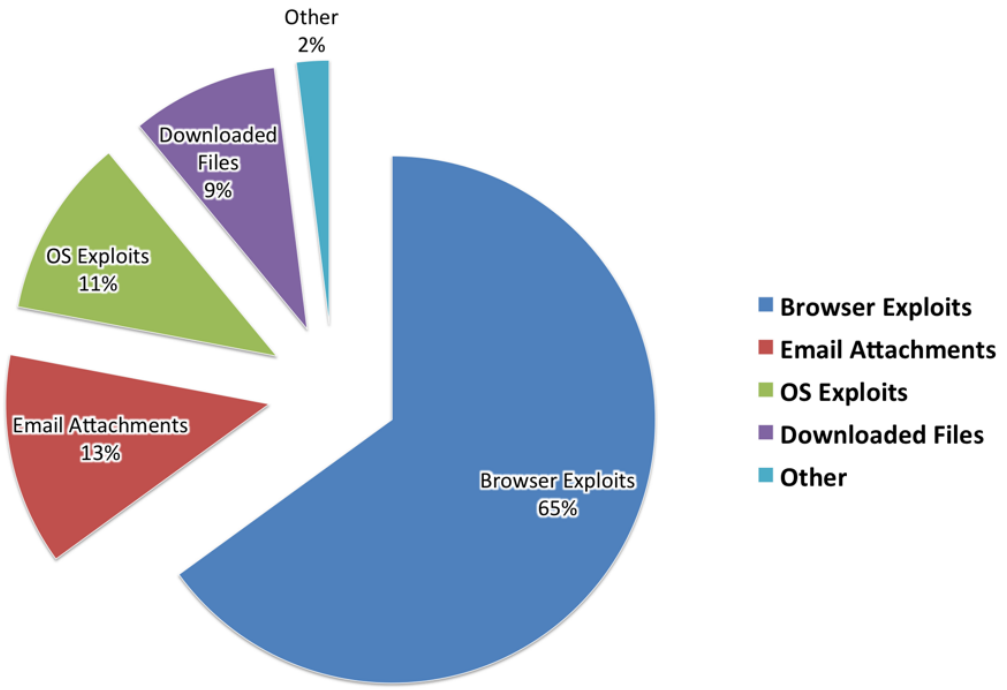

Figure 4.8: Typical Malware Attack Vectors

research into the infection vectors of botnets and found that the most common infection method was through browser exploits [119]. The complete results of their findings are shown in Figure 4.8.

Once a new machine is infected, many bots attempt to self-propagate by sending emails to the victim's address book, instant message contacts, etc. Some bots will also connect to popular P2P file-sharing systems in an attempt to dupe users into downloading the infection.

4.3.2 Secondary Code Injection Phase

Many of the attack vectors for the malware involve a memory buffer overflow exploit in software. Due to the nature of the attack, it is common that the initial "break-in" or "dropper" binary might not contain the entire bot client. It may merely serve the function of gaining the required operating system access rights to facilitate the install of the client software [120]. During this

phase, the binary will download and install the latest version of the botnet client onto the infected machine. Seeing as a common method for malware distribution is to email the malware as an executable or to entice a download from a P2P file-sharing network, it is also common for the version of the malware distributed to be out of date. For example, the default behaviour of most P2P file-sharing systems is to automatically share all downloaded files. As a result of this potentially outdated software, it is also during this secondary phase that the bot software will update itself to the latest version from the C&C server.

4.3.3 Command and Control Phase

It is during the C&C phase that a newly infected computer will become part of the botnet. A traditional client-server bot, once installed on a new machine, will immediately attempt to "phone home" through an IRC network or contacting a HTTP C&C server. Decentralised P2P bots are distributed with a predefined bootstrapping method to connect to the relevant DHT. Once connected, the newly compromised machine will ask one of its peers for the latest command. Some of the P2P bots require that a specific port is open for the peers to be able to communicate with each other [121]. Through the deployment of a firewall, many of the unnecessarily open ports on any given machine will be blocked. Any new application that attempts to access the network for any reason can also be flagged to the user, e.g., immediately after a recent infection of the botnet malware. Newer versions of the bot client software will not use a predefined port number to aid in avoiding detection.

There are two different ways to spread a command in a botnet system, namely push and pull. IRC based bots belong to the push-based category as they sit in an IRC chat room listening for a new command. HTTP based bots periodically check with the server to verify if there is any new work. P2P bots can do both as they send and receive commands to/from other bots.

4.3.4 Attack Phase

The attack phase is the most important in the botnet lifecycle from the botmaster's perspective. The purposes of designing, developing and spreading the botnet client malware onto as many machines as possible is to conduct whatever distributed illegal activities the botmaster has in mind. When a bot receives a command, so long as it remains online it will execute that command until one of the following two events occur:

1. A predefined stopping condition is met – This condition may be when a specified execution time has elapsed or when the objective goal is accomplished, e.g., the taking down of a website or service, or a password having been cracked, etc.

2. A new order is received – Any new order received will overwrite the current operation. Modern botnet design facilitates the execution of multiple orders simultaneously and each job may need to be manually ordered to cease.

Examples of the types of attacks conducted by botmasters and real-world monetary rewards for each are outlined in greater detail in Section 4.5.

4.3.5 Update and Maintenance Phase

Botmasters have the ability to issue update commands to their entire system of slave machines. Due to the interest in botnet detection and investigation, the attackers need to have a facility to upgrade their tools. This stage allows the botmaster to update the existing binaries and/or configuration to make the entire system more resilient to new digital forensic techniques.

The maintenance phase of the botnet lifecycle involves feeding execution, uptime and update information back to the botmaster. Figure 4.9 shows some of the information available to a botmaster including the number of currently

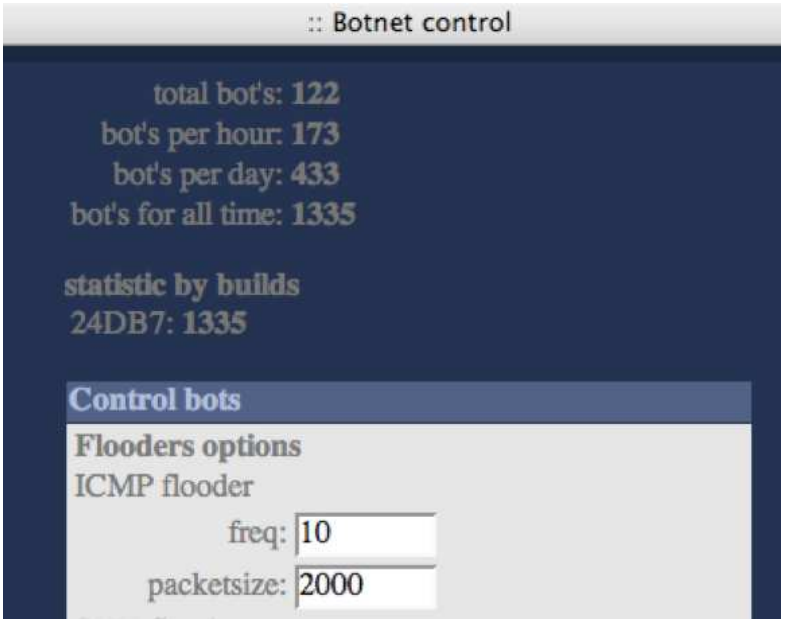

Figure 4.9: Screenshot from the Blackenergy Botnet C&C Server

active nodes, the churn rate per hour or day and the total number of detected bots [114].

4.4 Underground Economy

Given that financial gain is perhaps the biggest driving force in the growth of botnet technologies and volume of attacks, it was a natural progression that a large underground botnet enabled economy would develop. Numerous vendors, merchants, malware authors and customers are involved in the daily trade of personal information, attacks, spamming services and botnet technologies. Among the network of cybercriminals, a vibrant market has emerged trading in compromised credit cards and financial information. Much of this collected information would have been harvested by botnets. Those botmasters who target the collection of these financial details are usually unable to extract the funds directly from the accounts, so they usually sell them at a fraction of their value to experienced criminals or organisations who have a greater infrastructure available to them [100].

Much research has been conducted into how this large underground market

of trading criminal tools and technologies can be stopped, or at least hindered. Specifically to the trade of botnets, one method proposed to stem the profitability of this economy was proposed by Li et al. in 2009 [122]. This method suggests that by introducing virtual bots into the system, an uncertainty level in the performance of the network is introduced, e.g., a botmaster needs a specific amount of active nodes to perform a DDoS attack on a server and if many of the nodes currently active in the botnet are fake, the goal cannot be accomplished. This makes the task of achieving the optimal botnet attack size infeasible for botnet operators and will ultimately effect their profitability.

4.4.1 Valuation

A significant underground economy of selling, trading or renting botnets has developed in recent years. A botnet with 10,000 infected machines can fetch approximately $300-$800, depending on the geolocation of the nodes and the quality of the nodes. This quality is determined by the nodes being infected solely by a single botnet client and the nodes' uptime and internet connection speeds. Botnets with infected nodes based in the United States are the most valuable at $125 per 1,000, with European based botnets valued at $35 per 1,000 and Asian botnets valued at just $13 per 1,000 [123]. In 2010, Danchev found that the average price for renting a botnet is $67 for 24 hours and $9 for hourly access [124]. Often more money can be made through the renting of a botnet to multiple customers concurrently, referred to as "Botnet as a Service (BaaS)" [125].

4.4.2 Spamming

In 2010, it was estimated that over 89% of all emails sent were spam, resulting in over 262 billion spam emails being sent per day. In 2013, dealing with the volume of spam will cost over $338 billion in network bandwidth and infrastructure costs in 2013. The majority of spam originates from botnets and

it is estimated that 80-85% of all spam is produced by 6-10 huge botnets [126]. Almost all of this spam is illegally distributed under current laws in North America and Europe. Rao et al. estimate that the sending of spam is a $200 million per year business for the botmasters [127].

In 2011, Stone-Gross et al. analysed a popular underground web-based forum known as "Spamdot.biz" [123]. The forum required significant social engineering to gain access, with the authors requiring a reference from at least three existing members of the forum before they were granted access. It was found that this forum was used by almost 2,000 users to advertise their spamming services and to buy/sell information. E-mail address lists were worth between $25 and $50 per one million, contingent on the geolocation of the users and the proportion of addresses belonging to email providers with stronger spam filters, e.g., Gmail, Hotmail or Yahoo.

Kanich et al. found in one specific instance that a major spam campaign involving the sending of almost 350 million emails using the Storm botnet, only made $2,731.88 in revenue for the advertiser [128]. The campaign required over 75,000 active bots in the network to send the emails and resulted in 28 purchases from the associated online pharmacy website, or a conversion rate of just 0.0000081%. All but one of these purchases were for male pharmaceutical products, such as Viagra and Cialis, and the average purchase price was close to $100. The authors continue to estimate that the cost of such a campaign would be in the order of $25,000 and as a result, speculate that the botmasters of the Storm botnet may be the purveyors of the pharmacies advertised.

An obvious approach for ISPs and email providers to blocking spam emails is to refuse communication, or blacklist, known IP addresses that are found to be sending high volumes of messages. In the traditional spamming model, whereby a spammer hires a server or number of servers to send out the emails, this approach can prove very effective. However, blacklisting is not an efficient approach for blocking spam originating from infected nodes in a botnet. Any single node in the botnet is most likely a regular home or

business user who is likely to get assigned a new IP address regularly, due to the common ISP practice of DHCP based IP address allocation. As a result, blocking every IP address that is found guilty of sending spam will result in the email service provider ultimately blocking a high number of legitimate users (who innocently may have been assigned an IP address previously used by an infected bot).

4.4.3 Phishing

Phishing generally involves an attempt to trick the user of an infected machine to enter personal or confidential information through faux webpages [104]. These often convincing data harvesting webpages are injected into the user's regular browsing habits, popped up in their browser or OS or sent as links in emails. Phishing attacks are also commonly distributed by email spam. Webpages emulating those of a bank, online payment provider or lottery are common themes used in email based phishing attacks to entice the unsuspecting user into parting with valuable personal information.

Links have been found between distributed online phishing attacks collecting credit card information and the funding of terrorism. In one example, three men were arrested in the United Kingdom in 2008 and were found guilty of funding the terrorist organisation, al-Qaeda [100]. The trio were found to be in the possession of over 37,000 stolen credit card numbers, along with associated personal information from victims. They had made over $3.5 million in fraudulent charges and had purchased over 250 airline tickets.

4.4.4 Scamming the Scammers

In 2010, Herley et al. documented that a large proportion of the underground economy has evolved offering bogus botnet software, email addresses, botnets, etc., to unsuspecting criminals [129]. The presence of these scammers ultimately represents a tax on every "honest" transaction, where neither party

might be familiar with the other. It was also found that a two-tier underground economy now exists. The top tier consists of elite cybercriminals where their organisation, alliances and trust is established. At this tier, transactions take place between known or "reputable" criminals. The lower tier, generally conducted on IRC marketplaces, is occupied by criminal newcomers without any experiential skills or alliances and are easily cheated out of their money by fake sellers or "rippers", who have no intention of providing the goods or services offered.

4.5 Botnet Powered Attacks

As proven by the Anonymous "hacktivist" attacks in recent years, innocent victims' infected machines do not solely contribute to the the distributed power of botnet attacks. Regular Internet users with shared political or activist views can voluntarily decide to contribute their processing power to a collaborative cause. Partaking in the Anonymous attacks involved users downloading and configuring an open source network stress testing tool called "Low Orbit Ion Cannon" (LOIC). Alternatively, users can donate their computational power via a JavaScript-based version facilitating anyone who visits the site to participate in the attack [130]. Regular P2P file-sharing networks, such as BitTorrent, can also be manipulated by malicious users to aid in the execution of a DDoS attack through the exploitation of vulnerabilities in the protocol and operation of the network [131]. In a cyberwarfare scenario, it is conceivable that citizens of countries with limited computational infrastructure or supporters of terrorist organisations could similarly be called upon to donate their systems to aid in a collaborated attack on an enemy's infrastructure.

The potential for attacks originating from a growing number of sources is a concern for the security of many nations across the globe. In 2012, Amoroso et. al defined five possible motivations behind cyberattacks [132]:

1. Country-sponsored warfare – This is whereby national infrastructure is attacked by enemy cyber forces in an attempt to disable critical resources of the opposing country in a similar manner to traditional physical warfare. The intensity of this attack is only limited by the resources and devotion of the attacking nation. In a P2P botnet facilitated attack, citizens could voluntarily donate their computing power towards the national goals in a similar manner to the Anonymous attacks outlined above.

2. Terrorist Attack – Groups driven by terrorist motivations could quickly gain sufficient funding and expertise to conduct their attacks. Also in this scenario, regular Internet users could partake in a terrorist operation without requiring any skill or expertise by donating their regular computer equipment to the attack.

3. Commercially motivated attack – Competing companies might target their competitors' e-commerce infrastructure in order to prevent regular users from purchasing anything from their online stores. A popular e-commerce site being taken offline has the added effect of harming the victim company's reputation in their customers' eyes.

4. Financially driven criminal attack – These types of attacks could target individual computer users by recording their Internet banking details, online payment services and other financial services. Companies can also be targeted with extortion threats against their online infrastructure.

5. Hacking – This scenario generally involves an individual or group of hackers attacking targets motivated by little more than mischief or attaining online recognition of their achievements.

A number of popular botnet powered attacks are outlined in the following subsections:

4.5.1 Infection

Often botnet developers integrate some method of self-propagation into their design. Bots can scan for and infect more vulnerable computers with network or browser based exploits. Increasingly, web-based infection mechanisms have been observed online, whereby drone machines infect legitimate websites with a drive-by exploit and consequently, visitors to those websites can become infected [100]. These website based attacks are often targeted towards popular websites to try to infect as many web users as possible.

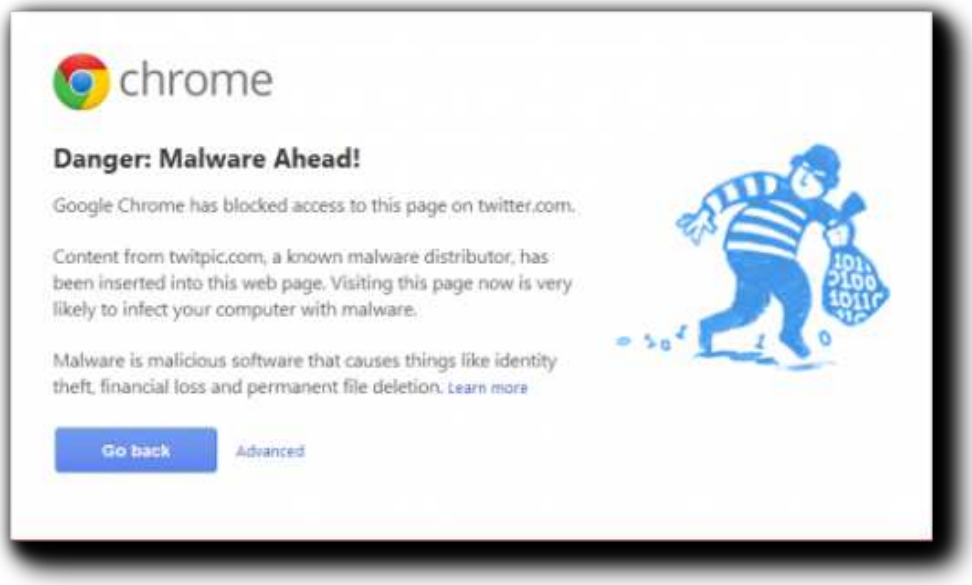

Figure 4.10: Google Chrome Malware Warning

The prevalence of web based malware distribution has resulted in most of the major web browsers implementing malware detection to aid in the protection of their users. A sample browser warning is shown in Figure 4.10.

4.5.2 Distributed Denial of Service Attacks (DDoS)

Distributed Denial of Service attacks are attempts to overload or monopolise a machine or web service resulting in the resource becoming unavailable for its

intended purpose. The first DDoS attack occurred in the summer of 1999 and ranged from several hundred to more than two thousand computers [133]. This type of attack can result in significant financial losses for their targets, as their web services are rendered useless for the duration of the attack. While many DDoS attacks result in the target being effectively offline for a certain period of time, "kinetic world" cyberattacks also exist through firmware code injection into physical hardware resulting in destroyed routers, firewalls, motherboards, etc.

The first DDoS attacks were motivated by petty online fights on IRC channels but soon shifted to monetary motivations with extortion of online gambling, e-commerce and pornography websites in 2003. These attacks evolved into politically motivated attacks against national infrastructures in 2007 [133].

In December 2010, the Anonymous "hacktivist" group launched "Operation Payback" in response to the controversy of the attempted take-down of the whistle-blowing website, Wikileaks [134]. The operation involved a coordinated DDoS attack on the financial organisations (e.g. MasterCard, Visa, Paypal), Wikileaks's DNS provider and political and legal websites involved in the controversy. LOIC is the network stress testing application that was used by Anonymous to accomplish its DDoS attacks. LOIC was utilised as a botnet whereby individual users download the client application and voluntarily contribute their computing resources to the collective attack of targets instigated by Anonymous.

```
!lazor default targethost=www.moneybookers.com subsite=/ speed=3
threads=15 method=tcp wait=false random=true checked=false
message=Sweet_dreams_from_AnonOPs port=80 start
```

Snippet 4.1: Example Low Orbit Ion Cannon Instruction

One example attack command sent by Anonymous to the LOIC bots targeting the online payment provider Moneybookers.com and the included parameters can be seen in Snippet 4.1 [130]. This particular attack was focused

on overloading the Moneybookers.com web server running on TCP port 80, with each bot running 15 concurrent threads to the server and repeating the request every 3 seconds without waiting for any reply from the server.

Botnet driven DDoS attacks are also commonly used for extortion. In this scenario, a website or service is sent a threatening demand and must pay extortion or else be faced with their website being taken offline. A sample extortion threat is shown in Figure 4.11 [124].

"Hello. If you want to continue having your site operational, you must pay us 10 000 rubles monthly. Attention! Starting as of DATE your site will be a subject to a DDoS attack. Your site will remain unavailable until you pay us. The first attack will involve 2,000 bots. If you contact the companies involved in the protection of DDoS-attacks and they begin to block our bots, we will increase the number of bots to 50 000, and the protection of 50 000 bots is very, very expensive.

You will also receive several bonuses. 1. 30% discount if you request DDoS attack on your competitors/enemies. Fair market value ddos attacks a simple site is about $ 100 per night, for you it will cost only 70 $ per day. 2. If we turn to your competitors / enemies, to make an attack on your site, then we deny them."

Figure 4.11: DDoS Extortion Example

4.5.3 Espionage

An infected machine is in complete control of the botmaster and often the intended targets of the criminal activity are the home or corporate owners of the compromised computer. Local documents, passwords, keystrokes, financial and personal information are all desirable material that is capable of being sent back to the controller of the bot. In a similar vein to the original trojan horse viruses described above, desktop screenshots and webcam snapshots have been observed getting relayed to the botmaster [100].

4.5.4 Proxies

Many botnet systems have proxy capabilities which effectively let the botmaster tunnel their internet traffic through one or more of the zombie machines. This can help facilitate anonymity for the botmaster and aid in

the evasion of capture from the authorities in a similar vein to anonymous proxies, as outlined in Section 3.7.1.

4.5.5 Clickthrough Fraud

"Clickthrough fraud", often referred to as just "click fraud", involves automatically gathering hits, advertising impressions and advertising clicks on specific websites operated by the cybercriminal. As each bot on the P2P network has a unique IP address, each infected machine appears to the advertising network provider (e.g. Google Adwords) as though it is any other unique visitor to the website. Generally a subset of the entire botnet will "choose" to click on the advertising available on the site, emulating regular visitor usage in an attempt to avoid detection. Advertisers are forced to trust that the advertising engine providers detect and prevent clickthrough fraud even though the engines still get paid for every undetected fraudulent click [135].

4.5.6 Cyber Warfare

Cyberattacks on critical domestic or wartime infrastructure, e.g. power, water, communication and emergency systems, could bring a country and much of its military coordination to its knees during a time of war. Next generation warfare will involve coordinated attacks on two fronts; both "traditional" or "kinetic" ground, sea and air based fighting and attempting to remotely destroy a nation's cyber-infrastructure.

In July 2009, several government and business websites in the United States and South Korea were reportedly invaded. Initial suspicion was focused on North Korea as a source of these attacks, though no conclusive evidence was discovered [136]. The rules and principles that govern physical warfare are largely dictated by easy to comprehend physical laws and limitations. Due to the fact that the majority of cyberwarfare attacks are conducted online,

physical limitations, such as distance between warring nations, becomes almost irrelevant [137].

4.6 Existing Detection Methods

As quickly as botnet technology is evolving, so too must the methodologies attempting to keep up to date with the latest botnet advancements. Primarily, the objective of any botnet investigation is to attempt to decipher the methods of communication used by the system. This is in order to eavesdrop on the botnet chatter in an attempt to record the manner with which the botnet propagates itself, what commands the botnet is executing, what systems are at risk and how many machines are infected. There are three main entry points to P2P botnet investigation [113]:

1. Deliberately infect a host and participate in the botnet. This is the most realistic scenario as a real machine is infected and, as a result, no flags should be raised to either the bot client or any other peers that an investigation is taking place. In this instance, the network traffic of the machine can be monitored and analysed.

2. Deliberately infect a virtual host (or multiples thereof). This allows multiple bot clients to run on the same physical machine allowing much more network traffic to be gathered in a shorter period of time. However, many modern bots have the ability to detect if their host is a virtual machine and may adjust their behaviour accordingly.

3. Create a crawler and mimic the protocol used by the botnet. In order for a crawler to be built, the bot itself will need to be completely reverse engineered. The crawler can then act as though it were a regular bot on the network to every other peer. This method awards the investigator much control over the network, from enumeration to forwarding bogus commands and potentially destroying the botnet.

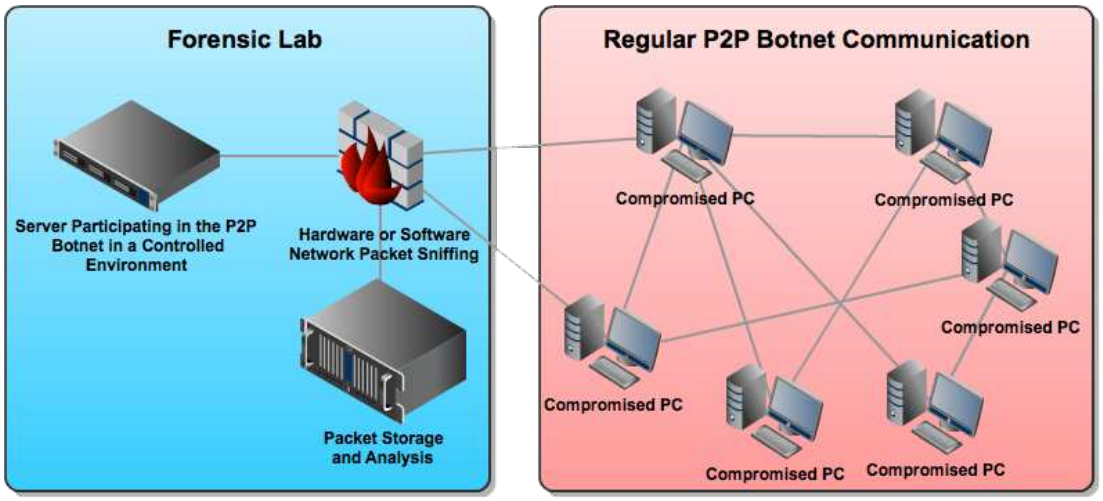

Figure 4.12: Typical Investigation Topology

Irrespective of the method used, the investigation will appear similar to that outlined in Figure 4.12. A client machine in a controlled, forensically sound environment will attempt to partake in the botnet. In order not to raise any flags to any built-in, counter-forensic measures to either the botnet client or any other peers on the network, this client machine must appear as any other regular infected machine. All network communication from that client machine can then be monitored, recorded and analysed.

In 2009, Feily et al. highlighted four main categories of detection available in botnet investigation [115]:

1. Signature based detection – This can only be used to aid in the detection of known bots and operates in a similar fashion to regular anti-virus signature detection, besides being applied to the identification of network traffic streams using an Intrusion Detection System.

2. Anomaly based detection – This attempts to identify botnet activity based on network anomalies such as high network latency, high volumes of traffic, suspicious port usage and other unusual system behaviour.

3. DNS based detection – Due to the prevalence of dynamic DNS (DDNS) providers being employed to avoid hardcoding C&C server IP addresses, the unusual querying of a DDNS provider may trigger detection.

4. Mining based detection – Identifying C&C based traffic may prove difficult, as they generally operate on commonly used ports, e.g., HTTP traffic on TCP port 80. C&C traffic is also generally quite infrequent and of low volume. As a result, data mining techniques, such as classification and clustering can be used efficiently to detect C&C traffic.

Detection Type	Unknown Bot Detection	Protocol & Structure Independent	Encrypted Bot Detection	Real-time Detection	Low False Positive
Signature	No	No	No	No	Yes
Anomaly	Yes	No	Yes	No	No
DNS	Yes	Yes	No	No	No
Mining	Yes	Yes	Yes	No	Yes

Table 4.1: Comparison of Botnet Detection Techniques

In 2013, Vania et al. published the Table 4.1 which presents the most up to date facts about the various detection methods outlined above [138]. From the data, it is clear that no single detection method is perfect and as a result, multiple methods should be deployed in any detection system.

4.6.1 Host Based Approach

In 2007, Nummipuro outlined the three main methods available for detecting and identifying P2P botnets on an infected host machine [139]:

1. Tracking Network Data – This involves tracking the remote machines that a specific process is in communication with. Communication with known C&C servers can aid in identifying a botnet system.

2. Analysing Network Data – This involves the analysis of the payload of individual packets to identify common botnet communication patterns.

3. Behaviour Based Identification – When a specific piece of malware is running on an infected machine, it calls on specific Microsoft Windows API functions. The behaviour of these calls, in combination with one or both of the above methods, can help identify the infection.

4.6.2 Hardware Based Approach

Honeypots are a common system used to detect security threats, collect malware and to understand the behaviours of malware and their perpetrators [104]. Honeypots are specially constructed computers or network traps which attract malicious attacks. However, advancements in botnet technology has resulted in more intelligent honeypot aware, self-destructing bots. From 2006, forensic researchers began documenting ways that a bot could detect that it was running in a honeypot [140]. The earliest detection methods were based on the assumption that security and forensic professionals have liability constraints, such that they cannot allow their infected honeypots to participate in real (or too many real) attacks. Subsequently honeypot detection methods expanded to include firewall, anti-virus and virtual machine detection.

Deep packet inspection (as outlined in Section 3.6.2) is another common hardware based approach and may be used in conjunction with honeypots.

4.7 Investigation Types

4.7.1 Anatomy

Investigating the anatomy of a particular botnet includes analysis of the behaviour of the bot binary and analysis of the network communication patterns. This type of investigation attempts to classify the botnet as centralised/decentralised, client-server or P2P based command and control. The classification can continue past the architecture of the system to cover some of its counter-detection and anti-forensic techniques. For example, Goel et al. discovered that "Agobot" had a built in defence mechanism to kill an upgradable list of over 610 anti-virus programs [141].

4.7.2 Wide-Area Measurement

Wide-area measurement investigations concentrates on attempting to enumerate the population of the botnet, the bandwidth usage, the computational capabilities as well as the commands being issued. Gathering the population of a botnet is a non-trivial task, as the number of nodes connecting to a C&C server may only ever count for a small proportion of the total infected nodes. There are two definitions of a botnet's size, as specified by Rejab et al. [142]:

1. Footprint – This indicates the aggregated total number of machines that have been compromised over time.

2. Live Population – This measure denotes the number of compromised machines that are concurrently in communication with the C&C server.

A relatively straightforward method for measuring the size of a botnet is to run a bot on a deliberately infected machine and monitor the resultant traffic. The number of IP addresses the infected node is in communication with can be easily counted, having eliminated all non-botnet related network traffic. While it would be unsafe to assume that a single node will ultimately communicate with every other node over any reasonable timeframe; increasing the number of infected machines (physically or virtually) and amalgamating the results should lead to a more accurate representation.

Byung et al. proposed in 2009 a methodology for improving botnet size estimates through the implementation of a botnet crawler, called Passive P2P Monitor (PPM) [143]. PPM acts as though it were the same as any other node on the network by implementing the "Overnet Protocol", as explained below. This method involves mimicking the functionality of a regular bot with regards to maintaining the DHT. For each peer the crawler connects to, it can ask for a list of all known peers. In this manner, a list of all known peers on the network can be compiled. This approach closely resembles that employed in the crawling of P2P file-sharing networks described in Section 3.6.1.

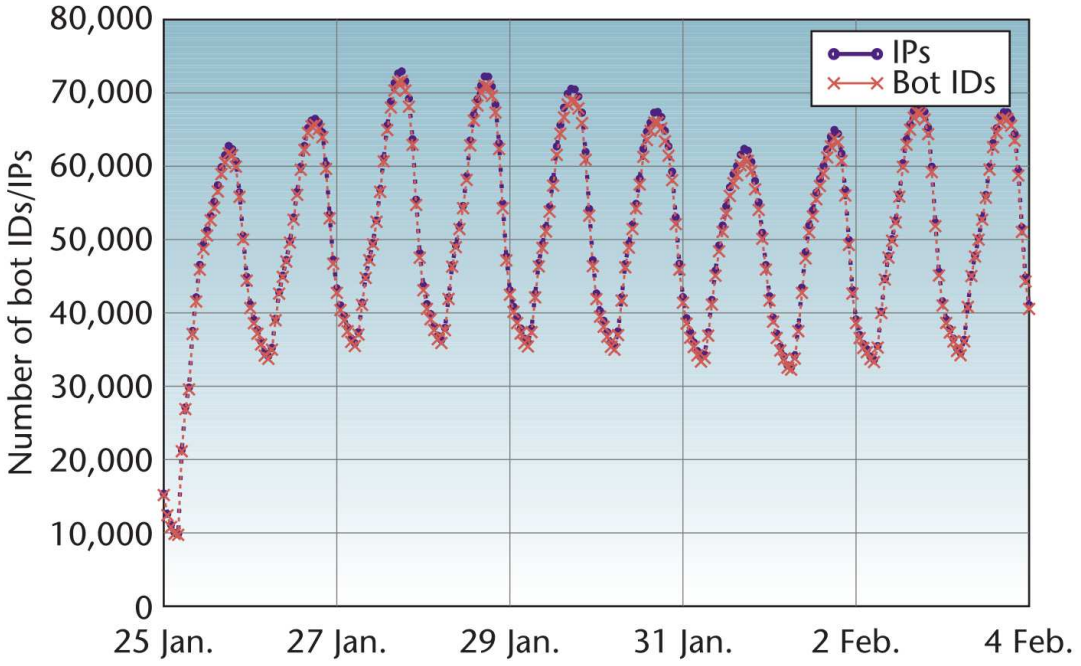

Figure 4.13: Unique Bot IDs and IP Addresses per Hour

4.7.3 Takeover

Botnet takeover involves a third party gaining control of a botnet from its owner. This third party could be law enforcement, researchers or another botmaster. Once control of the botnet has been gained, the new botmaster is able to issue commands, update configurations and operate the botnet as desired. In 2009, Stone-Gross et al. successfully took over the Torpig botnet for 10 days [125]. During this time, the researchers identified more than 180,000 compromised machines and were sent over 70GB of automatically harvested personal information.

The number of discovered unique Torpig bot IDs and corresponding number of IP addresses can be seen in Figure 4.13 [144]. The discrepancy between the number of bots and IP addresses found is accountable by network effects such as DHCP churn and NAT. Stone-Gross et al. discovered 182,914 different bot IDs originating from 1,247,642 distinct IP addresses over the ten day controlling window.

4.7.4 Investigation Obstacles

Many of the obstacles facing an investigation on P2P botnets are shared by the investigation of any P2P network, documented or undocumented [6]:

1. DHCP – Due to a typical lease from an Internet service provider lasting in the order of 2-7 days, dynamic reallocation of the same IP address may result in two or more infected machines participating in the network appearing as a single peer.

2. Proxy servers – Similar to the issue caused by DHCP, any bots that access the Internet through a transparent or anonymous proxy server will also appear as a single bot.

3. NAT – Numerous machines behind a shared router may appear to the outside world as a single machine, as a result of sharing a single external IP address.

4. Encrypted Communication – Should the bot employ encrypted communication, the only method available for investigation is to attempt to reverse engineer the bot. The decryption key for any incoming commands must be stored within the bot's client.

5. Difficulty in Take Down – Fighting back against botnets is often a matter of discovering a vulnerability in the design. Traditionally, this has meant attempting to take down their centralised C&C server [145]. However, with the popularity of employing a fully decentralised network design, the ability to take down a botnet has been made considerably more difficult. Should the bot be reverse engineered, it is possible that the botnet could be destroyed or "imploded", i.e., through the issuing of an uninstall command to each infected node.

4.8 Case Studies

4.8.1 Nugache

Nugache used a list of 22 hardcoded IP addresses which each newly infected host attempted to connect to [118]. These 22 hosts maintained a list of active nodes, which they shared with each new node. The list of active nodes that any given peer maintained always contained the initial 22 hosts, along with any newly shared active IP addresses. The weakness of this design is that once these 22 hardcoded nodes are taken down, no newly connecting peer will be able to gather its initial list of active peers to communicate with. The Nugache botnet communicates across its own bespoke network protocol. The communication between each node is not encrypted, but there is a degree of obfuscation employed [121]. In June 2007, Dittrich et al, discovered that there were at least 6,000 active IP:port pairs in the Nugache botnet at any given time and a total infected footprint of nearly 11,000 IP:port pairs [113]. It was also discovered that Nugache propagated itself through two remotely accessible exploits in the Window's LSASS and RPC-DCOM services, emailing copies of itself to targets found in the Windows Address Book and via instant messenger clients, such as AIM and MSN Messenger.

4.8.2 Storm

The "Storm" botnet, first discovered in January 2007 [146], was the first botnet discovered that utilised a P2P protocol. It spread through a mixture of social engineering and exploiting vulnerabilities in Windows XP and Windows 2000. The social engineering aspect of the worm was realised through the sending of topical, newsworthy emails with attachments or links to videos and pictures, which were in fact executables to infect the user's machine. When it infected any given machine, it would disable the Windows firewall and open a number of TCP and UDP ports. Communication in the Storm botnet relies on the "Overnet Protocol". Once the malware was installed and the host machine

was configured, it would then bootstrap onto the Overnet network and start listening for commands. The worm was also engineered to aggressively attack anyone who attempted to reverse engineer it [147].

The Overnet Protocol utilises a DHT, storing the IP addresses and unique IDs of each active peer in the network [148]. It is based on the Kademlia algorithm, similarly to BitTorrent [147]. Kademlia assigns a 160-bit hash ID to each participating peer on the network. Each peer maintains a local routing table consisting of the binding values for other peers that are ”close” to their own ID. In order to bootstrap onto the DHT, the Storm bot has a hardcoded list of over one hundred peers it can connect to [139].

4.8.3 Waledec

The Waledec botnet has striking similarities to the Storm botnet, while simultaneously exhibiting unique refinements to aid in network uptime and performance, but in part more vulnerable to attack. Waledec follows a hierarchical architecture design. The lowest level were the spammer nodes, which, as their name implies, were responsible for sending spam emails. These spammer nodes communicated exclusively with repeater nodes or super-nodes. These super-nodes, in turn, were in control of the communication with the spammer nodes and would receive their commands from the next level up, known as the sub-controllers [149]. The highest level in the hierarchy, the C&C server, only communicated directly with these sub-controllers.

Similar to the Storm botnet, the Waledec binary contains a list of hardcoded nodes to use to bootstrap onto the network. In the event of all of these hardcoded nodes being offline, a dynamic URL is also included in the binary to fall back on HTTP to receive commands. Due to this HTTP fall-back, this category of botnet can be referred to as a “HTTP2P” botnet [112]. Communication between nodes is encrypted, initially using a constant key for all nodes, which later evolved into a frequently changing key, which would

be created at the C&C server and passed down the hierarchy [149].

4.8.4 Zeus

The Zeus botnet is one of the largest botnets in the world [**?**]. Zeus uses an entirely decentralised P2P architecture and, like the majority of other botnets, it originally only operated on Microsoft Windows, but variants have been discovered infecting Blackberry and Android mobile phones [150]. An infected machine attempts to connect to its C&C channel by bootstrapping on to any one of hundreds of predefined nodes. The command and control channel consists of many thousands of server nodes [151]. Its purpose is primarily to spy on the users of infected machines, with the intent of gaining financial benefits for the botmaster [152]. It has the ability to log any information entered by the unsuspecting user, as well as injecting data displayed on visited web pages. The targeted information includes email addresses, passwords, online banking accounts, credit card details and transaction authentication numbers.

4.8.5 Stuxnet

Discovered in June 2010, Stuxnet was the first cyberwarfare weapon targeting physical infrastructure [120]. It is believed to have been developed by the United States and Israel in an attack against a nuclear power plant and processing facility in Natanz, Iran, although no conclusive evidence has been discovered about who lies responsible [153]. Stuxnet was not remotely controlled; it was completely stand-alone and spread itself without any further interaction. The C&C servers that Stuxnet contacted while in operation appear to have primarily been used for recording evidence of compromise. It spread via a Microsoft Windows vulnerability and targeted Siemens industrial software and equipment. This equipment included electronic controllers for pumps, valves, thermometers, motors and tachometers used in the nuclear facility. During the attacks in 2010, Stuxnet temporarily shut down nearly

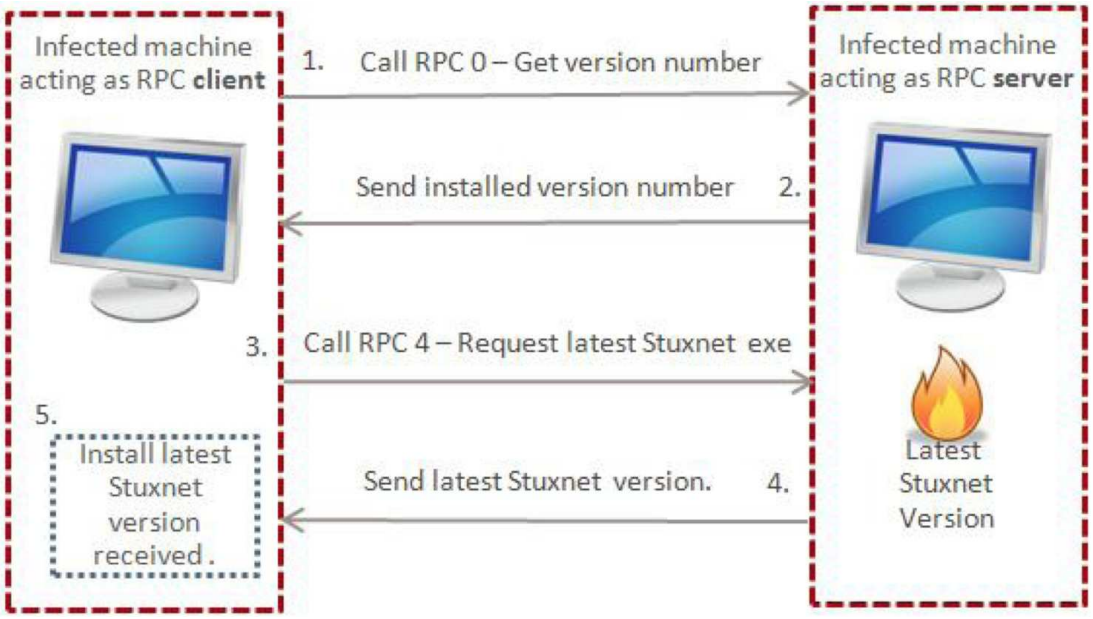

Figure 4.14: Example of an Old Client Requesting Latest Version of Stuxnet via P2P

1,000 of the 5,000 centrifuges Natanz had in operation purifying uranium [153].

The Stuxnet dropper client was designed to spread itself to as many machines as possible and spread through network shares, infecting removable storage devices and exploitation of software vulnerabilities. It utilised P2P communication for updating itself, as can be seen in Figure 4.14 [154]. The P2P component had two parts, namely an RPC server and client. When the malicious code compromises any machine it starts the RPC server. Through P2P chatter, any other infected machines on the network can update themselves from any peers that are running code with a higher version number.

4.9 Ethics of Botnet Mitigation/Takeover

With a lack of precise legal guidance in the investigation of botnets, much of the decisions required in botnet investigation are left in the hands of investigators to make the right ethical choices. When Stone-Gross et al. took

over the Torpig botnet in 2009 as described in Section 4.7.3, they used two principles to guide their investigation [125]:

1. The compromised botnet should be operated so that any harm and/or damage to victims and targets of attacks would be minimised.

2. The compromised botnet should collect enough information to enable notification and remediation of affected parties.

4.10 Summary and Discussion

Factors	Centralised (IRC,HTTP)	Hybrid DDNS	Peer-to-Peer P2P
Detection	Easy	Medium	Hard
Resilience	Low	Fairly High	Very High
Latency	Low	Medium	Fairly Hard
Traceback	Fairly Hard	Hard	Very Hard
Complexity	Easy	High	Medium
Experience	Very High	None	Medium

Table 4.2: Comparison of Botnet C&C Architectures

In this chapter, the evolution of botnet design culminating in Peer-to-Peer architectures was introduced. Table 4.2 presents a summary of the various C&C botnet architectures available and the corresponding difficulties associated with each of the considerations in a botnet developer's design decisions [138].

Ultimately, the P2P botnet topology is a desirable option to choose for botmasters, as it affords them an additional level of anonymity when conducting their crimes. The ideal design for a P2P botnet is one that is completely decentralised, utilises unique encryption methods and operates on a bespoke network protocol for communication. Investigation of such a botnet may prove particularly difficult. However, a combination of research, network monitoring, deep packet inspection and network crawling should result in successful, albeit more labour intensive, investigations. The fundamental requirement for any newly infected node (or a node coming online) to have

a starting point to bootstrap onto the P2P network and discover other active nodes will always leave an avenue of investigation open for the digital investigator.

CHAPTER

FIVE

UNIVERSAL P2P NETWORK INVESTIGATION FRAMEWORK

5.1 Introduction

As outlined in Chapters 3 and 4, P2P networks are widely used as a low-overhead, efficient, self-maintaining, distributed alternative to the traditional client/server models, across a broad range of areas. As a result of these desirable attributes, the technology also lends itself well to being utilised for malicious purposes, due to the minimal setup and maintenance costs involved. However, the investigation of these networks is often a cumbersome process with significant duplication of efforts from investigative bodies.

Since P2P networking has become mainstream, the technology has been deployed across a broad range of systems and services. While the level of variation in topologies is significant, all P2P networks must share a number of common features:

1. The capability to bootstrap onto the network – When a new node attempts to join the network, it must be able to contact at least one other active node in the network. Depending on the network design, this may take the form of a hardcoded list of active nodes (typical in a

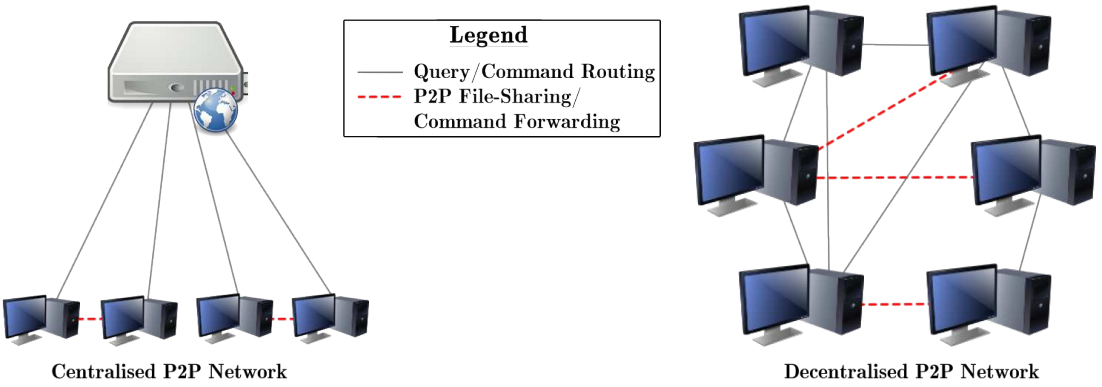

Figure 5.1: A Comparison of Centralised (left) and Decentralised (right) P2P Network Architectures.

decentralised topology) or a list of bootstrapping servers (typical in a centralised topology) [155].

2. Maintenance of a Live Record of Active Nodes – In a decentralised network, the peers themselves must all contribute to the recording of active nodes on the network. No single peer has the entire list, with each peer contributing to a collective distributed database, typically a DHT. In a centralised design, this duty falls on the controlling server. As each new node comes online, it announces its presence to the centralised list and requests a list of other active peers to begin working.

3. Query/Instruction/File Propagation – In order for a P2P network to fulfil whatever the purpose it was designed for, intra-peer communication is requisite. As a result of this necessity, each peer must be able to receive requests or commands and pass these communications onto other known peers.

4. Software Maintenance – The P2P enabled software itself can quickly become outdated. The upgrade process must be simple to perform, while maintaining node uptime. While newer versions of the application might have additional functionality, it must also ensure backwards compatibility otherwise the network as a whole may suffer.

In this chapter, a framework is introduced which enables forensic investigators and researchers to fast-track the investigation of any P2P network. The

framework exploits many of the common attributes of these networks, as outlined above. As it can be seen in Figure 5.1, each node on a P2P network has two main functions:

1. The first function involves fulfilling the objective of the P2P network, e.g., file sharing, query/command routing, etc.

2. The second involves participating in the maintenance of the network itself – it is aware of a number of other nodes, and is in communication with a subset of the overall network population.

This framework can aid in the discovery of P2P communication and attempts to identify common communications, e.g., peer discovery, query/command/file propagation, etc. Even within the same P2P application, different versions may result in different traffic patterns. As the usage of UP2PNIF increases, the database of identifiable traffic patterns and the value of using the framework will become larger.

Irrespective of what P2P network is being investigated, there is a similar methodology for its investigation. Due to the commonalities in the design of every P2P network design, a common forensic investigative process can be created. Each node requires some method of joining the network, discovering active peers, receiving commands/queries and some procedure defined for query/command execution. The proposed solution described in Section 5.6 outlines a five step investigative process.

5.2 Technical and Legal Framework Requirements

The design of any digital forensic investigation and evidence acquisition framework, such as that described as part of this thesis, must include consideration of a number of technical and legal requirements. These requirements include:

- *Verifiability* – All evidence collected using the system must be a true and verifiable representation if the original source. The framework must interact with the network and potential evidence in a forensically sound manner. All evidence interactions must be controllable, recordable and reproducible in order to ensure its admissibility to court, i.e., compliant with the Daubert Test, as outlined in Section 2.8.

- *Compatibility* – The framework should be capable of working on a variety of computer systems built on many types of different categories of hardware, e.g., servers, workstations, laptops, netbooks, etc. The manufacturer of the computer system should have no influence over the compatibility or the output of the system. The framework should have the ability to aid in the identification and investigation of any P2P network – documented or undocumented, centralised or decentralised, irrespective of size or counter-forensic features. Minimal customisation should be required for the system to operate on bespoke P2P network investigation devices.

- *Cost efficiency* – The cost of implementing the system for researchers, law enforcement agencies or private digital investigators should not be prohibitive. Current network investigation tools and software can be prohibitively expensive to implement, e.g., a bespoke network investigation tool might take thousands of man hours to produce or the significant costs involved in investing in deep packet inspection hardware devices. Due to the above compatibility requirement, the cost of conducting an investigation should merely amount to the bandwidth used during the investigation's lifetime.

- *Usability* – Any client application built on top of the framework should not be overly complicated to use. The target user groups and their technical knowledge, i.e., mainly law enforcement officers, should be considered when designing the system. Any tools designed should require minimal training to use. The creation of a new network signature

should be intuitive for the user to include in the reference database.

- *Scalability* – The framework should be able to scale up to any required size. The use of a globally maintained network signature database should merely be used as a centralised reference repository. Concurrent investigations should be possible using a distributed network of investigation servers. Moore's Law has been commonly used to predict the progression of computer equipment since 1965 [156], and can be empirically used to predict the increasing prevalence and computational power of P2P systems, e.g. botnets. Computer processing power, storage capabilities and network speeds will all vastly increase in the future. The framework should be built in such a manner that it is capable of taking advantage of this predictable performance advancement.

- *Extensibility* – The system must have the ability to be updated with new extensions, such as conforming to a specific digital forensic evidence storage format as outlined in 2.5. The framework should allow for the investigation of any P2P networks through the integration of a formal definition of each network's characteristics and communication protocols. The system should have the ability to be used to investigate multiple networks simultaneously. This means that the system must be capable of collecting evidence from numerous geographically separated sources simultaneously.

- *Reliability* – The system must be reliable. Should an individual instance of the tool go offline, the workload should be capable of being redistributed to another instance. If the network goes down for any individual node, the system should fail cleanly, i.e., existing gathered evidence should be saved to disk before entering a "sleep" state until network connectivity returns.

- *Security* – As with any Internet enabled application, the system should be secure. Scheduling and distribution of jobs should be conducted through secure, verified channels ensuring protection against

a “man-in-the-middle” attack.

- *Maintainability* – Functional components of the system may periodically require maintenance or updates to confirm to new requirements/specifications as the system evolves over time. The system should be implemented in such a manner that easy updating can occur with minimal downtime.

5.3 Architecture

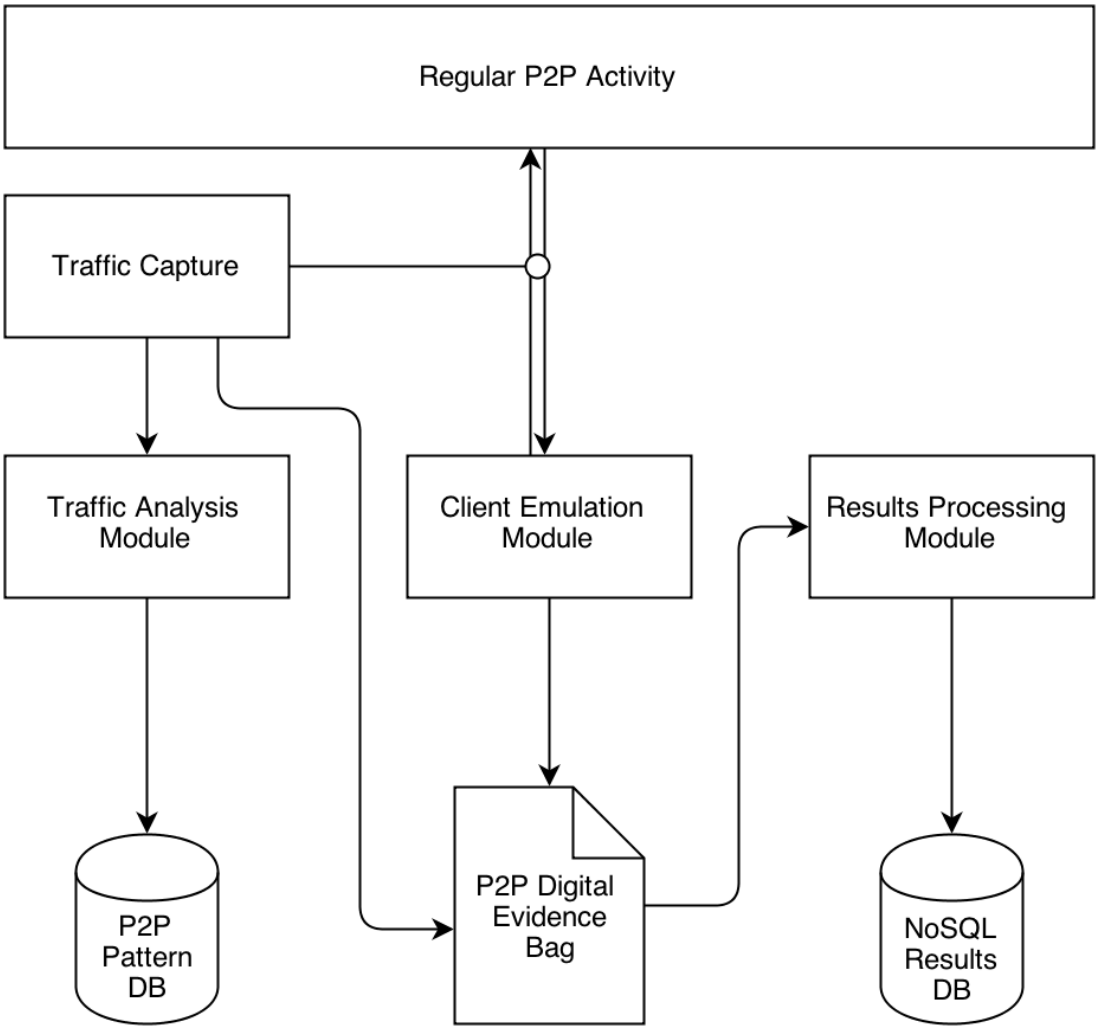

Figure 5.2: UP2PNIF architecture with the regular P2P activity on the top and the modules of UP2PNIF on the bottom.

5.3.1 Traffic Collection Module

The UP2PNIF architecture is designed to significantly reduce the time required to commence a P2P network investigation. Traditionally, in order to investigate a network, an entirely new network specific crawler has to be developed from scratch. Exploiting the commonality between different P2P networks, as outlined in Section 5.1, allows for a more efficient, less costly (in terms of both time and money) investigations.

This module monitors the network traffic of a specific machine, or a group of machines. The packet sniffing is conducted using "libpcap" (or its windows alternative, "winpcap") [157]. This module is also responsible for packaging the collected data streams and associated metadata into a custom P2P digital evidence bag (as outlined in Section 5.5) and, if necessary, securely transferring the evidence to an external storage device or network/cloud based storage.

5.3.2 Traffic Pattern Database

This component stores the patterns of known networks. The types of metadata stored for each network include common hostnames and IP addresses, peer discovery methods and frequency of updates, common commands, update methods, etc. This database is used for referencing by the analysis module to aid in the identification of discovered unknown network traffic.

5.3.3 Traffic Analysis Module

The module analyses the collected network packets from the traffic collection module. The frequency, content/pattern and destination of the packets contribute to the identification of the specific P2P traffic. In order for a P2P network to function, each peer must regularly check-in with a centralised server or with other active peers at specific intervals. Each suspected packet can be compared to the above database of known network usage patterns. This is particularly useful in the identification of botnets as each specific

botnet software system can be used by multiple botmasters in the creation of many separate networks.

5.3.4 Client Emulation Module

An intelligent client application is capable of performing various different forensic investigations. Depending on the specific network, it is possible to conduct each of the following investigations as required by the case at hand:

1. Network Enumeration – This investigation concentrates on attempting to enumerate the population of the entire network, as well as the combined bandwidth and computational power. Gathering the population of a network is traditionally a non-trivial task, as the number of nodes simultaneously connecting to any one node generally only accounts for a small subsection of the entire network. The client emulation module overcomes this limitation by intelligently amplifying regular client usage.

2. Network Usage – This investigation is focused on finding out what the network is being used for. In the case of P2P file-sharing, this type of investigation would be targeted towards the identification of the content being distributed and the logging of infringing IP addresses. In the case of a P2P botnet, the investigation might be targeted at finding out what commands each node receives, i.e., what is the crime the botmaster has chosen to exploit his botnet for?

3. Network Anatomy/Modelling – This investigation is focused on attempting to understand the design and structure of the network and client software. Based on the gathered evidence, it should be possible to extrapolate the network's topology. The results obtained can help in determining whether the network is centralised, decentralised or a hybrid, the frequency of intra-peer communication, the contribution of

each node to the maintenance of a DHT, the attack vector utilised by the malware, etc.

Through the emulation and amplification of regular P2P client usage, the machine conducting the investigation appears to each individual peers as though it is any other regular peer using the network. Through not overly querying any single node on the network, the chances of the investigation being discovered by the network as a whole are low.

5.3.5 Results Processing Module

The results processor is responsible for processing the generated evidence bags into a NoSQL database. To perform this processing, each gathered IP address is geolocated. The identification of each IP address is recorded as being active at an individual crawl level, content/file level, investigation level (potentially involving multiple pieces of content) and at an ISP level. This facilitates the results and analysis outlined in Chapter 6. This module is also responsible for producing some of the graphical representations created to summarise the investigation, as seen in Appendix A.

5.4 Modular P2P Network Signature

As stated above, the benefit of the proposed framework is to facilitate the investigation of any P2P network through the exploitation of the common characteristics of each of the networks. In order for the framework to function as part of a target network, the network signature must be defined. This signature consists of two separate pieces of network specific information: both a network settings configuration and a protocol definition, as outlined in the following subsections:

5.4.1 Network Configuration

In order for the system to be capable of investigating any given P2P network, the bespoke protocol characteristics and communication methods need to be recorded in an understanding format. In order for a specific P2P network to be compatible with the UP2PNIF system, the following information is required:

1. File-sharing (*boolean*) – If the network is a P2P file-sharing network, its primary purpose is the distribution of files. As a result, a range of specific investigation types should be enabled for this network in the framework, such as those required for copyright infringement detection, the downloading of content from a peer, etc.

2. Botnet (*boolean*) – If the network is a P2P botnet, the investigation can attempt to enumerate the network or record and identify the commands that the botmaster is issuing.

3. Centralised (*boolean*) – If the network has a centralised component, it is not sufficient to assume that a network with the "centralised" flag set to *true* would automatically have the proceeding "decentralised" flag set to *false*. Hybrid networks (such as BitTorrent) have both a centralised and a decentralised component and as a result would have both flags set to *true*.

4. Decentralised (*boolean*) – Networks having a decentralised component would generally employ a distributed hash table for maintaining the list of active nodes.

5. Configuration File (*boolean*) – This flag lets the system know whether to expect a network configuration file, e.g., a "*.torrent" file on the BitTorrent network or a "*.bin" file for the Zeus botnet. The file itself and the location of the file are stored in the signature database.

6. Encrypted (*boolean*) – If the network is encrypted, the system would need the key to be supplied before the investigation can commence.

Table 5.1: BitTorrent Network Communication Format

Feature	Value
httpFormat	%httpURL%?info_hash=%fileID%&peer_id=%peerID% &port=6881&numwant=200&compact=1&uploaded=0 &downloaded=0&left=0
pexFormat	%peerIP%:%peerPort%?param&param
dhtFormat	%dhtIP%?param&param

Traditional malware reverse engineering techniques might be employed by the investigator to extract this required information from the memory of an infected machine.

7. HTTP Peer Discovery (*boolean*) – In the case of a centralised network, commonly the centralised server will serve a list of known active peers responding to a HTTP request. As part of the response or as part of the client's configuration, there will likely be an option outlining the allowable frequency of this request to reduce server load.

8. Peer Exchange (*boolean*) – When queries/orders/files are passed from peer to peer, often the networks facilitate the communicating peers to exchange their local lists of other active nodes. In this manner, the regular operation of the network helps to ensure that each active node has an up-to-date list of other active nodes. Similarly to the preceding configuration option, there will generally be an associated allowable update frequency for this exchange.

9. Distributed Hash Table (*String*) – This value will outline the type of DHT involved. The system should include an implementation for communicating with each of the known DHT variants such as Kademlia, Pastry, Chord, etc.

5.5 P2P Digital Evidence Bag

The collection and handling of court admissible evidence is a fundamental component of any digital forensic investigation. The procedures for handling digital evidence take much of their influence from the established policies for the collection of physical evidence. However, due to the obvious differences in dealing with non-physical evidence, a number of extra policies and procedures are required.

When dealing with physical forensic evidence, the commonly used handling procedure is the "chain of custody" [28]. The chain of custody commences at the crime scene where the evidence is collected. Here, the investigator collects any evidence he finds and places it into an evidence bag. This evidence bag will be sealed to avoid any contamination from external sources and signed by the officer who will detail some facts about the evidence, e.g., description of evidence, location it was found, date and time found, etc. The chain of custody will then be updated again when the evidence is checked into the evidence store in the forensic laboratory. When it comes to analysing the evidence, it will be checked out to the forensic analyst's custody, and any modification to the evidence required to facilitate the investigation, e.g., taking a sample from a collected fibre to determine its origin or identification, etc., will be logged and documented.

The procedures outlined above for physical evidence need to be extended for digital evidence acquisition and analysis. Due to the fact that digital evidence is generally analysed on forensic workstations, most of the above sequences can be automated into concise logging of all interactions. During a digital investigation, there is no requirement to modify the existing evidence in any way. This is because all analysis is conducted on an image of the original source, and any discovered evidence can be extracted from this image, documented and stored separately to both the original source and the copied image. It is imperative when dealing with all types of evidence that all procedures used are reliable, reproducible and verifiable. In order for

evidence to be court admissible, it must pass the legal criteria for the locality in which the court case is being heard, as outlined in greater detail in Section 2.8.

A new P2P focused digital evidence bag is proposed to specifically deal with evidence gathered from any tools built upon the UP2PNIF framework. This bag incorporates the bit-by-bit network stream captured during the investigation and on-the-fly metadata result generation to aid in expedited identification and analysis of captured evidence. The precise format of this metadata result file is outlined in Section 5.5.3.

Evidence collection from the framework will fall into two main categories; a bit-by-bit copy of all network traffic and processed evidence resulting from participating in, or crawling the network, as outlined in Sections 5.5.2 and 5.5.3.

5.5.1 Forensic Integrity

Due to the sensitive nature of digital evidence collection, it is imperative that the data collected by any forensic tool is completely verifiable and identical to the original source. This integrity is ensured in the UP2PNIF system through the implementation of a new P2P focused digital evidence bag. This evidence bag is capable of storing all relevant information, e.g., IP addresses, packets, running processes, etc. Once a network traffic is collected, each packet is time-stamped and logged. The time-stamping facilitates real-time event reconstruction packet by packet, emulating the original traffic.

Forensic integrity is insured in the UP2PNIF system through the implementation of regular hash checking on the data being collected using SHA-512, a 512-bit secure hashing algorithm. The system collects a stream of information, the stream itself is hashed and both are stored on the external drive or can be uploaded to secure cloud storage. During the transmission process, the integrity of each of the segments being transferred is maintained due to a SHA-512 hash being computed as the segment is being transmitted.

Server-side, once the transmission is completed, a SHA-512 hash is taken on the segment and verified against the original. If these hashes do not match, i.e., the integrity of that segment has been compromised in transmission, a failure notification is sent to the client, which queues that segment up again for retransmission.

5.5.2 Network Packet Evidence Storage

The "raw" data storage format outlined in Section 2.5.2 was chosen as the storage format for all network evidence collected using the UP2PNIF system. This format was chosen for a number of reasons:

1. The raw format for storing data is the de facto standard for all digital evidence acquisition tools, whether they operate on physical storage media or network traffic [24]. While some tools may have their own proprietary standards, every tool has an option to capture data using the raw format.

2. All digital/network evidence analysis tools are capable of reading and analysing the evidence contained in a raw format file.

3. Due to the fact that the raw format is an exact bit-by-bit copy of the original evidence, it lends itself well to being split into small segment sizes, as required for transmission of the collected evidence in a live forensic capture scenario.

4. Acquiring a network traffic copy using the raw format requires the least amount of processing power from the client's side. This can be particularly advantageous when collecting evidence from low-powered computers, e.g., older computers, netbooks, etc.

The UP2PNIF system also stores some metadata alongside the evidence collected such as disk information (unique disk identifier, size, partition

information, hash sum), number of segments used to transfer the image and associated hash sums and time stamps of transmission.

5.5.3 Common UP2PNIF Result Storage Format

The output from any P2P network investigation is a common UP2PNIF XML evidence storage format. Due to the commonality of features across P2P networks, a standardised evidence results file is produced as the output from the system. This XML will include some investigation specific information, e.g., investigation ID number, start time, end time, network investigated, number of peers, etc., alongside the precise results of the investigation such as the content investigated (in the case of a copyright infringement focused investigation), each individual discovered peer's specific information, etc. This common result storage format is verifiable against the corresponding network byte stream captured during the investigation process. With respect to delivering court-admissible evidence, any incriminating results gathered from the UP2PNIF system are capable of being reproduced by a forensic expert. Figure 5.4 displays a truncated version of this common storage format.

```
<?xml version="1.0" encoding="UTF-8"?>
<crawl>
   <info>
      <crawl_id>1</crawl_id>
      <start_time>2013-01-01 00:00:00</start_time>
      <end_time>2013-01-01 00:00:00</end_time>
      <network>BitTorrent</network>
      <peer_count>10000</peer_count>
   </info>
   <configurationfile>
      <saveAs>Lady GaGa - The Fame Monster</saveAs>
      <name>CD101. Bad Romance - Lady GaGa -.mp3 ... </name>
      <pieceHashValuesHex> </pieceHashValuesHex>
      <totalLength>182457665</totalLength>
      <infoHashHex>
         00BA1B210942ED07160B30404B9EDE6F8A50AE36
      </infoHashHex>
      <announceURL>http://tracker.openbittorrent.com/announce</announceURL>
      <announceList>http://tracker.openbittorrent.com/announce</announceList>
      <comment></comment>
      <createdBy>uTorrent/1840</createdBy>
      <creationDate>2009-11-18 03:32:06</creationDate>
      <encoding>UTF-8</encoding>
      <pieceLength>262144</pieceLength>
   </configurationfile>
   <peers>
      <peer>
         <ip>109.165.156.246</ip>
         <discovery_time>2012-02-28 22:36:47</discovery_time>
         <discTracker>true</discTracker>
         <discDHT>false</discDHT>
         <discPEX>false</discPEX>
         <connected>false</connected>
         <port>28688</port>
         <PEXEncryption>false</PEXEncryption>
         <PEXHolepunch>false</PEXHolepunch>
         <PEXOutgoing>false</PEXOutgoing>
         <PEXSeed>false</PEXSeed>
         <PEXuTP>false</PEXuTP>
      </peer>
    ...


    ...
   </peers>
</crawl>
```

Figure 5.3: Common UP2PNIF XML Evidence Format. This example shows a truncated example XML file produced from an enumeration investigation of a BitTorrent swarm.

5.5.4 P2P Bag for Known Network Identification

In the scenario of attempting to investigate an existing, known network, the simplest form of the UP2PNIF evidence bag will contain the network traffic byte stream and an on-the-fly generated clustering metadata file highlighting similar packets and recording their frequencies in an attempt to aid in the identification of the various communication components. As a result of the network communication patterns, it is possible the network traffic can be correlated to a specific network. It is possible that the automated network identification can occur for clear cut cases. For uncertain identification cases, a network forensic specialist will be able to verify the collected, indexed traffic against known P2P network patterns. Once the network is identified, the investigation procedure can begin.

5.5.5 P2P Bag for Unknown Network Discovery

In the scenario of attempting to investigate a new, unknown network, similarly to that for the known network outlined above, this simple form of the UP2PNIF evidence bag will contain the network traffic byte stream and the corresponding metadata file. Ultimately, it will still require the expertise of a network forensic specialist to reverse engineer the protocol and deduce its characteristics. However, once the network is reverse engineered and the corresponding signature is created, as outlined in Section **??**, the investigation of the network can commence instantaneously.

5.5.6 Evidence Handling

The data collected using the UP2PNIF system is acquired directly from the network and recorded in a "raw" format for analysis. The recording and on-the-fly analysis requiring to expedite the investigation process is conducted without interfering with the original communication. This ensures that the origin of a particular packet cannot be compromised by any of the

operations of the UP2PNIF system. The captured data is stored directly to an external hard drive or can be sent to a remote/cloud server in an uncompressed format. This ensures that no potentially relevant evidence is lost. Due to the potentially large size of the data transmission, the evidence is sent from the client in segments and these segments are then recombined at the server side to produce the complete image of the original communication stream. As the captured data is treated as binary content during the splitting and recombination processes, the chance of either process compromising the integrity of the data is eliminated. This is proven at the server when the data is recombined and the hash sum of the final image is compared to the untouched hash sum of the original source.

5.5.7 Comparison to Existing Evidence Bags

Existing evidence bags and storage formats, outlined in detail in Section 2.5 are primarily focused on the storage of digital evidence collected from physical storage media. The evidence bag outlined above is focused on network forensics and is capable of fast-tracking the network identification and traffic analysis process for both existing and newly identified networks.

5.6 Investigation Methodology

The UP2PNIF framework is designed to aid the forensic investigator to conduct a broad range of P2P network investigations and, if necessary, in the expedited development of a P2P network investigation tool. Depending on whether the network is already known or not will result in a slightly different methodology being required, as outlined below.

The framework can help or entirely eliminate each of the "traditional" forensic P2P network investigation tool development steps required for the investigation of a new network. These steps and how the framework can aid in each step is outlined below:

1. *Assess* – Discover as much as possible about the network to be investigated. This can be achieved through the network sniffing of an active node in the network, e.g. deployed as part of a honeypot. The framework can record the network stream of an infected node and group together common network patterns. The number and frequency of each packet type facilitates with the identification of the communication methods. Although the amount known at the end of this step will be network dependent, an ideal discovery scenario would result in the following characteristics being known:

    - Topology of the network – This will include identifying characteristics such as whether the network is centralised/decentralised, node discovery methods, e.g., peer exchange, centralised server, DHT, etc.
    - Communication methods employed – This includes the protocol, the port ranges along with any required information to partake in the network such as maximum request frequencies, etc.
    - Common commands and queries available – This will incorporate a list of each type of communication required for the regular operation of the network, e.g., get peers, request command/file, forward command, etc.
    - Method of encryption (if any) – If the network traffic is capable of being reverse engineered and understood without needing to reverse engineer the P2P client itself; it will lead to a much quicker investigation. If encryption is employed in the network, then the P2P client will need to be reverse engineered to extract the required encryption/decryption procedure.
    - Bootstrapping methods – Each time a new node attempts to connect to the network, it must have a method of bootstrapping onto the network. In order for the system to aid in the investigation of any particular network, it too must be capable of bootstrapping onto the network using the same method as a regular client.

The required characteristics of each newly discovered network will be added to a centralised identification database. This will vastly decrease the time required for further investigation of the same network. Assuming that the network to be investigated exists in the database, the development step (step 2 as outlined below) would be skipped.

2. *Develop* – Traditionally, this step includes the development of a P2P client emulator for the specific network being investigated. This client must have the ability to bootstrap onto the network while appearing as a regular node to every other active node on the network. Regular client operations must also be available, including peer discovery and peer communication. Using UP2PNIF will facilitate fast investigation of any network. Once the network signature is determined, the investigation process can commence immediately.

3. *Monitor* – Depending on the desired investigation outcome, monitoring of the network over a specified time period will record the number of active peers, the network churn rate, the network reliability and uptime, the resource size available and the commands and queries received.

4. *Analyse* – Finally, the collected evidence gathered from the monitoring step can be analysed to determine the following:

    - The types of cybercrimes the network is being used for.
    - Any inherent weaknesses in the network, such as a single point of failure or the possibility of issuing imitation commands to the network.
    - Encryption methods employed.
    - If it is possible to identify the source of the commands. This will include the identification of ”super peers”, i.e., nodes responsible for distributing the commands across the network and the possible identification of the controlling machine or bot master. The time required to complete this step will be greatly reduced using the

UP2PNIF evidence bag, as the gathered metadata will facilitate the collation of much of the above information.

5. *Results* – The results from this investigation will enable law enforcement agents to investigate crimes on the Internet that may be otherwise flying below their radar. Any reporting on the investigation should clearly explain the complete methodology of the investigation, should the results need to be reproducible during court proceedings.

Due to the ability of the UP2PNIF system being capable of emulating and amplifying normal P2P client usage creates the possibility of on-the-fly processing and real-time result delivery. Partaking in the network itself helps to reduce the volume of network traffic recorded, focusing the investigation at an early stage on only the relevant evidence. This helps to negating some of the legal concerns resulting from the blanket collection of network traffic, discussed earlier in Section 2.9.

5.7 Verifying Data Integrity

The verification of the collected data against the original source is fundamental to the successful operation of the UP2PNIF system. The data integrity is insured in the UP2PNIF system through the implementation of regular hash checking on the network evidence being collected using SHA-512, a 512-bit secure hashing algorithm. Once a network capturing tool using UP2PNIF is deployed, a hash is taken periodically of the total network traffic collected.

If the data collected is to be transferred to remote storage, the integrity of each of the traffic segments being transferred is maintained due to a SHA-512 hash being computed as the segment is being transmitted. At the server side, once the transmission is complete, a SHA-512 hash is taken on the segment and verified against the original. If these hashes do not match, i.e. the integrity of that segment has been compromised in transmission, a failure notification is sent to the client, which queues that segment up again for retransmission.

5.7.1 Overhead for Ensuring Data Integrity

The requirement for any digital forensic evidence capturing tool to ensure integrity is paramount. While one of the primary objectives of UP2PNIF is to verify the integrity of the evidence, it is also important that any additional computational overhead is minimised. This is achieved by overlapping the computational tasks with data transmission.

When the first segment is transmitted completely to the server, the client immediately starts sending the second segment. When the server receives the first segment and its corresponding SHA-512 fingerprint (computed client-side), it then calculates a SHA-512 hash on the segment received and compares it to the client-side hash. If these hash values match, an acknowledgement is sent to the client to signify a successful transmission. This process is then repeated for the third and all subsequent segments. Due to the computational/transmission overlap, the additional cost of forensically verifying the evidence captured as part of the UP2PNIF system amounts to the time taken to compute the SHA-512 hash server-side of the last segment and compare this to the hash value taken client-side.

5.7.2 Hashing and Evidence Transmission Module

The hashing and evidence transmission module is the segment of the framework responsible for the external or remote copying of captured evidence. This module executes the file transmission and monitors the evidence copying process.

5.8 Resilience Against Detection

Through the emulation of regular client usage, the investigating machine appears to any neighbouring node as any other regular node on the network. This is achieved through emulating the behaviours of each peer on the

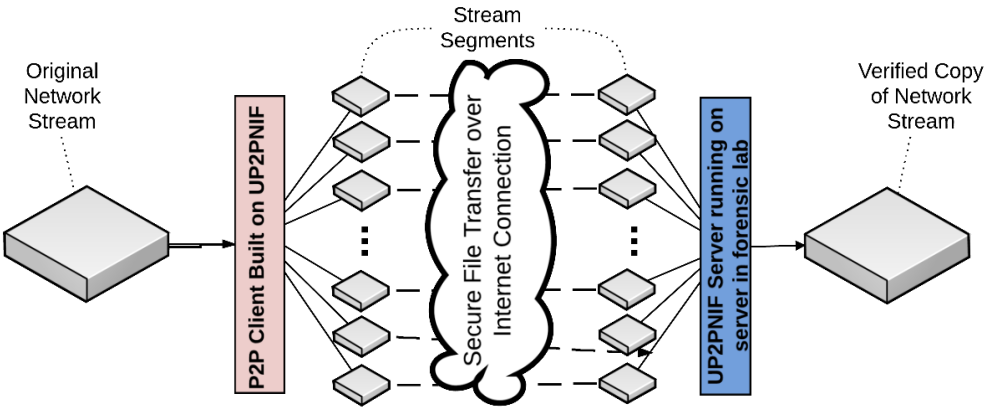

Figure 5.4: Overview of UP2PNIF evidence transmission architecture. This diagram shows the remote transmission from the UP2PNIF framework X = number of segments to transfer entire network stream.

network, i.e., obeying communication frequencies, forwarding commands, requesting files, exchanging peer neighbourhood information with others, etc. As a result, the possibility of any individual node becoming "suspicious" and blocking communication or alerting others to malicious activity is extremely low. While some networks have evolved to include a global blacklisting mechanism, any such mechanism is vulnerable to the utilisation of a distributed proxy or routing tool, such as Tor [90].

5.9 Results Processing

The processing of P2P network related results and evidence is largely similar for a variety of potential investigation types. The UP2PNIF system consists of a number of common result processing capabilities:

1. Geolocation – Determining the geographic location of the nodes identified during an investigation is of the utmost importance in deciding the subsequent course of action (if any) regarding attempting to pursue and prosecute the individual(s) involved. Due to the

cross-jurisdiction legal challenges involved, it may be infeasible to advance investigations with specific local authorities. The geolocation of the peers is conducted at an early stage in the investigation can aid in focusing resources towards the desired target of the investigation.

2. Internet Service Provider (ISP) Identification – Each identified IP address is automatically related to its corresponding ISP. This information can be useful to law enforcement in demanding the identification of the users behind any IP address of interest identified during the investigation.

3. Enumeration – The enumeration of the nodes involved in a given network is a common result parameter for most investigations. In combination with the geolocation and ISP level lookup of discovered IP addresses, the largest offending countries, regions and ISPs can be easily computed.

4. Results Correlation – This capability can aid in the correlation of results between different investigations or different snapshots in time. For example, in a copyright infringement investigation, determining that a given peer is appearing across multiple infringements might prioritise that specific case.

5. Evidence Verification – Capturing and identifying evidence is an important aspect of any forensic investigation. This capability aids the investigator in the recording and organisation of the gathered evidence.

6. Churn Rate Computation – In order to get an understanding of the number of peers involved in a particular network, any single snapshot is insufficient to determine how many peers have gone through the network in a given timeframe. The churn rate can help calculate/predict the number of peers connecting to each network. For example, if the entire network was enumerated once every ten minutes, the percentage difference between individual snapshots can help to predict the churn rate in the future.

5.10 Advantages over Existing Tools

Existing tools are all entirely bespoke applications focused on the investigation of a single P2P network. As a result, any potential elimination of the significant, duplicated development efforts is not possible resulting in the significant redundancy of efforts. At the time of writing this thesis, there are no other universal P2P network investigation tools. With sufficient propagation of the use of the framework outlined in this thesis, it is envisioned that much of the redundancy and effort required to build and maintain a bespoke network investigation tool will be eliminated. The processing of gathered evidence from the operation of the tool will also be reproducible irrespective of the network being investigated. The deployment of a universal P2P network investigation framework will also enable interesting cross-network results to be achieved, e.g., how many compromised machines participating in botnet A are also compromised by botnet B?

5.11 Potential Limitations

As with any P2P network investigation, there are a number of limitations when it comes to correlating results over any significant period of time. A number of potential limitations and solutions are outlined below:

1. DHCP – Due to a typical lease from an Internet service provider lasting in the order of 1-7 days, dynamic reallocation of the same IP address may result in two or more peers participating in the network appearing as a single peer.

2. Proxy servers – Similar to the issue caused by DHCP, any nodes that access the Internet through the same transparent or anonymous proxy server will also appear as a single node to the outside world.

3. Identification of peers using anonymous Internet services – This facilitates the identification of services such as Tor (The Onion Router)

and I2P (Invisible Internet Project). By comparing the IP addresses discovered during the investigation with a list of known Internet traffic proxy or pass-through services, the quality of the results collected can be greatly improved. One such proxy services identification list is maintained by MaxMind Inc. [158].

4. Network Address Translation – Numerous machines behind a shared router may appear to the outside world as a single machine, as they share a single external IP address.

5. Encrypted Communication – Should the network employ encrypted communication, the only method available for investigation is to attempt to reverse engineer the client software. The decryption/encryption methods and any associated keys for any incoming/outgoing communications must be stored within the P2P client.

6. Difficulty in Take Down – In order to take down a P2P network, it is often a matter of discovering a vulnerability in its design. Traditionally this has meant attempting to take down its centralised server [145]. However, with the popularity of employing a fully decentralised network design, the ability to take down such a network has been made considerably more difficult. Should the client be reverse engineered, it is possible that the network could be disturbed or have even "imploded", e.g., through the issuing of an uninstall command to each infected node in a botnet.

7. Network Transfer Speed – The time taken to transfer the gathered evidence over the Internet will take longer than the time required if the investigator had physical access to specialised forensic hardware in a forensic laboratory. Where UP2PNIF can improve on this time required for traditional hard drive image acquisition is when the time wasted by the investigation in travelling, transportation and storage of the suspect computer is taken into consideration. While high-speed broadband Internet access is becoming more and more common place on both residential and commercial levels, it would be unrealistic to assume

that every suspected computer would have an Internet connection with a favourable upload speed, i.e., many asymmetric broadband connections are significantly weighted towards download speeds. This limitation could again be overcome through the use of a mobile Internet connection.

The limitations outlined above with respect to accurately enumerating unique peers can be overcome on P2P networks that employ a unique ID number, such as the Torpig botnet [144], the BitTorrent DHT [6], etc. Other heuristic metadata, such as client version information, detected data speeds/latency and the list of available files (in the case of file-sharing networks) can each contribute to unique identification. UP2PNIF is designed to partake in the network as any regular client. While this means that the network must be fully understood or reverse engineered, any encrypted communication methods would also be employed by the resultant investigative tool. By being a part of the network and appearing as any other node, any issues traditionally involved in attempting to decrypt captured network packets are rendered irrelevant.

5.12 The Case for Collaboration

In the ever-evolving space of botnet technology, reducing the time lag between discovering a newly developed or updated botnet and gaining the ability to mitigate against it is paramount. Often, numerous separate investigative bodies, such as law enforcement agencies and security firms, duplicate their efforts in creating bespoke tools to combat particular threats. The framework outlined in this thesis is capable of fast tracking the investigative process through collaboration between key stakeholders.

Figure 5.5: Steps Involved in a Typical P2P Botnet Investigation

5.12.1 Advantages

Implementing a universal collaborative framework quickly and easily facilitates different investigative bodies conducting similar investigations worldwide. Some of the main advantages of such a collaborative system are outlined below:

1. Compatibility – Having a centralised framework facilitates the integration with numerous other network forensic tools as required. Expandability is also ensured through the modularisation of the components and specification of the individual networks.

2. Cost – The cost savings for participating in such a collaborative framework are significant. Eliminating the redundancy due to the duplicated development of bespoke tools will greatly reduce the resources required.

3. Automated Identification – With a sufficiently large database of known networks established, automated identification of known networks from the capturing network traffic should be possible. This will greatly aid the forensic investigator in expediting the investigative process.

4. Cross-border cooperation – Different network focuses across different countries should be advantageous to the system overall. For example, if a predominantly European targeted botnet evolved to target machines in the United States, the US led investigation should be greatly expedited through reusing the work conducted by European agencies. Differing bodies developing the signatures required for their own national operational requirements will result in a more complete robust database

of signatures.

5. Speed – The regular P2P botnet investigation process is shown in Figure 5.5. Using the framework described, the entire investigation process is fast-tracked as far as step 4 for each contributing body (assuming the network is already documented). The final analysis phase can also be expedited through common evidence processing procedures.

5.12.2 Potential Issues

As with any system designed with a requirement for collaboration between differing global bodies and organisations, there are a number of potential issues:

- Key Stakeholder Buy-in – Without a sufficient contribution towards the framework (both the core functionality and the botnet signatures), the system can rapidly become outdated in comparison with the quick progression in botnet technologies.
- Access Maintenance and Control – A framework that potentially has the ability to infiltrate and take control over any reverse engineered botnet is something that would be highly valuable if it got into the wrong hands. Controlling access to such a framework is paramount to its reliability and usefulness to law enforcement and forensic investigators. One solution to this issue would be for the framework to require server-side authentication/update before any investigation can commence.
- Corporate Participation – While having collaboration from private corporations would be advantageous to the system as a whole, from a corporate perspective having exclusivity on the solution to any problem is fundamental to many business models. As a result, it is envisioned that the system will primarily be contributed to and used primarily by law enforcement and research institutions.

5.13 Summary

This chapter introduced the Universal Peer-to-Peer Network Investigation Framework, its modular components and its architecture. As a result of partaking in the target P2P network of an investigation, a novel method for creating and storing metadata alongside the resulting evidence gathered.

As the usage of the framework increases over time, the benefit will become increasingly apparent. Currently, much time is wasted on redundant efforts between different law enforcement agencies and digital investigation specialists globally in the design and development of similar investigative software. It is envisioned that the use of a centralised, accessible system, such as UP2PNIF, will greatly improve the productivity of future P2P investigations and aid in the ever-evolving fight against malicious P2P activity on the Internet.

CHAPTER

SIX

FORENSIC INVESTIGATION OF BITTORRENT

This chapter outlines a prototype developed to test the design of the system outlined in Chapter 5. In the upcoming sections, three BitTorrent investigations of varying focus and duration are presented. The first two investigations, described in Sections 6.5 and 6.6, track the unauthorised activity across two separate content types. The third investigation, discussed in Section 6.7, outlines a significantly larger investigation, monitoring and tracking the top 100 most popular pieces of content shared on the BitTorrent network over a week long period.

6.1 Development

An investigation framework and BitTorrent crawling application was developed that is capable of executing various types of P2P network based investigations, as outlined in greater detail in Chapter 5. This framework was built to emulate and amplify regular client usage. This emulation has the advantage of negating some of the legal concerns raised earlier in Section 2.9. A customised Ubuntu Server Linux image was created with the crawling application, configured for high I/O operations and with the security credentials required for processing the data and storing it in the

results database.

The results database was built using the NoSQL database management system, MongoDB, due to its performance benifits over its relational counterparts [159]. This database facilitated powerful cross-crawl and cross-investigation comparisons, as discussed in Sections 6.5, 6.6 and 6.7.

6.2 Investigation Methodology

1. Assess – Discover as much as possible about the network to be investigated. This can be achieved through the sniffing of an active node in the network, e.g., deployed as part of a honeypot. Although the amount known at the end of this step will be network dependent, an ideal scenario would result in the following characteristics being known:
    - Topology of the network
    - Protocols employed
    - Method of encryption (if any)
    - Common commands and queries available
    - Peer discovery methods
2. Develop – This includes the development of a P2P client emulator for the specific network being investigated. This client must have the ability to bootstrap onto the network, while appearing as a regular node to every other active node on the network. Regular client operations must also be available, including peer discovery and peer communication.
3. Monitor – The monitoring of the network over a given time period will record the number of active peers, the network churn rate, the network reliability and uptime, the resource size available and the commands and queries received.
4. Analyse – The data gathered from the monitoring step must be analysed to determine the following:

Table 6.1: BitTorrent Network Profile

Feature	Value
fileSharing	true
botnet	false
centralised	true
decentralised	true
configFile	true
encrypted	false
httpPeerDisc	true
httpFreq	0
peerExchange	true
peerExchangeFreq	600
dht	true
dhtFreq	0

- The types of cybercrimes the network is being used for.
- Any inherent weaknesses in the network such as a single point of failure or the possibility of issuing commands to the network.
- Encryption methods employed
- If it is possible to identify the source of the commands. This will include the identification of "super peers", i.e., nodes responsible for distributing the commands across the network and the possible identification of the controlling machine or bot master.

5. Results Visualisation – The visualisation of the results from any investigation will enable law enforcement to quickly identify the key focus targets of their investigation. Any reporting on the investigation should clearly explain the complete methodology of the investigation as these results should be verifiable if required for court proceedings.

To start, the basic profile for BitTorrent was recorded in the network database, as shown in Table 6.1. Alongside this identifying information, the format of the communication methods are also stored, as displayed in Table

5.1. A hierarchical object oriented approach to network investigation was implemented for each of the BitTorrent peer communication methods. In testing the crawler, it was found that employing a hierarchical approach aided in the distribution of scheduled jobs and in the efficient processing of the gathered data.

6.3 Experimentation Setup

In order to conduct a crawl of a P2P network using the system developed, a number of different options were considered. While running the investigation locally on a workstation or server in a forensic laboratory is possible, the multi-threaded application is limited by the processing power and physical network performance. In order to scale the system up to be capable of handling any investigation size, running the application from the cloud become a necessity. The specifications of the infrastructure used are outlined below:

6.3.1 Infrastructure

The experiments outlined below were executed on Amazon Web Services Elastic Compute Cloud (EC2). An "Extra Large" High-CPU instance was chosen for the investigation due to the high I/O required of any P2P network focused investigation. The extra large high-CPU instance consists of the following specifications:

- 64-bit Platform
- 7GB Memory
- 20 EC2 Compute Units (One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor)
- 1690GB Local Instance Storage

Running the application on such a system resulted in optimum performance. Testing on more powerful EC2 instances did not result in any performance benefits over the extra large instance when investigating a single piece of content. Investigating multiple pieces of content simultaneously is better served with multiple EC2 instances, as opposed to more powerful instance types. This is due to the network bandwidth allocation limits per instance.

6.4 Assumptions and Accuracy

The country, city and ISP level geolocation database used is maintained by MaxMind Inc. and is the most accurate database currently available [160]. In May 2013, MaxMind stated that their geolocation databases were 99.8% accurate at a country level, 100% accurate at the ISP level and 81% accurate at a US city level within a 40 kilometre radius (other countries ranging from 50% to 96%, e.g., in the Philippines and Cote D'Ivoire respectively) [158].

There are a number of assumptions that have to be made for the purposes of the geolocation of the IP addresses and for the enumeration of the total number of P2P users involved in the networks investigated:

1. For any investigation taking place over a significant time period (12+ hours), each IP address found to be participating in the network investigated is assumed to only ever be allocated to one end user. Due to a typical DHCP lease from an Internet service provider lasting somewhere in the order of 2-7 days, dynamic IP address allocation may result in the reallocation of the same IP address to two or more end users during the investigation. Should this have occurred during the investigation, it is ignored for the interpretation of the results outlined below. It is deemed technically infeasible to identify precisely when this may occur on a global level within the scope of this work.

2. No anonymous proxy servers or Internet anonymity services, e.g., I2P [91], Tor [90], etc., are used by the IP addresses discovered. While

anonymous proxy detection methods exist, they are generally focused on detecting their use from a server-side perspective, as discussed in [161] and [162].

3. It is practically infeasible for users on dial-up Internet connections to download the very large files that typically justify distribution via the BitTorrent protocol. For the purposes of the analysis presented in Section 6.7, it was assumed that a negligible amount of dial-up users participate in the BitTorrent swarms. The average content size for the swarms investigated as part of the investigation outlined in Section 6.7 was 1.62GB, which would take a typical 56kbps dial-up user over 69.5 hours to download, assuming no bandwidth competition from other Internet traffic and that the connection was achieving the maximum theoretical dial-up connection speed.

4. Mobile cellular/WiMax based connections, without cooperating with the provider, are only capable of being resolved to the exchange or hub used by the provider. As a result, the city level accuracy for these connection types is much larger than for fixed line broadband services.

6.5 Album Piracy

For this investigation, the most popular music album was selected (as identified on The Pirate Bay's most popular music listing), "Random Access Memories" by Daft Punk. When the investigation commenced, the swarm was just over two days in existence and the torrent's creation date was before the official release date of the album, i.e., it was an early leaked copy of the album. The investigation monitored the activity of the swarm for a period of 24 hours.

```
<torrent>
    <saveAs>Daft Punk - Random Access Memories</saveAs>
    <name>Daft Punk - Giorgio by Moroder.mp3,
          Daft Punk Feat Paul Williams - Touch.mp3,
          Daft Punk - Contact.mp3,
          Daft Punk Feat Pharrell Williams - Get Lucky.mp3,
          Daft Punk Feat Pharrell Williams - Lose Yourself to Dance.mp3,
          Daft Punk - Motherboard.mp3,
          Daft Punk Feat Julian Casablancas - Instant Crush.mp3,
          Daft Punk - The Game of Love.mp3, Daft Punk - Beyond.mp3,
          Daft Punk Feat Todd Edwards - Fragments of Time.mp3,
          Daft Punk - Give Life Back to Music.mp3,
          Daft Punk Feat Panda Bear - Doin' It Right.mp3,
          Daft Punk - Within.mp3
    </name>
    <pieceHashValuesHex>
        6B454A3907EDE5AED0E1FDCCCA373B82334D3620,
        [...]
    </pieceHashValuesHex>
    <totalLength>179584713</totalLength>
    <infoHashHex>DFED837082F1BAB3E77C0945701CC0768C87B9E0</infoHashHex>
    <announceURL>udp://tracker.publicbt.com:80/announce</announceURL>
    <announceList>udp://121.14.98.151:9090/announce.it,
                  http://tracker.publicbt.com:80/announce
                  udp://newhorizons.twentyforty.me:2710/announce,
                  http://tracker.thepiratebay.org:80/announce
    </announceList>
    <comment>Torrent downloaded from torrent cache at http://torcache.net/</comment>
    <createdBy>uTorrent/3300</createdBy>
    <creationDate>2013-05-16 17:20:03</creationDate>
    <encoding>UTF-8</encoding>
    <pieceLength>262144</pieceLength>
</torrent>
```

Figure 6.1: Daft Punk Torrent Information (Truncated Piece Hash Values)

Hour	Active IPs	Churn Rate	Hour	Active IPs	Churn Rate
0	5,538	1.41	12	6,195	1.58
1	5,229	1.33	13	6,073	1.55
2	5,079	1.30	14	5,976	1.52
3	4,981	1.27	15	5,959	1.52
4	4,944	1.26	16	6,078	1.55
5	4,749	1.21	17	6,179	1.58
6	4,619	1.18	18	6,223	1.59
7	4,633	1.18	19	6,323	1.61
8	4,879	1.24	20	7,005	1.79
9	5,345	1.36	21	6,938	1.77
10	5,823	1.49	22	6,557	1.67
11	6,096	1.55	23	6,188	1.58

Table 6.2: Daft Punk: Active Peers Discovered Per Hour (GMT)

6.5.1 Content Overview

The information contained within the ".torrent" file for the album is displayed in Figure 6.1. The content's file size was 171MB (or 179,584,713 bytes), which is split into 686 separate 256Kb (262144 byte) pieces. The torrent was originally registered with the four specified trackers, although the torrent was also aided by numerous additional trackers. Additional trackers can get added to the torrent whenever it gets uploaded onto BitTorrent indexing sites.

6.5.2 Churn Rate

The minimum number of active peers detected in the swarm over the 24 hour window was 2,244 and the maximum number was 3,921. The swarm expansion/contraction over the timeline of the investigation is shown in Figure A.1. If this content was investigated once per day, the number of peers actively sharing the content may appear to be 3,921 at most. In order to calculate the churn rate of the swarm, 2,098 separate consecutive crawls were conducted, taking an average of 41.2 seconds to complete. Over the 24 hour window of the investigation, 45,588 distinct IP addresses were detected. Based on the maximum swarm size, the churn rate of this swarm can be calculated to be 11.6 times per day.

Position	Country	Number	Position	Country	Number
1	United States	10419	11	Poland	721
2	United Kingdom	5278	12	Portugal	695
3	Australia	4325	13	Spain	685
4	France	3263	14	India	608
5	Canada	2909	15	Philippines	582
6	Netherlands	1970	16	Ireland	576
7	Italy	1176	17	Argentina	401
8	Mexico	1001	18	Sweden	384
9	Brazil	977	19	Denmark	378
10	Belgium	919	20	Turkey	375

Table 6.3: Daft Punk: Top 20 Countries

After the initial crawl to set a baseline, the number of new peers discovered between crawls ranged from 0 to 117 new peers. These numbers show the importance of frequent crawling to get a true picture of the number of peers involved in the downloading of any particular piece of content. Figure A.2 shows the number of new peers found per crawl.

6.5.3 Peer Connection Time

The average number of crawls that each peer appeared in was 117.5 crawls. Taking the average crawl time of 41.2 seconds, the average peer connection time was just over 1 hour and 20 minutes over the 24 hour window of the investigation. This cannot be considered the average download time due to the default behaviour of most BitTorrent clients. Most clients seed the content while both actively downloading other pieces, and when the content is complete. Figures A.3 and A.4 show the number of discovered IP addresses and the corresponding number of crawls each IP address appeared in. Over 87% of peers were connected for less than 200 crawls (or 137 minutes). 18.8% of peers were connected to the swarm for less than 6.5 minutes over the 24 hour window.

ISP	Country	Count of IP Addresses
Comcast Cable	United States	2505
Road Runner	United States	1716
Telstra Internet	Australia	1389
Virgin Media	United Kingdom	1120
SBC Internet Services	United States	1115
British Telecommunications	United Kingdom	1076
Free SAS	France	1050
Verizon Internet Services	United States	1018
France Telecom	France	979
Sky Broadband	United Kingdom	745
Optus Internet - Retail	Australia	740
Shaw Communications	Canada	693
Telewest Broadband	United Kingdom	655
Cox Communications	United States	630
Neuf Cegetel	France	617
TPG Internet Pty Ltd.	Australia	574
Rogers Cable	Canada	523
Uninet S.A. de C.V.	Mexico	463
Opal Telecom	United Kingdom	441
Telecom Italia	Italy	439

Table 6.4: Daft Punk: Top 20 IP Addresses Detected per ISP

6.5.4 Geolocation

As shown in Table 6.3, the United States was the country with the highest number of active IP addresses in the swarm, followed by the United Kingdom and Australia. Figure A.5 shows the activity of the top ten countries over the 24 hour period. Given the timezone differences, peak activity can generally be seen rising throughout the day to its highest point in the evenings. Most countries also show a peak in activity in the mid-afternoon which may indicate increased Internet usages by minors outside of regular school hours. The global city level, European city level and globel country level geolocated results can be seen in Figures A.6, A.7 and A.8 respectively. The top 20 ISPs detected are outlined in Table 6.4.

6.6 TV Show Piracy

To investigate the level of piracy in the TV show category, the most popular weekly TV show was selected according to The Pirate Bay, "Game of Thrones" (GOT). The two most recent episodes were selected from the third season, episodes seven and eight (generally stylised across BitTorrent indexing websites as "S03E07" and "S03E08"). Both of these episodes were investigated during the same 24-hour window to identify the differences between a week-old torrent and a new torrent (less than 5 hours old when the investigation commenced).

6.6.1 Content Overview

The file sizes of the MP4 video files shared in these swarms were 443Mb and 358Mb for episode seven and eight respectively. These files were split into numerous chunks of 512kb and 256kb, as decided by the torrent creator. It is worthy noting that the TV show was scheduled to play on the United States television network HBO at 9pm EST (2am GMT). Quickly after the time of the show finishing (approximately 10pm EST/3am GMT) the posters of the content removed the ads from the recording and encoded it into a suitable compressed web format. According to the torrent files, the episodes were available online at 3:15am and 3:03am respectively, as shown in Figure 6.2.

6.6.2 Enumeration

The swarm size of the week-old torrent (S03E07) varied between 16,306 and 28,211 IP addresses with 204,451 IP addresses identified in total. In comparison, the swarm size of the 5 hour old torrent (S03E08) varied between 52,793 and 65,080 with 759,521 IP addresses identified over the same 24 hour window, as can be seen in A.9. The drop off in the swarm size of S03E07 (outlined in blue in the Figure) can be attributed to a "competing" torrent of the same episode from another release group. In total, there were 882,105

```
<torrent>
    <saveAs>Game.of.Thrones.S03E07.HDTV.x264-2HD.mp4</saveAs>
    <name>Game.of.Thrones.S03E07.HDTV.x264-2HD.mp4</name>
    <pieceHashValuesHex>
        FBCB4FB98C9601ACDCD33FF0B7886CDDA7185737,
        [ ... ]
    </pieceHashValuesHex>
    <infoHashHex>A7CC7A2A5A179E7431EC7CDDB72C2BC45D764CDD</infoHashHex>
    <announceURL>udp://tracker.publicbt.com:80/announce</announceURL>
    <announceList>http://tracker.openbittorrent.com:80/announce,
                  udp://fr33domtracker.h33t.com:3310/announce,
                  http://tracker.istole.it:80/announce,
                  udp://tracker.ccc.de:80/announce,
                  udp://tracker.beeimg.com:6969/announce,
                  http://fr33dom.h33t.com:3310/announce,
                  udp://exodus.desync.com:6969/announce</announceList>
    <comment>
        Torrent downloaded from torrent cache at http://torcache.net/
    </comment>
    <createdBy></createdBy>
    <creationDate>2013-05-13 03:15:08</creationDate>
    <encoding></encoding>
    <pieceLength>524288</pieceLength>
</torrent>


<torrent>
    <saveAs>Game of Thrones S03E08 HDTV x264-EVOLVE[ettv]</saveAs>
    <name>Game.of.Thrones.S03E08.HDTV.x264-EVOLVE.mp4</name>
    <pieceHashValuesHex>
        F9C4036B915B11D769EE1A247CD4C907106BCB7B,
        [ ... ]
    </pieceHashValuesHex>
    <infoHashHex>CBC721C4D97950D5D7FEC51ED6235F38B8E7CA43</infoHashHex>
    <announceURL>udp://tracker.publicbt.com:80/announce</announceURL>
    <announceList>udp://leopard.raws.ws:6969/announce,
                  udp://tracker.thehunter.com:6969/announce,
                  udp://bt.firebit.co.uk:6969/announce,
                  udp://retracker.nn.ertelecom.ru:80/announce,
                  udp://best-torrents.net:6969/announce,
                  udp://bt.careland.com.cn:6969/announce,
                  udp://fr33dom.h33t.com:3310/announce</announceList>
    <comment>
        Torrent downloaded from torrent cache at http://torcache.net/
    </comment>
    <createdBy>ruTorrent (PHP Class - Adrien Gibrat)</createdBy>
    <creationDate>2013-05-20 03:03:50</creationDate>
    <encoding></encoding>
    <pieceLength>262144</pieceLength>
</torrent>
```

Figure 6.2: Game of Thrones S03E07/S03E08 Torrent Information (Truncated Piece Hash Values)

unique IP addresses detected churning through these two swarms.

6.6.3 Churn Rate

The churn of the swarms investigated was over 8.1 times over the 24 hour window and the average churn rate for the swarms was over 1.8 times per hour. These churn rates show that due to the larger content file size each peer generally needs to remain connected for a longer period of time (in comparison to the album investigation discussed above). Many peers did not complete the download in one continuous connection; disconnecting and rejoining the swarm at a later time.

6.6.4 Geolocation

Due to the time lag between the episodes airing in the US and the episodes airing on other networks worldwide (generally in the order of days or weeks, if at all), it might be expected that other predominantly English speaking countries would appear highly in the list of top countries. While they do appear highly in the list, the most popular country is the United States claiming over 10.8% of the overall activity, as shown in Figure A.10. The glboal city level distribution for S03E07 and S03E08 can be seen in Figures A.12 and A.15. European city level distribution can be seen in Figures A.11 and A.14, for each episode respectively. The global country level heatmaps for each episode can be seen in Figures A.13 and A.16.

6.6.5 Detection Overlap Between Weekly Episodes

The overlap between the two investigations was 79,867 IP addresses. This figure represents that over 10.5% of those involved in the most recent episode's (S03E08) swarm were also involved in the previous episode's swarm. The continuous monitoring of the weekly overlap between TV show swarms calculated in this manner could be useful for ISPs in predicting their network

bandwidth allocations across their infrastructure at specific locations. Figure A.17 shows the overlap between the IP addresses discovered in mainland Europe.

6.7 BitTorrent Landscape Investigation

In order to get a broader sample of the activity across a range of torrents, a larger investigation was conducted. This investigation shows the activity over a week-long investigation of the top 100 most popular torrents. The steps involved in the execution of this investigation were:

1. Connect to The Pirate Bay and download the ".torrent" metadata files for the top 100 torrents.
2. Connect to each swarm sequentially and identify each of the IP addresses currently active in the swarm until no new IPs are found.
3. Once the active IPs are found for the entire 100 torrent swarms, the process is repeated for the next 24 hours.
4. After 24 hours, the process was restarted again at step 1.

The investigation was conducted using a single dedicated server, which sequentially monitored each torrent swarm until all the IPs in the swarm were found. Over the course of the seven-day investigation, a total of 163 different torrents were investigated. None of the content appearing in these torrents was found to be distributed legally; each torrent swarm was distributing copyrighted material without the need for any documented authorisation.

6.7.1 Content Analysis

From the analysis of the daily top 100 swarms, video content was found to be the most popular category of content being distributed over BitTorrent. Movie and television content amounted to over 94.5% of the total, as shown in Figure 6.3. Music, games and software accounted for 1.8%, 2.5% and 1.2% respectively. One highly probable explanation for the popularity of television content is due to the lag between popular US produced television shows airing in the US and the rest of the world.

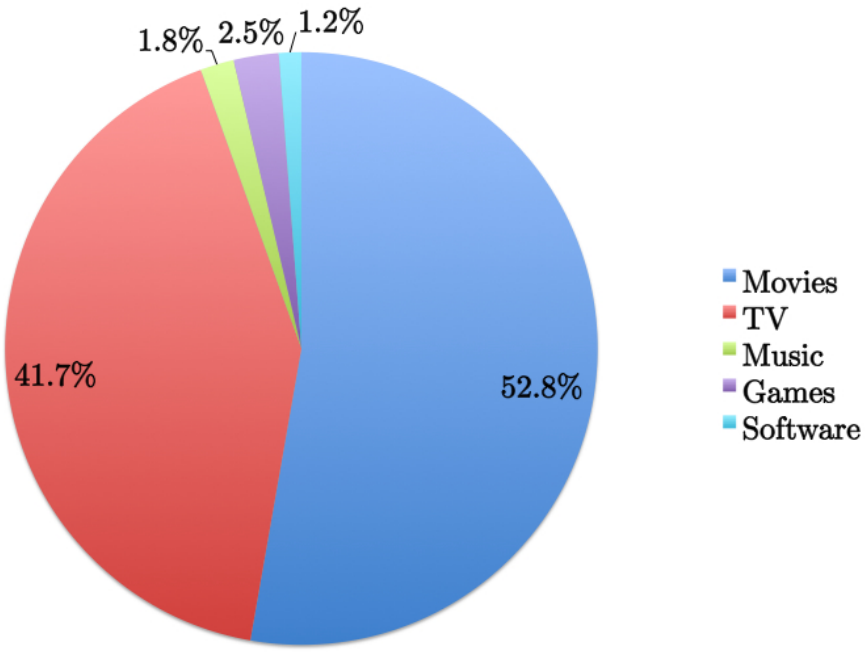

Figure 6.3: Category distribution of the top 100 torrents on The Pirate Bay

6.7.2 Geolocation and Visualisation

For each IP address detected during the investigation, the geolocation is obtained using MaxMind's GeoIP database [158]. This database helped produce location information such as city, region, state, country, latitude, longitude, ISP, etc. This information is then gathered and plotted as a heatmap to display the distribution of the peers involved in copyright infringement on a world map, as shown in Figure A.18. The most popular content tends to be content produced for the English speaking population, which is reflected in the heatmap, i.e., countries with a high proportion of English speaking population are highlighted in the results.

6.7.3 Results

Over the week long investigation, the total number of unique IP addresses discovered was 8,489,287. On average, each IP address detected was active in 1.75 swarms, or, almost three out of every four IP addresses were active

in at least two of the top 100 swarms during the week. The largest swarm detected peaked at 93,963 active peers and the smallest swarm shrunk to just 1,102 peers. The time taken to capture a snapshot of 100 swarms investigated varies due to the increase and decrease in the overall size of the swarms. The average time to collect all the peers' information for each swarm is 3.4 seconds.

The United States was the most popular country detected with over 1.1 million unique IP addresses, which accounted for 13.15% of all the IP addresses found. While accounting for the largest portion of the results obtained in this investigation, this relatively low percentage suggests that BitTorrent has a much more globally dispersed user base in comparison to other large P2P networks. For example, a 10-day investigation conducted on the Gnutella network in 2009, found that "56.19% of all [worldwide] respondents to queries for content that is copyright protected came from the United States" [78]. When the IP addresses detected during this investigation were geolocated and graphed onto a map, the population centres can be easily identified, as shown in Figure A.21. The state of California accounted for 13.7% of the US IPs found, with the states of Florida and New York accounting for 7.2% and 6.8% respectively. Similar maps are shown for Ireland and United Kingdom in Figures A.19 and A.20.

The objective of this investigation was to identify the scale of the unauthorised distribution of copyrighted material worldwide using the BitTorrent protocol. 2.43% of the broadband subscriber base was detected over the course of one week in the 163 torrents monitored. This number is far greater than the number of subscribers that could possibly be prosecuted for their actions. The number of end users involved in illegal downloading is undoubtedly much higher than this, due to the relatively small scale of this investigation. Some network factors will also have a negative effect on the results achieved, such as two or more end-users appearing as a single Internet IP address through Internet connection sharing, proxy services, etc. [163].

6.8 Results Summary

As it can be seen from the investigations conducted above, P2P piracy is indeed a significant problem for the content producing industry. The results obtained show the proliferation of P2P piracy that exists in modern society. The framework prototyped to conduct the above investigation proves the viability of such a system outlined in Chapter 5.

CHAPTER

SEVEN

CONCLUSION AND DISCUSSION

Currently a need exists with law enforcement for a universal P2P investigative tool capable of identifying the crimes and the criminals behind some of the world's largest P2P networks. This thesis proposes a solution to this problem incorporating the individual investigative techniques required and a methodology for completing them. A proof of concept tool was developed and tested on BitTorrent, the world's largest documented P2P network. The future plan for expansion upon this work is to produce an intelligent P2P monitoring tool. Such a collaborative, investigative tool would be of significant benefit to law enforcement in investigating cybercrimes that utilise P2P communications.

In total, over 4TB of evidence was gathered using the prototyped system. This consisted of evidence specific to a number of peers in the order of tens of millions. A precise number is unattainable due to resource contraints for data storage throughout the project. The latest prototype of the system processes the results into a NoSQL database (based on MongoDB) capable of quickly performing cross-swarm and cross-investigation queries.

7.1 Analysis of Outlined Approach

The approach discussed in Chapter 5 outlines a novel modular universal P2P network investigation framework. As of the date of this thesis, no other

collaborative P2P network investigation system exists. The primary benefit of this collaborative approach is that it can be easily expanded upon and updated to create a leading tool in the arsenal of the forensic investigator. Through shared resources and expertise, many wasted man hours could be reallocated to the analysis of the evidence and the prosecution of those responsible for P2P based cybercrimes. It is envisioned that this framework will be made available for collaboration to law enforcement. This should help to eliminate some of the redundancy of efforts by local law enforcement agencies in an attempt to combat P2P based cybercrimes.

7.1.1 Enhancements

Due to the aforementioned commonality in design and implementation of P2P networks, it is envisioned that the proof-of-concept P2P network investigation framework should be expanded to handle any P2P network.

Given the gathered network traffic from any active node of a new P2P network, the system should be enhanced to automatically determine the networks topology, protocols and available commands. This would greatly speed up the first two steps, as shown in Figure 5.5, and enable the monitoring procedures to commence as early as possible in the investigation.

7.2 Further Ideas

While the objectives of the research outlined in this thesis were met, there are some ideas and features which could be added to (or used in conjunction with the existing system) to improve the overall level of functionality. Potential modifications to the current system include implementing automated P2P network traffic pattern recognition, creation of a comprehensive database of P2P network signatures and automated result processing.

The framework developed was designed and prototyped in such a manner as

to easily facilitate the expansion of the tool to deal with any P2P network. It is hoped that in the future, numerous botnet investigation bodies will contribute to the maintenance and development of the framework.

7.2.1 Bespoke Hardware Device

A specific hardware device could be created to piggyback between an infected machine and its Internet connection. When in operation, this device would automatically acquire network evidence from the suspect computer's communication. This device could subsequently perform on-the-fly network identification and processing of the live communications.

7.2.2 P2P Audio/Video Reconstruction

With P2P technology being increasingly utilised for VOIP communication, the reconstruction of captured audio or video content could be crucial to forensic event reconstruction activities. Through the analysis of captured UDP packets, the voice/video call should be capable of being reconstructed. Using pattern analysis, collected evidence could be reconstructed to potentially better quality than the original call, i.e., patching collected packets together in the correct order.

With the popularity of P2P based file-sharing, this reconstruction could also be used the verification of suspected content as being a true copy of the original. In this scenario, a partial sample of the entire content could be used to verify the infringement of copyright.

7.2.3 Usability Test

As outlined as part of the technical requirements of the UP2PNIF system in Section 5.2, the framework should be relatively easy to use for regular law enforcement officers and should require minimal training. In order to measure

this requirement, a usability test should be conducted. This test should invite law enforcement officers and digital forensic investigators to take part. The groups should be randomly divided into two teams, each given the same task of collecting digital evidence from known P2P networks. The two teams would be divided as follows:

1. One team would not be given any instruction on how to use the framework.
2. The second team would be given a short introduction to using the framework, how it operates and the best practices while using the tool.

Should both teams achieve their task in a similar time frame, the ease of use of the tool would be proven. This result would also prove the reduced level required of digital forensic expertise to use the tool. Feedback received from the usability testing could be useful in building upon the current system.

7.2.4 NIST Computer Forensics Tool Testing

Computer Forensics Tool Testing (CFTT) is a standardised set of tests procedures, criteria and hardware compatibility checking performed by NIST to validate computer forensic tools for use by law enforcement. When a sufficient number of P2P networks are added to the system, the tool should be sent for independent, third-party verification.

7.3 Future Vision

7.3.1 P2P in the Cloud

With many everyday services being pushed to the cloud in recent years, one could assume that P2P networks themselves might become redundant in the future. However, alongside the push for cloud based services and storage,

there has also been a significant rise in P2P anonymity services and P2P-aided, cloud driven services, e.g., P2P based streaming services such as Spotify and BitTorrent Live. Controlling a botnet from the cloud could easily facilitate criminals in adding an additional, often temporary, layer of removal from the botnet itself. This potentially could aid the botmaster in avoiding detection completely.

7.3.2 Mobile P2P

The vast majority of botnets existing today are developed to be executed on desktop computers worldwide. However, in the future, it is envisioned that mobile botnets will become commonplace. Smartphones and 3G-enabled tablets are an ideal "next target" device for botnet developers as they are difficult to trace solely based on the data connection. The mobile devices themselves are becoming more powerful with each device having its own always-on Internet connection [164].

7.4 Conclusion

The phenomenon of the ever increasing number of crimes being aided by P2P networks is set to continue into the future due to the level of anonymity provided to cybercriminals. As a result of this inevitable increase in P2P based cybercrimes, digital forensic investigators' workload is set to drastically increase. Any saving of the investigators' time that can be allocated to performing the analysis of captured evidence will help to aid the turn around time for investigations.

This thesis proposed and validated the viability of a forensically sound, P2P evidence acquisition framework. This framework processes the network evidence into an "investigation-ready" state for the forensic laboratory as early into the investigative process as possible. The existing model for P2P network evidence acquisition generally requires a digital investigator to first

develop a bespoke tool capable of deciphering the captured packets from a compromised machine. The use of the UP2PNIF system can significantly improve on this traditional model by fast-tracking the investigation.

APPENDIX

A

GRAPHICAL RESULTS

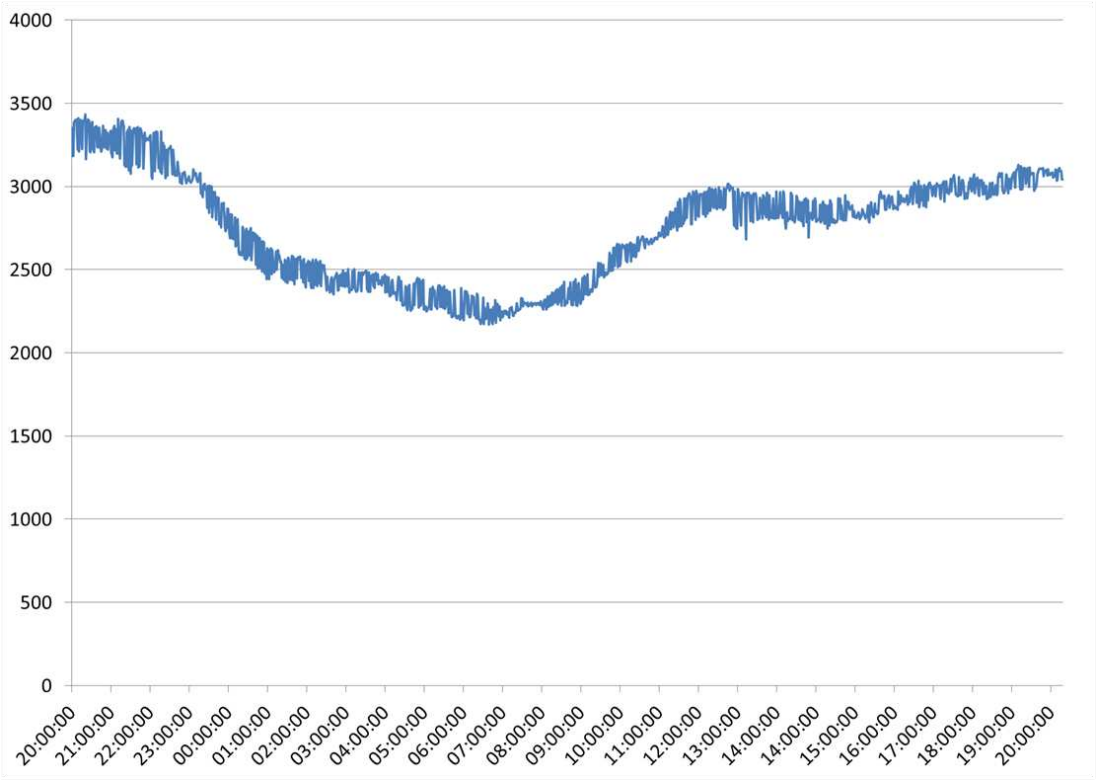

Figure A.1: Daft Punk: Active Swarm Size over 24 Hours

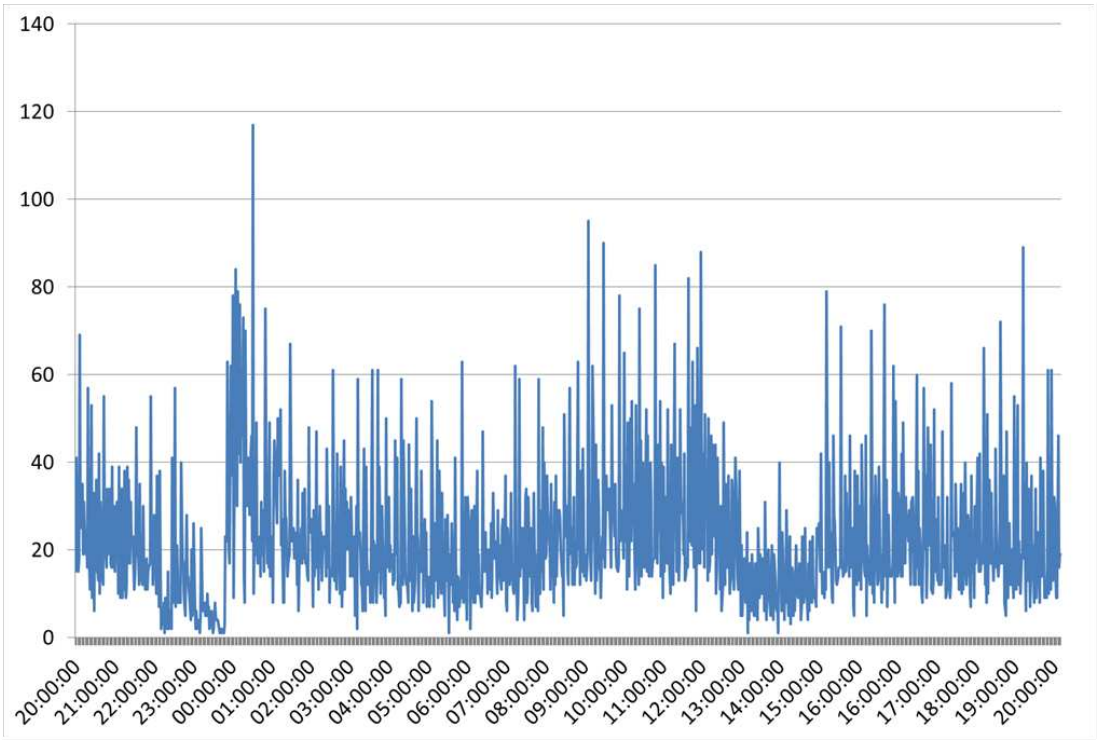

Figure A.2: Daft Punk: Newly Discovered Peers Identified per Crawl (Excluding the Initial Crawl)

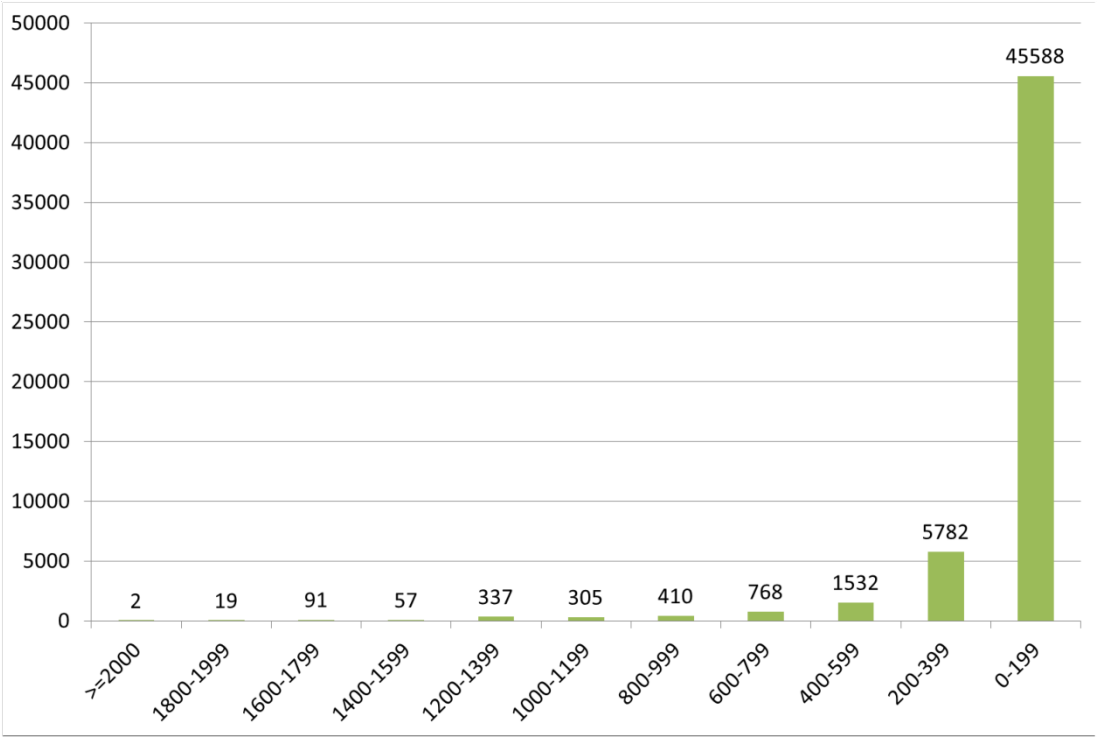

Figure A.3: Daft Punk: Overall Average Peer Crawl Count

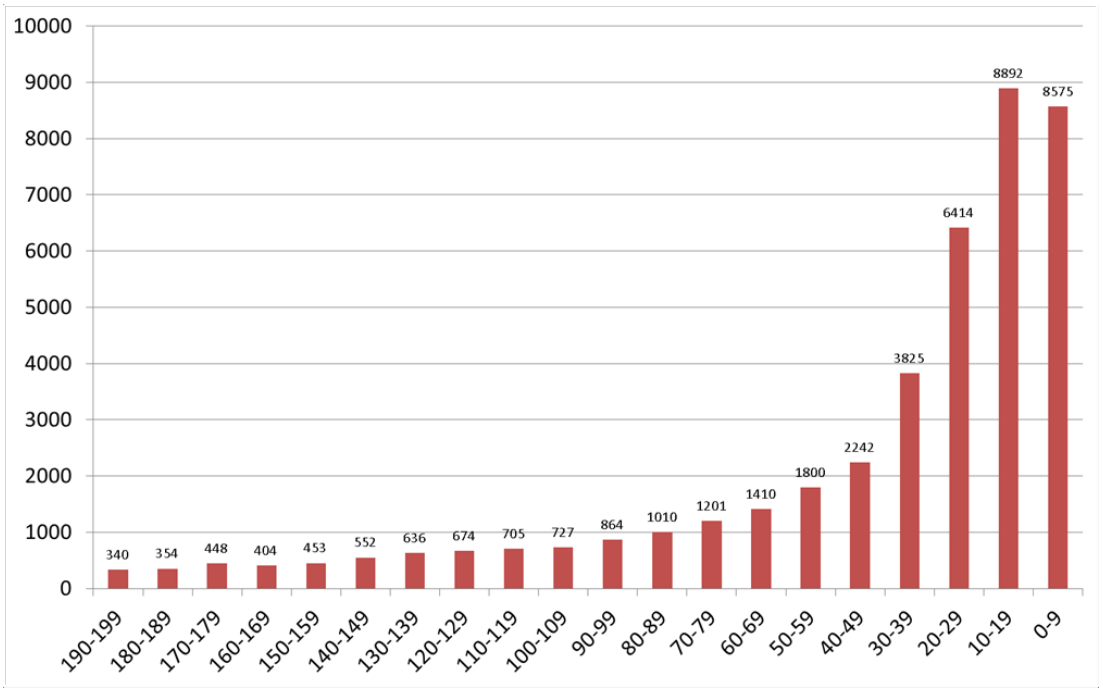

Figure A.4: Daft Punk: Average Peer Connection Time for 0-200 Crawl Count

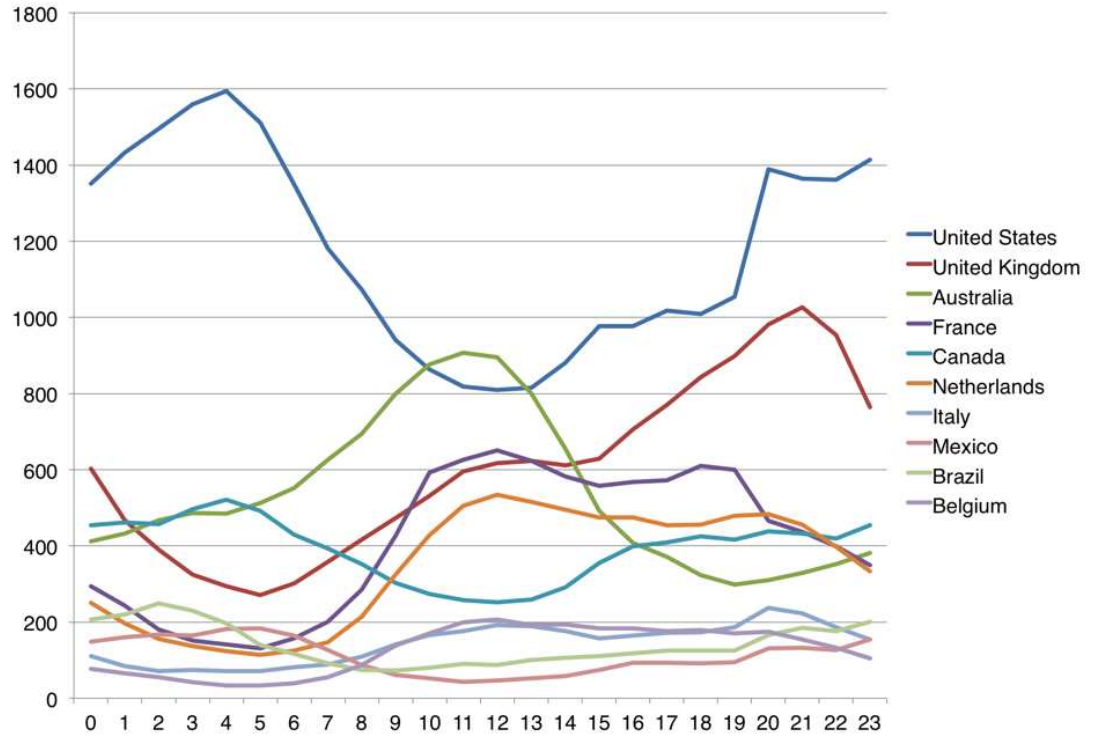

Figure A.5: Daft Punk: Top 10 Countries Hourly Activity (GMT)

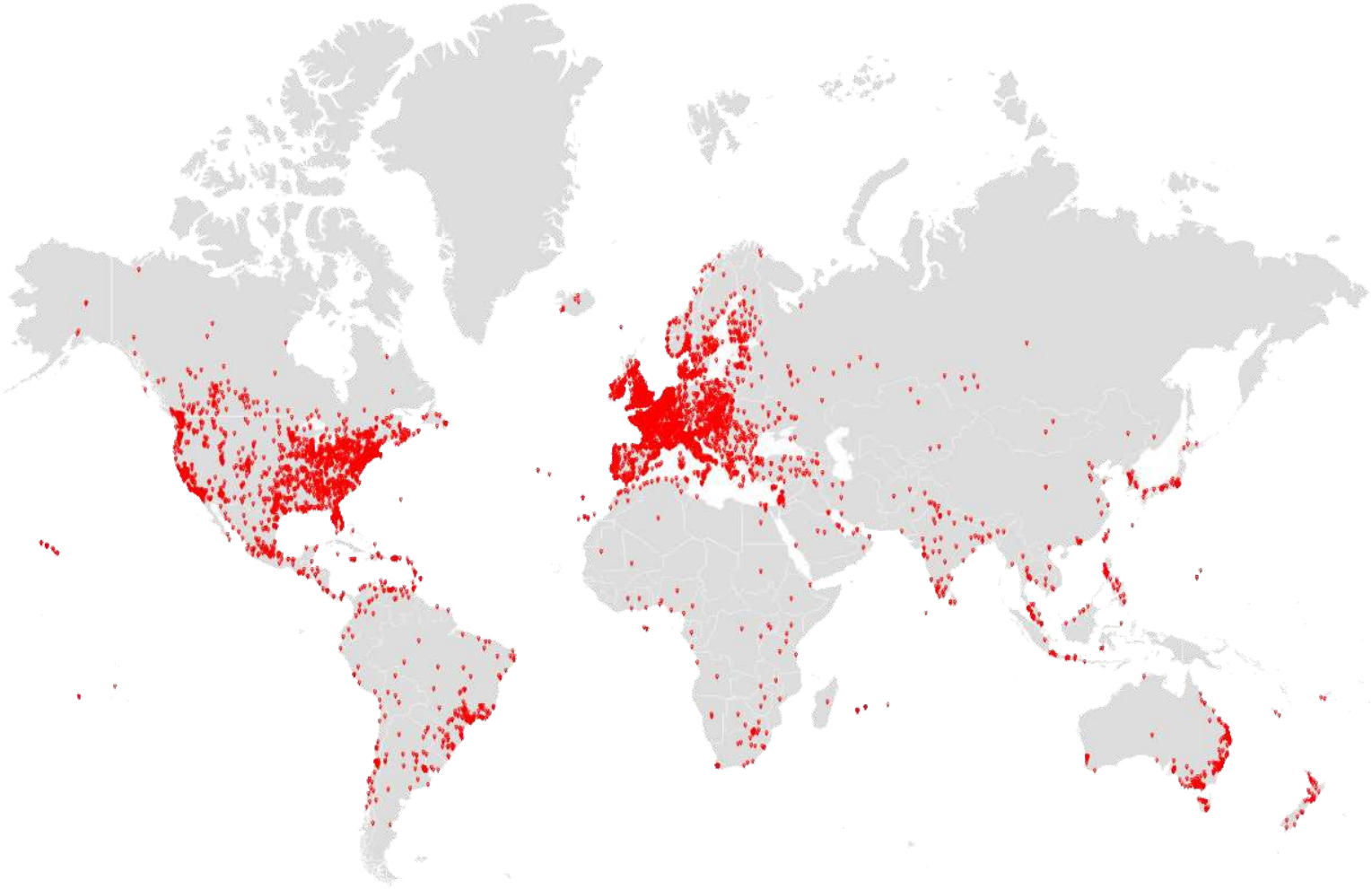

Figure A.6: Daft Punk: Geolocation for Worldwide Cities

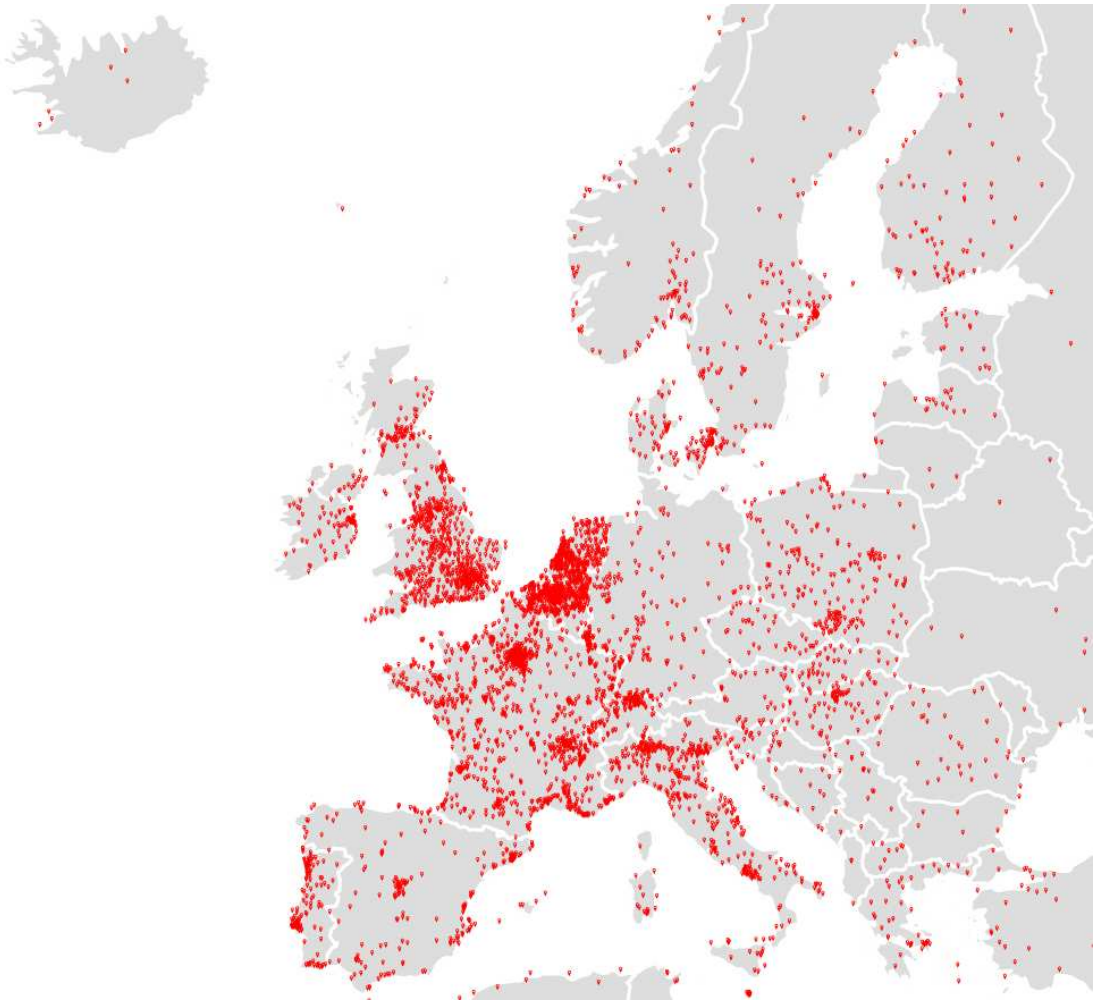

Figure A.7: Daft Punk: Geolocation for Mainland Europe

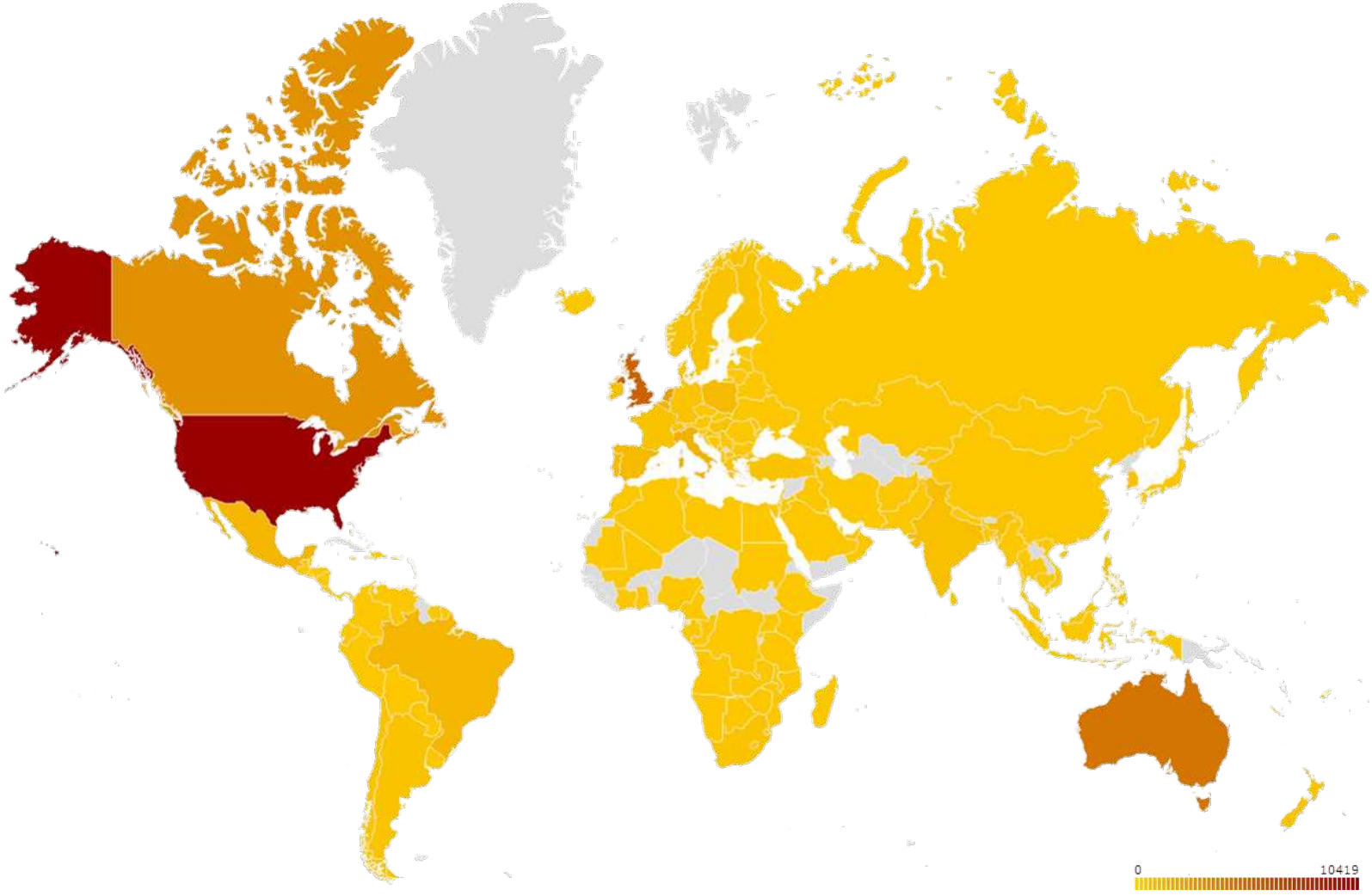

Figure A.8: Daft Punk: Global Heatmap

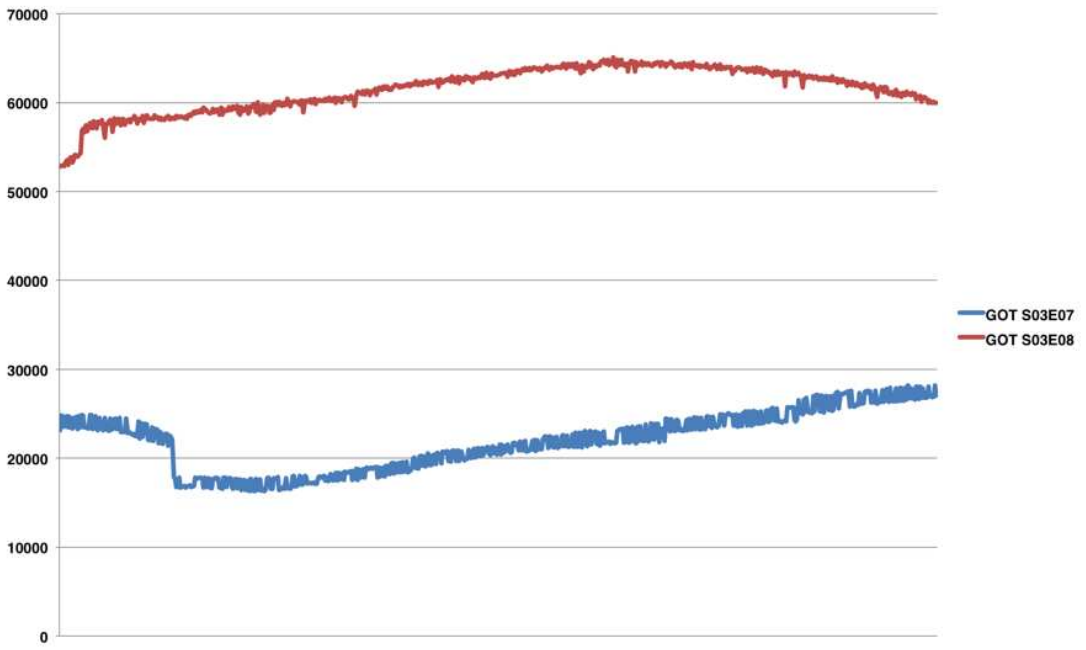

Figure A.9: Game of Thrones S03E07/S03E08: Swarm Sizes over 24 hours

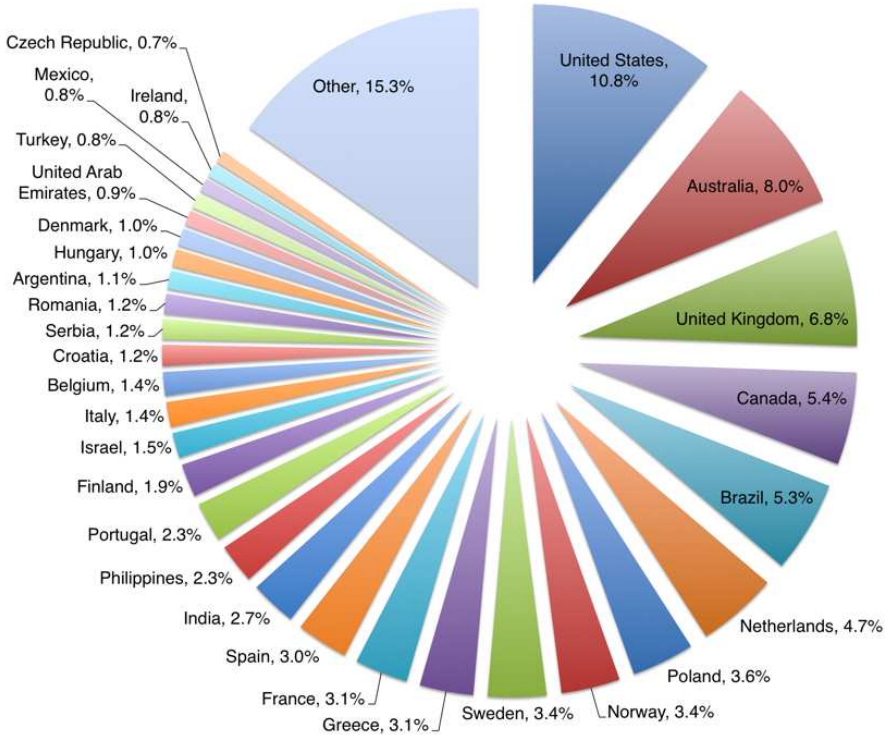

Figure A.10: Game of Thrones: Top 30 Countries

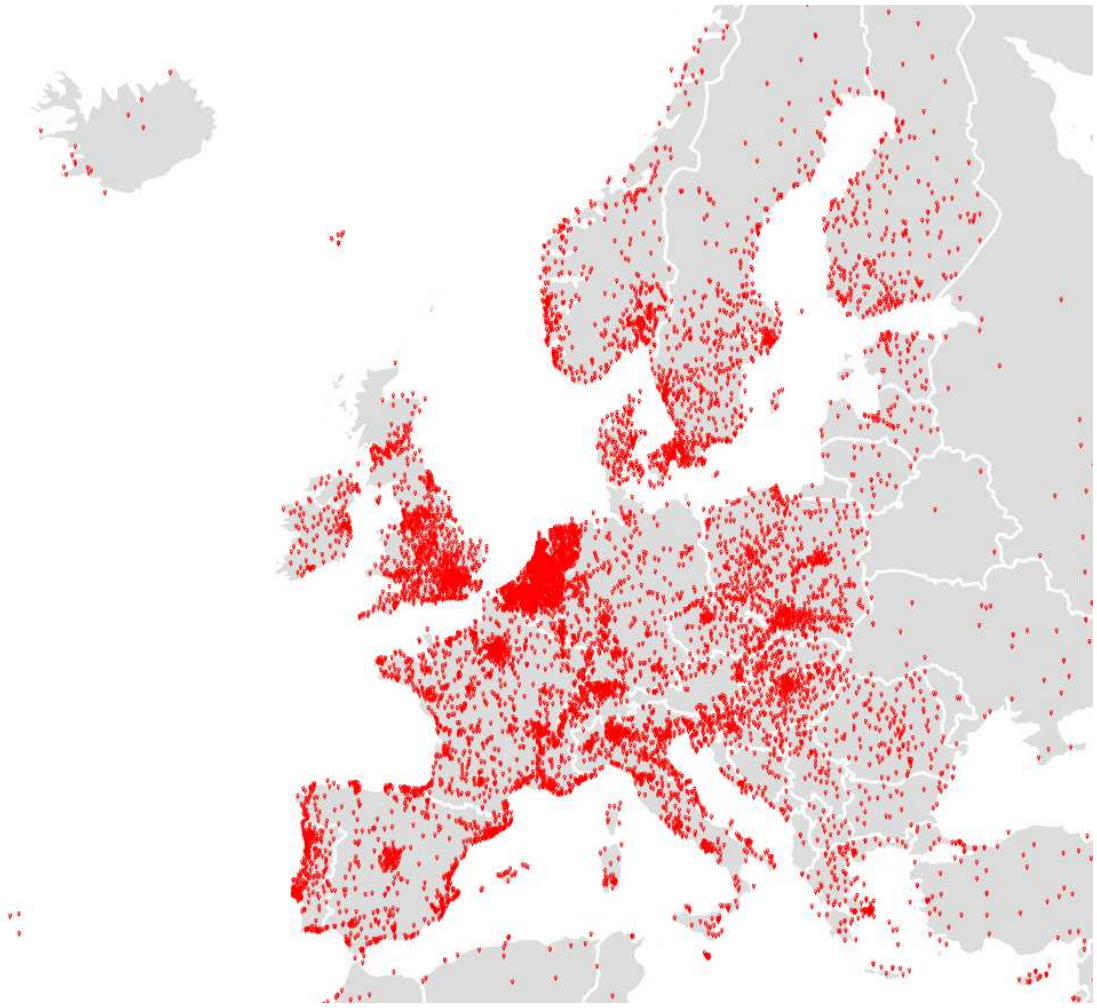

Figure A.11: Game of Thrones S03E07: Mainland Europe Activity

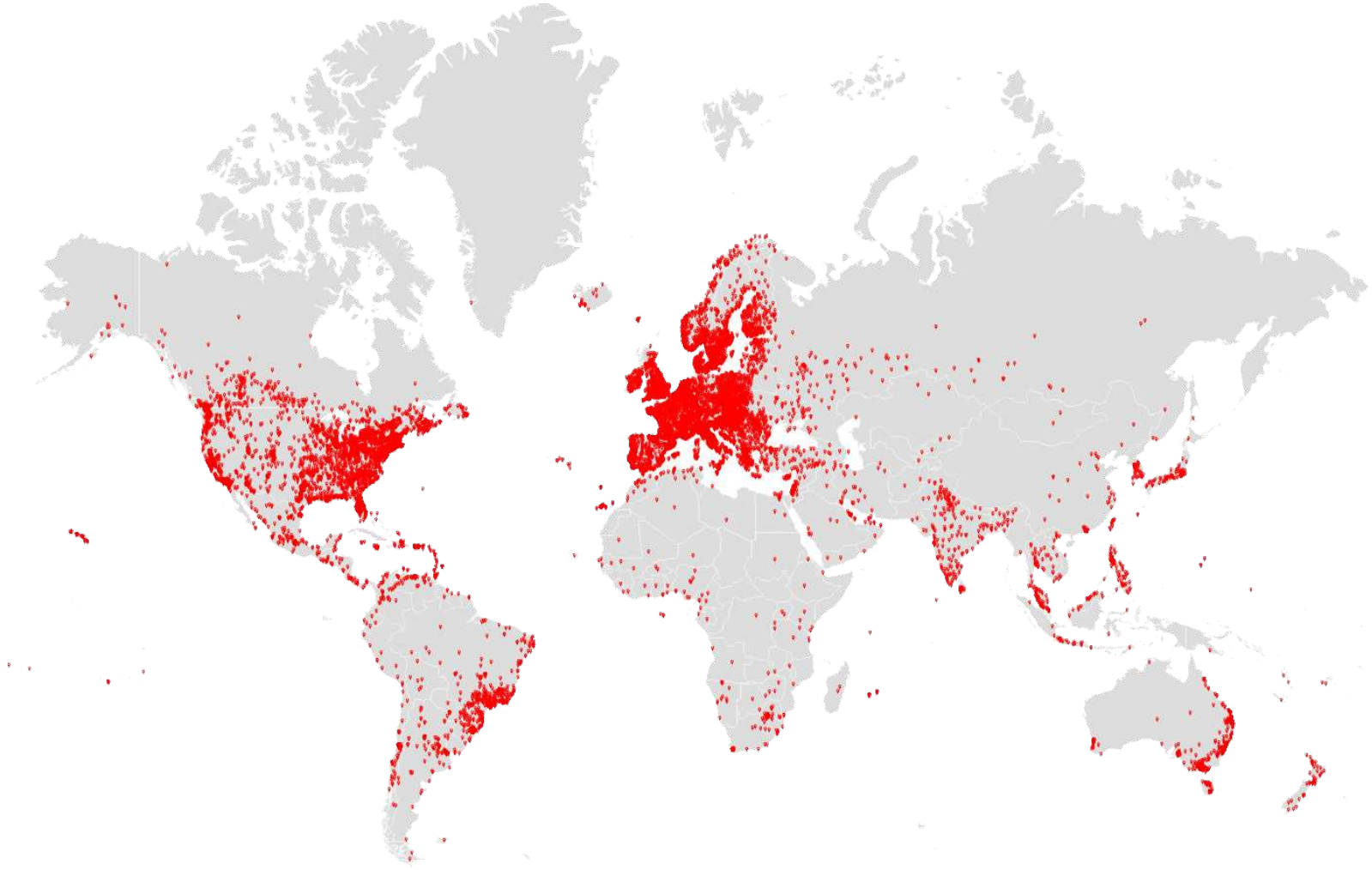

Figure A.12: Game of Thrones S03E07: Global City Level Activity

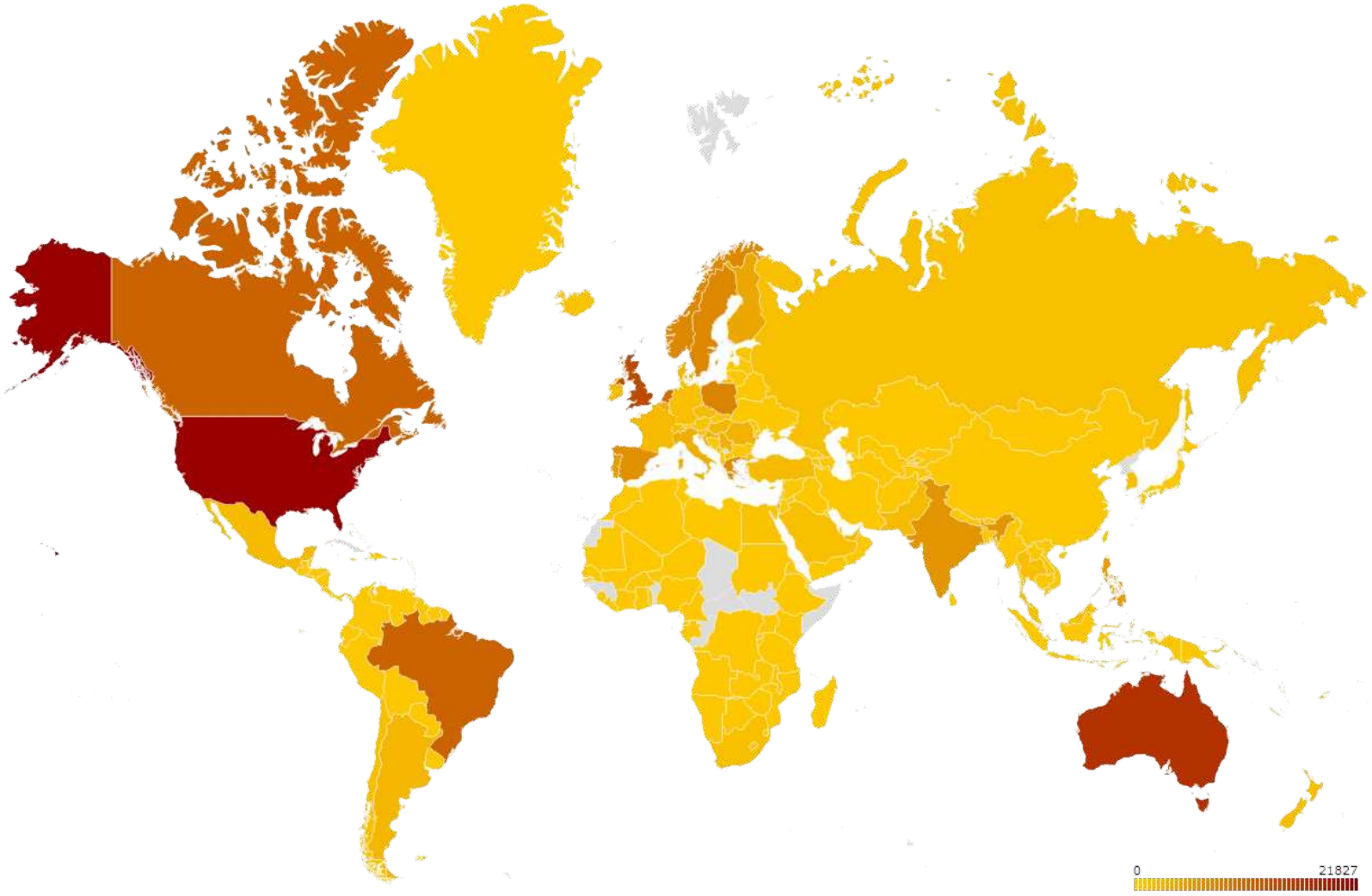

Figure A.13: Game of Thrones S03E07: Global Heatmap

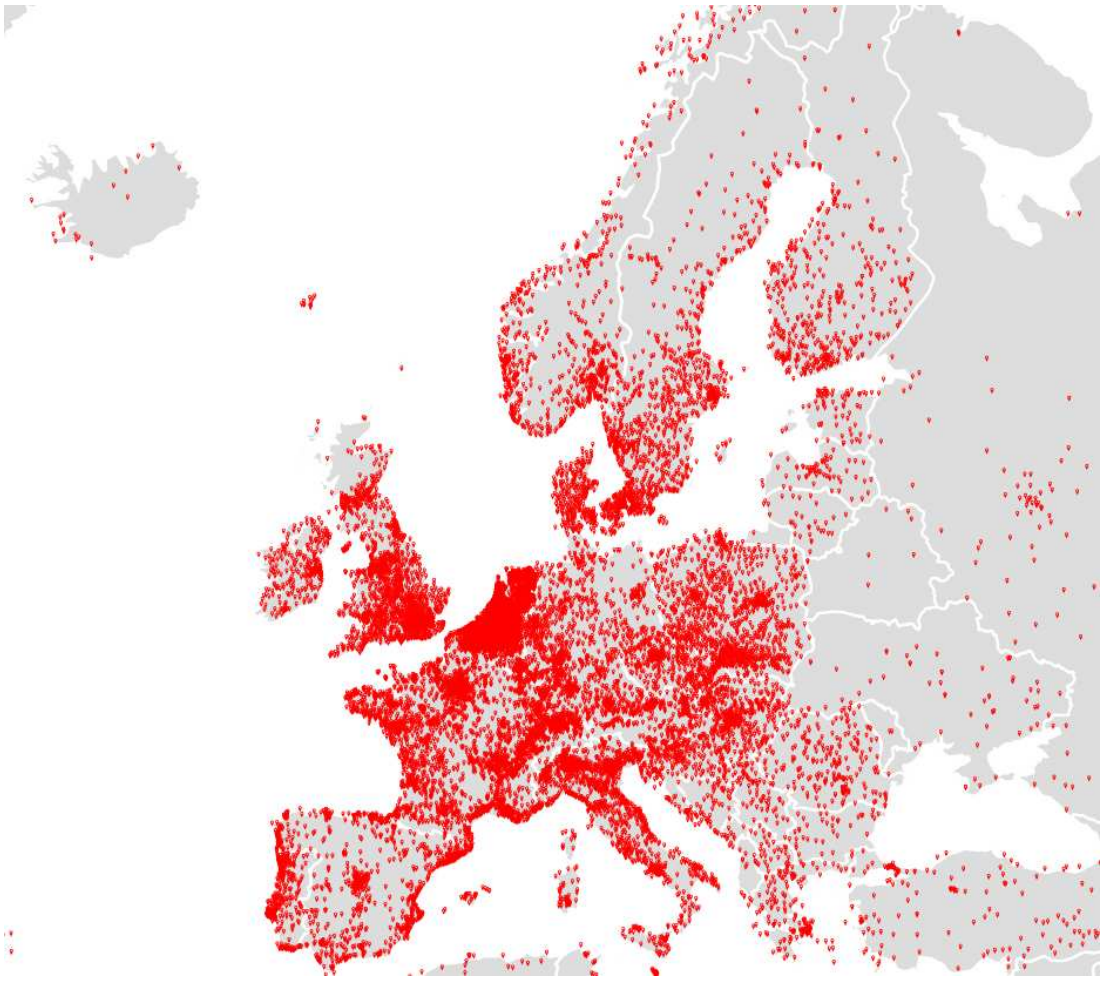

Figure A.14: Game of Thrones S03E08: Mainland Europe Activity

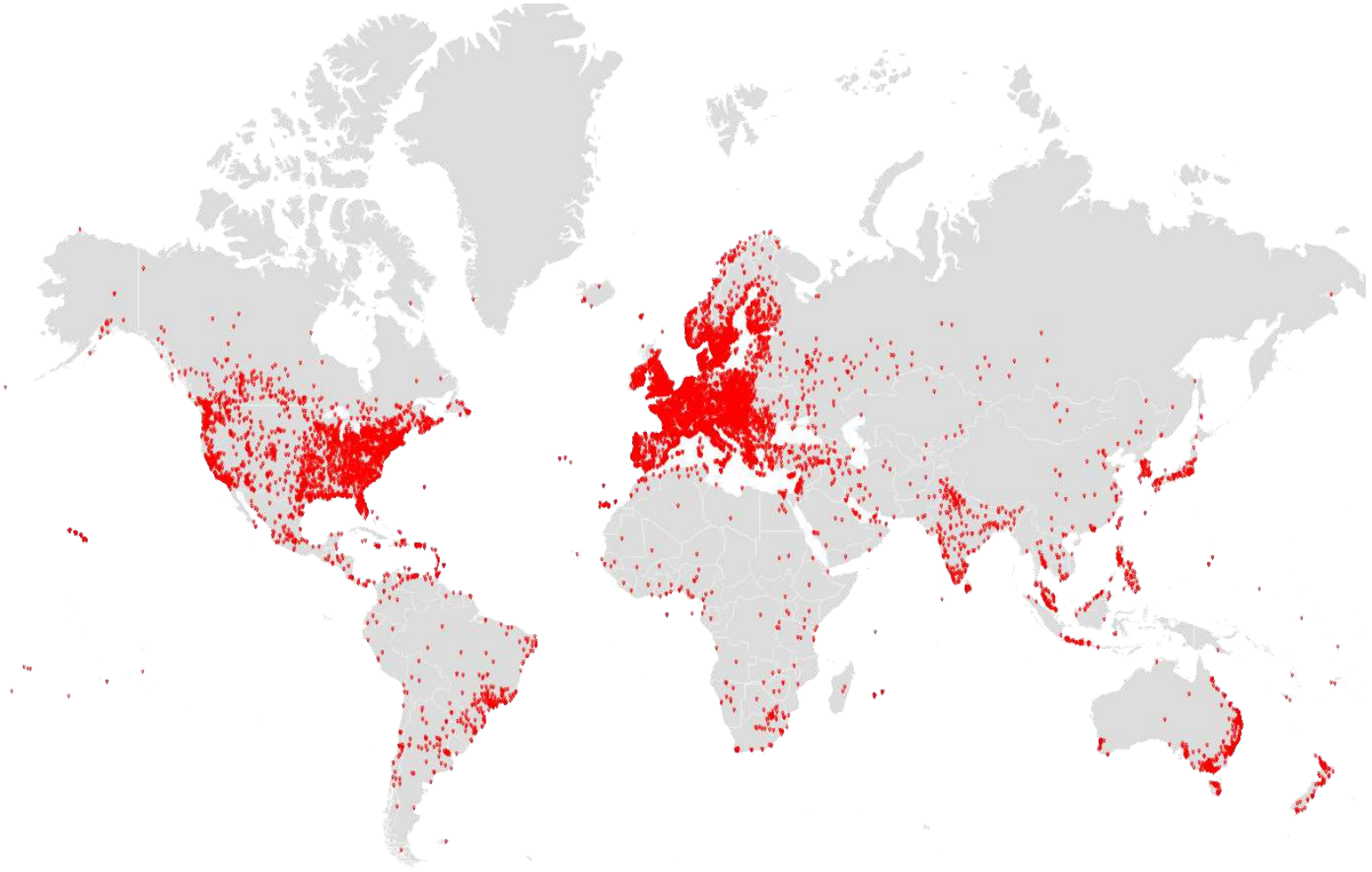

Figure A.15: Game of Thrones S03E08: Global City Level Activity

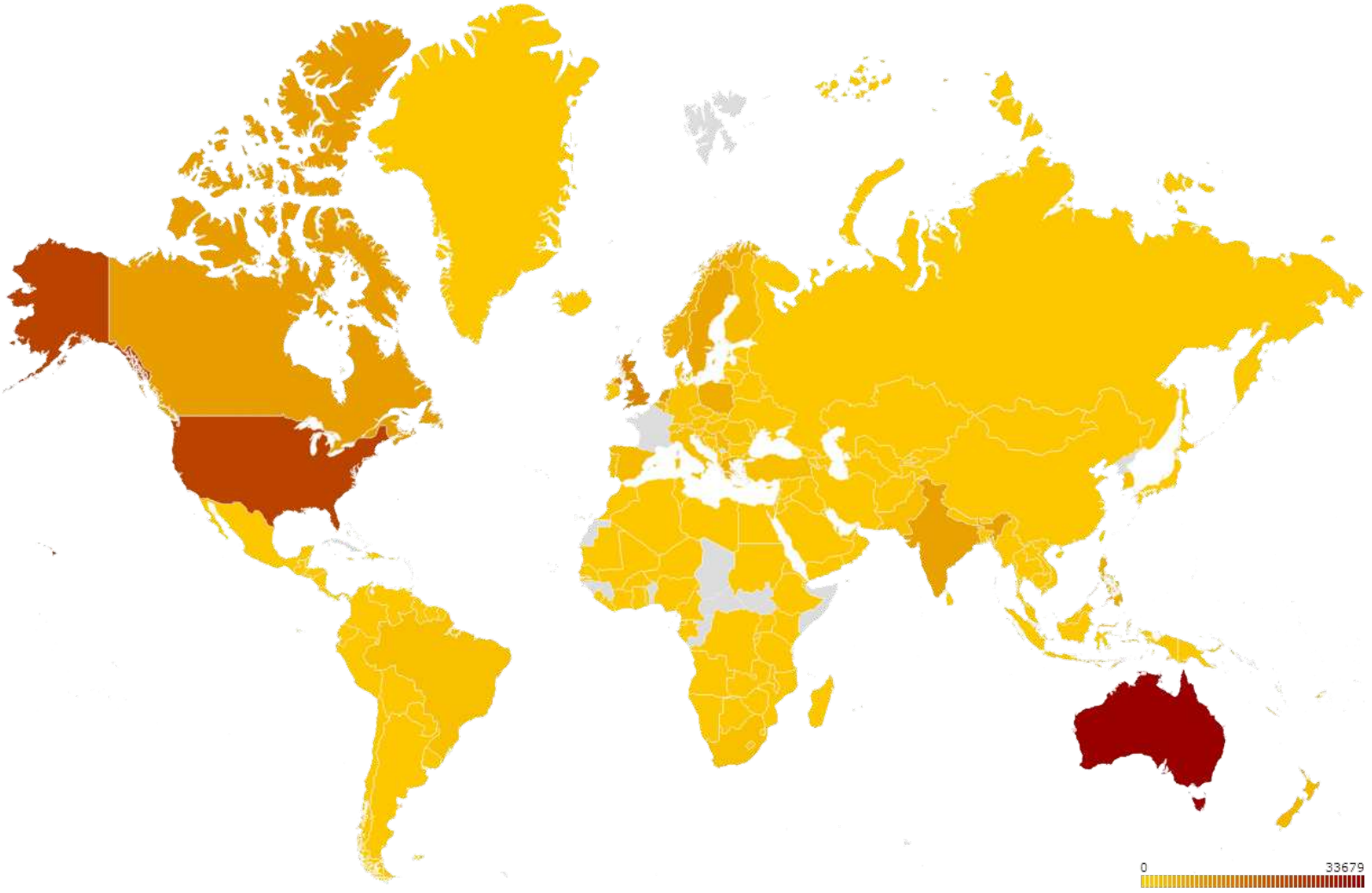

Figure A.16: Game of Thrones S03E08: Global Heatmap

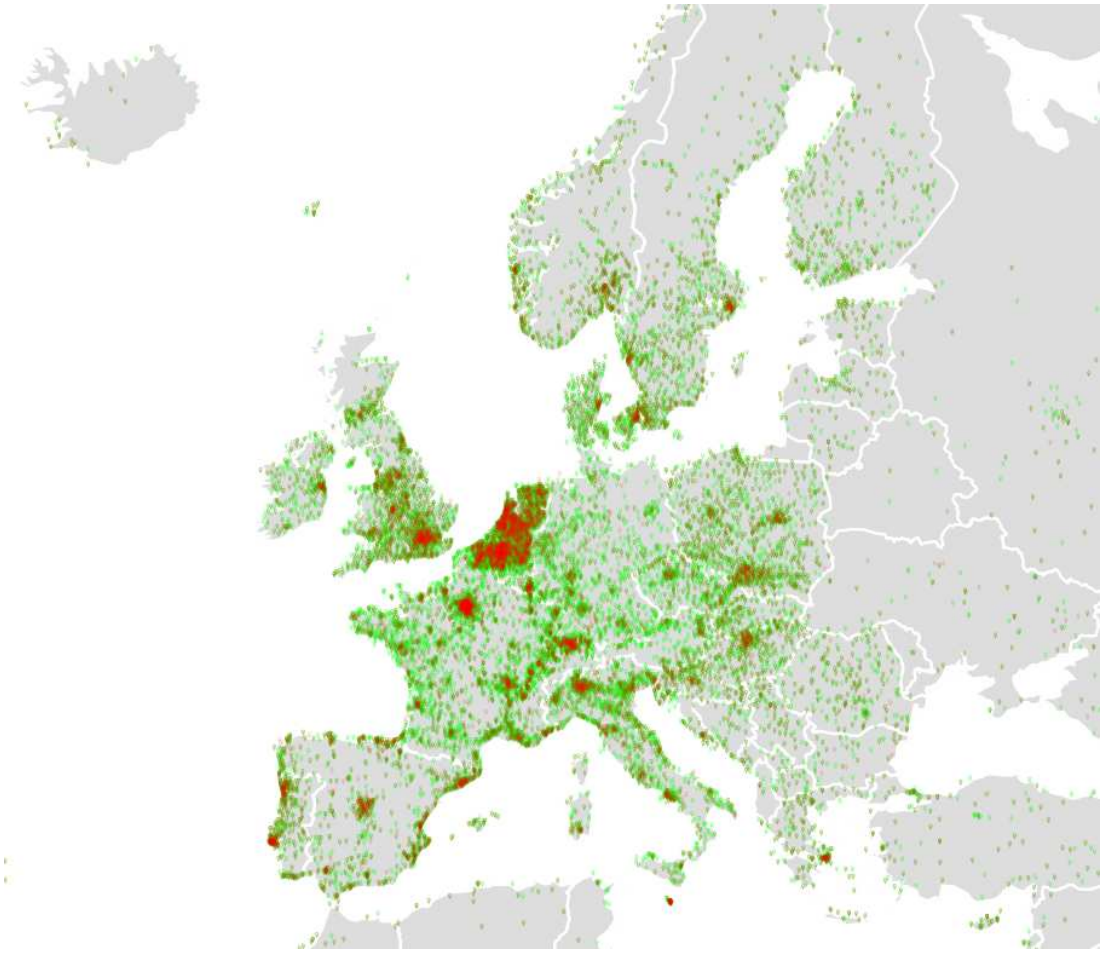

Figure A.17: Game of Thrones: Collated Results for S03E07 (Red) and S03E08 (Green) in Mainland Europe

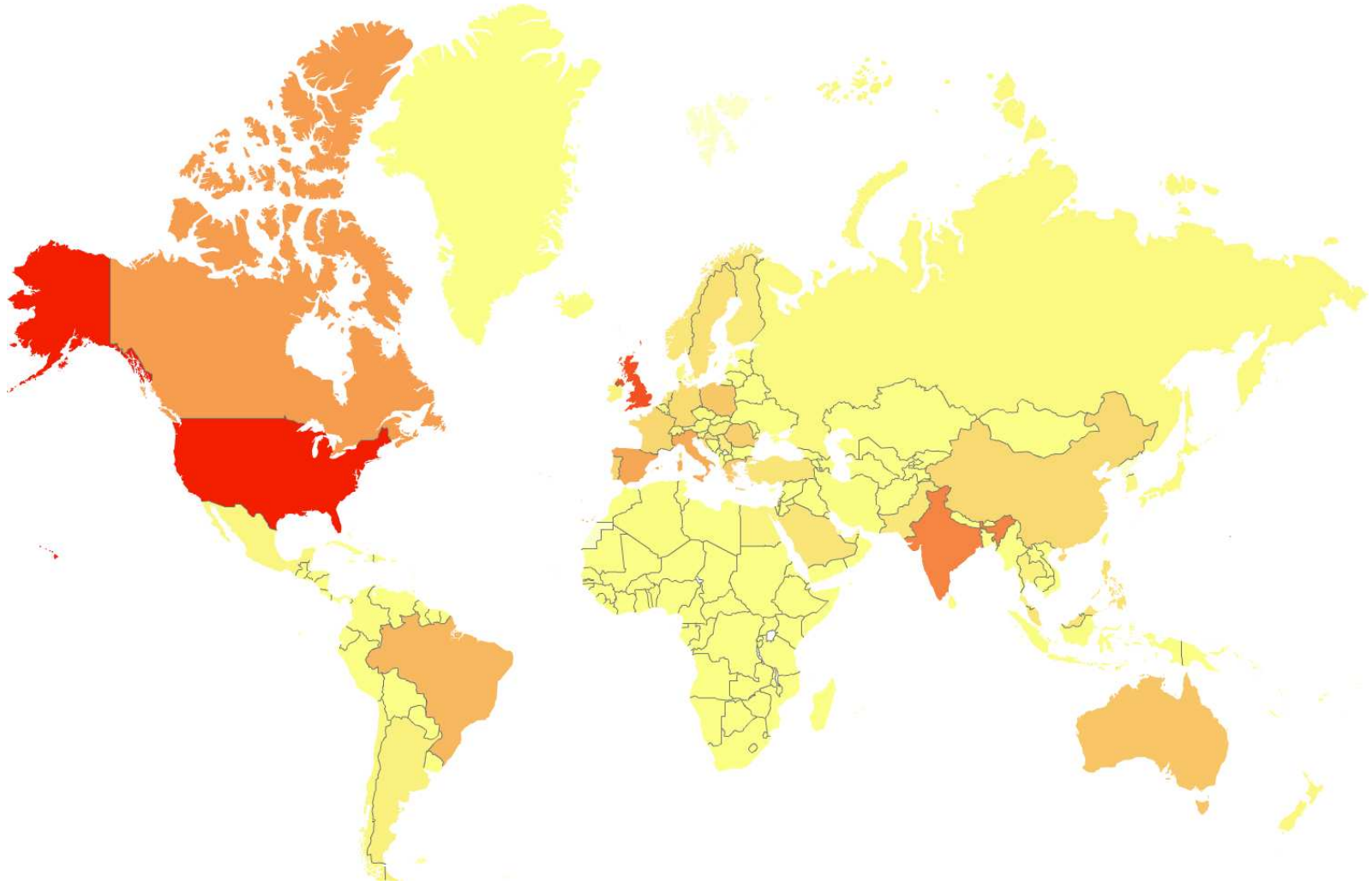

Figure A.18: Top 100 Swarms: Heatmap showing the worldwide distribution of peers discovered

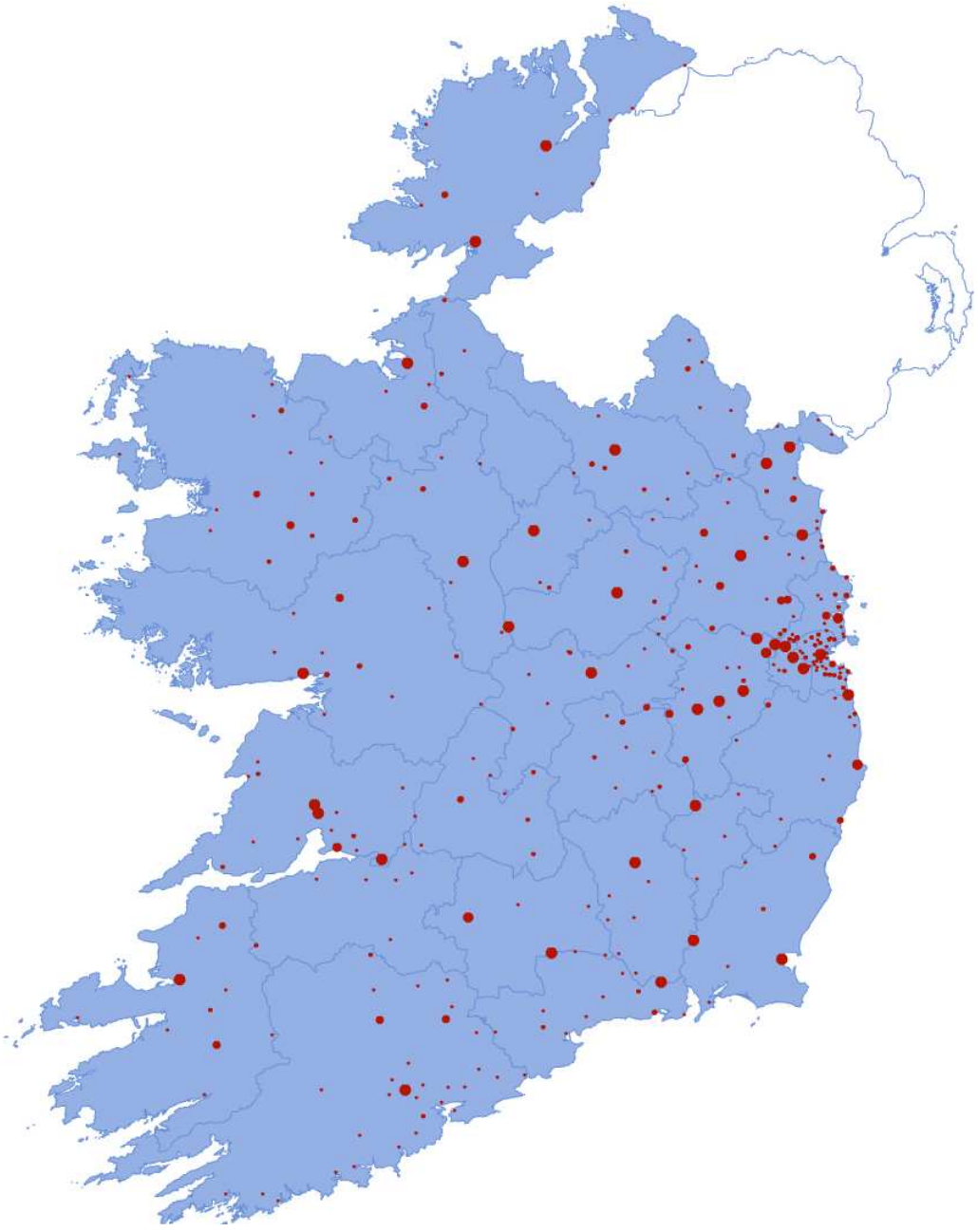

Figure A.19: Top 100 Swarms: Geolocation of the peers found across Ireland

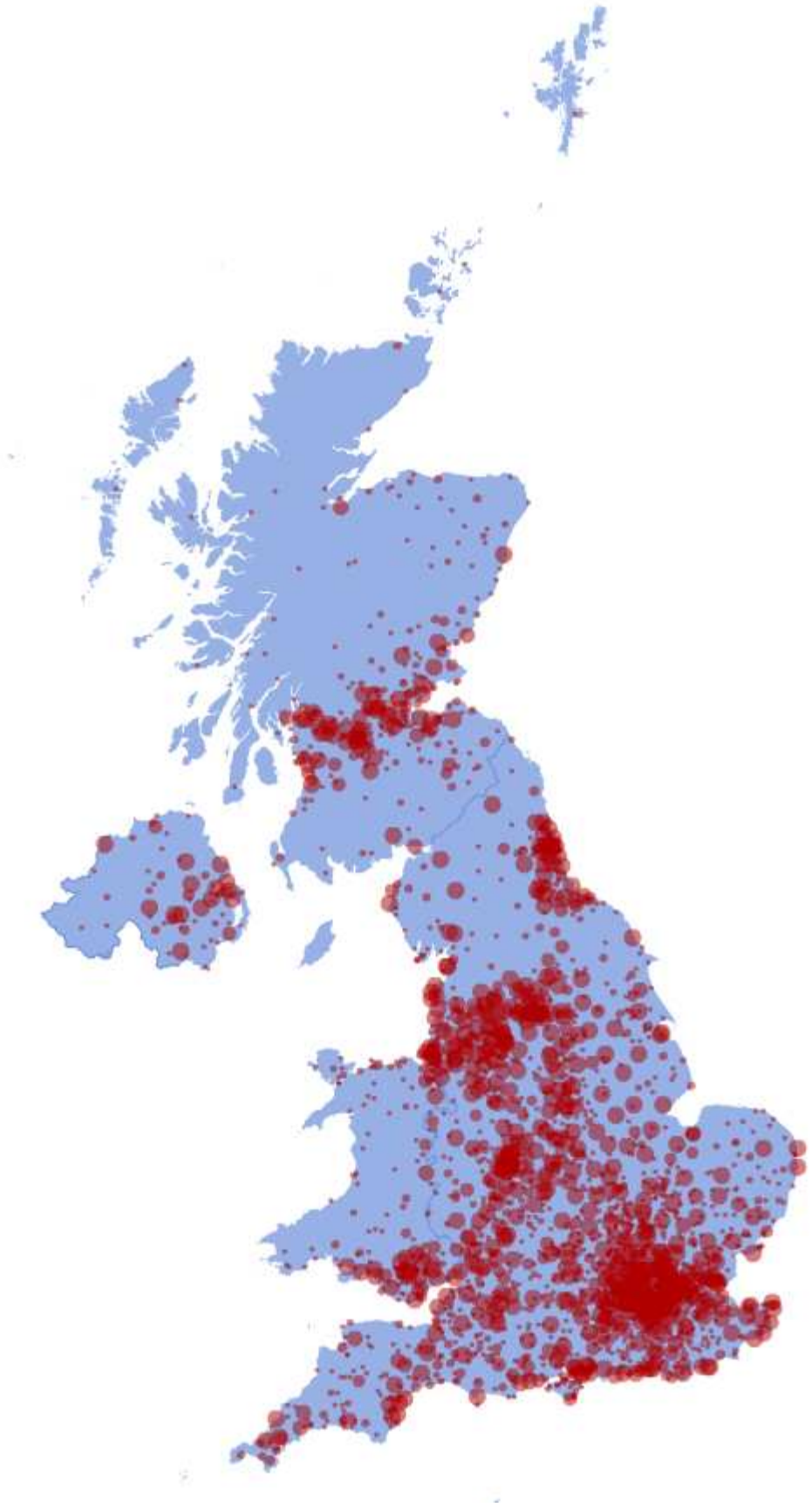

Figure A.20: Top 100 Swarms: Geolocation of the peers found across the United Kingdom

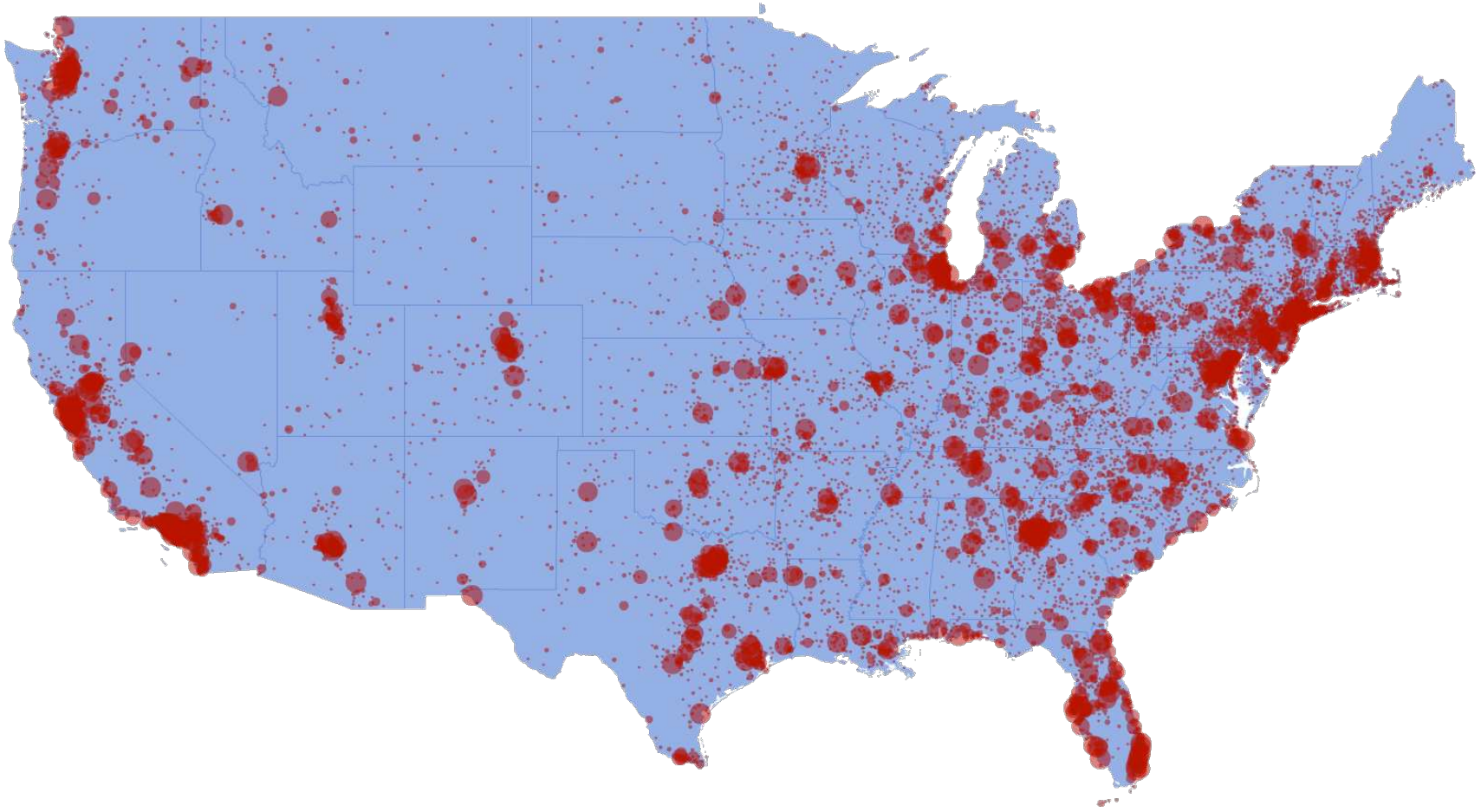

Figure A.21: Top 100 Swarms: Geolocation of the peers found across mainland USA

BIBLIOGRAPHY

[1] M. Scanlon, A. Hannaway, and M-T. Kechadi. Investigating Cybercrimes that Occur on Documented P2P Networks. *International Journal of Ambient Computing and Intelligence (IJACI)*, 3(2):56–63, 2011.

[2] M. Scanlon and M-T Kechadi. The Case for a Universal P2P Botnet Investigation Framework. In *9th International Conference on Cyber Warfare and Security ICCWS-2014*, West Lafayette, IN, USA, March 2014. ACPI, [Extended Abstract Accepted, Paper Pending Final Decision].

[3] M. Scanlon and M-T. Kechadi. Universal Peer-to-Peer Network Investigation Framework. In *International Workshop on Emerging Cyberthreats and Countermeasures (ECTCM 2013), part of the Eight International Conference on Availability, Reliability and Security (ARES2013)*, Regensburg, Germany, September 2013. IEEE.

[4] M. Scanlon and M-T. Kechadi. Digital Evidence Bag Selection for P2P Network Investigation. In James J. (Jong Hyuk) Park, Victor C.M. Leung, Cho-Li Wang, and Taeshik Shon, editors, *Future Information Technology, Application, and Service; The 8th International Symposium on Digital Forensics and Information Security (DFIS-2013)*, Lecture Notes in Electrical Engineering. Springer Netherlands, 2013.

[5] M. Scanlon and M-T. Kechadi. Peer-to-Peer Botnet Investigation: A Review. In James J. (Jong Hyuk) Park, Victor C.M. Leung, Cho-Li Wang, and Taeshik Shon, editors, *Future Information Technology, Application, and Service; The 7th International Symposium on Digital Forensics and Information Security (DFIS-12)*, volume 179 of *Lecture Notes in Electrical Engineering*, pages 231–238. Springer Netherlands, 2012.

[6] M. Scanlon, A. Hannaway, and M-T. Kechadi. A Week in the Life of the Most Popular BitTorrent Swarms. *5th Annual Symposium on Information Assurance (ASIA'10)*, 2010.

[7] M. Scanlon and M-T. Kechadi. Online Acquisition of Digital Forensic Evidence. In *Proceedings International Conference on Digital Forensics and Cyber Crime (ICDF2C 2009)*, Albany, New York, USA, September 2009. Elsevier Limited.

[8] M. Scanlon. Investigating Cybercrimes that Utilise Peer-to-Peer Internet Communications. In *Intel European Research and Innovation Conference (ERIC '10)*, 2010.

[9] M. Giesler and M. Pohlmann. The anthropology of file sharing: Consuming napster as a gift. *Advances in consumer research*, 30:273–279, 2003.

[10] E. Van Buskirk. Introducing the world's first mp3 player. *MP3 Insider*, 27, 2005.

[11] K.T. Greenfeld, C. Taylor, and DE Thigpen. Meet the napster. *Time Magazine*, 2:998068–1, 2000.

[12] Yun Gao, Golden G. Richard III, and Vassil Roussev. Bluepipe: A scalable architecture for on-the-spot digital forensics. *IJDE*, 3(1), 2004.

[13] A. Yasinsac and Y. Manzano. Policies to enhance computer and network forensics. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, pages 289–295, 2001.

[14] B. Carrier. Open source digital forensics tools: The legal argument. *@stake Research Report*, 2002.

[15] National Institute of Standards and Technologyl. *NIST Special Publication 800-86 Guide to Integrating Forensic Techniques Into Incident Response*. CreateSpace, Paramount, CA, 2012.

[16] F. Fuentes and D.C. Kar. Ethereal vs. tcpdump: a comparative study on packet sniffing tools for educational purpose. *Journal of Computing Sciences in Colleges*, 20(4):169–176, 2005.

[17] V. Corey, C. Peterman, S. Shearin, M.S. Greenberg, and J. Van Bokkelen. Network forensics analysis. *Internet Computing, IEEE*, 6(6):60–66, 2002.

[18] N. Hindocha and E. Chien. Malicious threats and vulnerabilities in instant messaging. In *Virus Bulletin Conference, vb2003*, 2003.

[19] Sourceforge.net. Ossim, the open source siem, July 2013.

[20] J.M. Khalife, A. Hajjar, and J. Díaz-Verdejo. Performance of opendpi in identifying sampled network traffic. *Journal of Networks*, 8(1):71–81, 2013.

[21] Digital Intelligence Forensic Recovery of Evidence Device (FRED). *http://www.digitalintelligence.com/products/fred/*. August 2009.

[22] Digital Forensic Research Workshop (DFRWS) Common Digital Evidence Storage Format (CDESF) Working Group. *http://www.dfrws.org/CDESF/index.shtml*. September 2006.

[23] Common Digital Evidence Storage Format (CDESF). Survey of existing disk image storage formats. In *Proc. Digital Forensic Research Workshop 2006*, September 2006.

[24] The Common Digital Evidence Storage Format Working Group. Standardizing digital evidence storage. *Communications of the ACM*, 49(2):67–68, 2006.

[25] Simson L Garfinkel. Aff: Storing hard drive images. *Communications of the ACM*, 49(2):85, 2006.

[26] G.G. Richard, V. Roussev, and L. Marziale. Forensic discovery auditing of digital evidence containers. *Digital Investigation*, 4(2):88–97, 2007.

[27] Generic Forensic Zip. *http://www.nongnu.org/gfzip/*. April 2009.

[28] P. Turner. Unification of digital evidence from disparate sources (digital evidence bags). *Digital Investigation*, 2(3):223–228, 2005.

[29] Chet Hosmer. Digital evidence bag. *Commun. ACM*, 49(2):69–70, 2006.

[30] WetStone Technologies Inc. *http://wetstonetech.com//*.

[31] L. Garber. Encase: A case study in computer-forensic technology. *IEEE Computer Magazine*, January 2001.

[32] EnCase Forensic Features. *http://www.guidancesoftware.com/WorkArea/DownloadAsset.aspx?id=671*. Guidance Software, August 2009.

[33] E. Casey. What does forensically sound really mean. *Digital Investigation*, 4(2):49–50, 2007.

[34] B. Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, 1993.

[35] X. Wang and H. Yu. How to break md5 and other hash functions. *Advances in Cryptology — EUROCRYPT 2005, Lecture Notes in Computer Science*, 3494:19–35, 2005.

[36] United States Computer Emergency Readiness Team (US-CERT). *Vulnerability Note 836068 http://www.kb.cert.org/vuls/id/836068/*. December 2008.

[37] X.-M. Zhang and Y. Zheng. Gac - the criterion for global avalanche characteristics of cryptographic functions. *Journal of Universal Computer Science*, 1(5):320–337, 1995.

[38] H. Feistel, W.A. Notz, and J.L. Smith. Some cryptographic techniques for machine-to-machine data communications. *Proceedings of the IEEE*, 63(11):1545–1554, Nov. 1975.

[39] Ronald Rivest. The md5 message-digest algorithm. 1992.

[40] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full sha-1. *Advances in Cryptology — CRYPTO 2005, Lecture Notes in Computer Science*, 3621:17–36, 2005.

[41] H. Gilbert and H. Handschuh. Security analysis of sha-256 and sisters. *Selected Areas in Cryptography, Lecture Notes in Computer Science*, 3006:175–193, 2004.

[42] National Institute of Standards and Technology (NIST). *SHA-3 Second Round Candidates*. August 2009.

[43] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C-W Phan. Sha-3 proposal blake. *Submission to NIST*, 2008.

[44] Søren S Thomsen. Pseudo-cryptanalysis of the original blue midnight wish. In *Fast Software Encryption*, pages 304–317. Springer, 2010.

[45] Daniel J Bernstein. Cubehash specification (2. b. 1). *Submission to NIST*, 2008.

[46] Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw, and Yannick Seurin. Sha-3 proposal: Echo. *Submission to NIST (updated)*, pages 1–5, 2009.

[47] Shai Halevi, William E Hall, and Charanjit S Jutla. The hash function fugue. *Submission to NIST (updated)*, 2009.

[48] Praveen Gauravaram, Lars R Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S Thomsen. Grøstl–a sha-3 candidate. *Submission to NIST*, 2008.

[49] Özgül Küçük. The hash function hamsi. *Submission to NIST (updated)*, 33:167, 2009.

[50] Hongjun Wu. The hash function jh. *Submission to NIST (round 3)*, 2011.

[51] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3, 2009.

[52] Christophe De Canniere, Hisayoshi Sato, and Dai Watanabe. Hash function luffa: specification. *Submission to NIST SHA-3 Competition*, 2008.

[53] Emmanuel Bresson, Anne Canteaut, Benoıt Chevallier-Mames, Christophe Clavier, Thomas Fuhr, Aline Gouget, Thomas Icart, Jean-François Misarsky, Marıa Naya-Plasencia, Pascal Paillier, et al. Shabal, a submission to nist's cryptographic hash algorithm competition. *Submission to NIST*, 2008.

[54] Eli Biham and Orr Dunkelman. The shavite-3 hash function. *Submission to NIST*, 9, 2008.

[55] Gaëtan Leurent, Charles Bouillaguet, and Pierre-Alain Fouque. Simd is a message digest. *Submission to NIST (round 2)*, 2009.

[56] Bruce Schneier, N Ferguson, S Lucks, D Whiting, M Bellare, T Kohno, J Walker, and J Callas. The skein hash function family. *Submission to NIST(Round 3)*, 2011.

[57] NIST Cryptographic Hash Function. *http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf*. November 2012.

[58] Science and Technology Comittee. Forensic science on trial. pages 75–76, 2005.

[59] Daubert v. Merrell Dow Pharmaceutils Syllabus Supreme Court of the United States. *http://supct.law.cornell.edu/supct/html/92-102.ZS.html/*. June 1993.

[60] United States National Institute of Standards Computer Forensic Tool Testing program and Technology. *http://www.cftt.nist.gov/*. August 2009.

[61] M. Karyda and L. Mitrou. Internet forensics: Legal and technical issues. In *Digital Forensics and Incident Analysis, 2007. WDFIA 2007. Second International Workshop on*, pages 3–12. IEEE, 2007.

[62] Geoffrey Fox. Peer-to-peer networks. *Computing in Science & Engineering*, 3(3):75–77, 2001.

[63] Rodrigo Rodrigues and Peter Druschel. Peer-to-peer systems. *Communications of the ACM*, 53(10):72–82, 2010.

[64] Cisco Systems Inc. Cisco visual networking index: Forecast and methodology, 2008-2013, July 2008.

[65] Cisco Systems Inc. Cisco visual networking index: Forecast and methodology, 2012-2017, July 2012.

[66] H. Schulze and K. Mochalski. ipoque internet study 2008/2009. `http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009`, 2009.

[67] F. Moore. Ifpi digital music report 2011, 2011.

[68] Adrian Adermon and Che-Yuan Liang. Piracy, music, and movies: A natural experiment. 2010.

[69] F. Moore. Ifpi digital music report 2012, 2012.

[70] A. Zentner. Measuring the effect of file sharing on music purchases. *Journal of Law and Economics*, 49(1):63–90, 2006.

[71] D. Serbin. The graduated response: Digital guillotine or a reasonable plan for combating online piracy? *Intellectual Property Brief*, 3(3):4, 2012.

[72] Marc Fetscherin. Movie piracy on peer-to-peer networks. the case of kazaa. *Telematics and Informatics*, 22(1):57–70, 2005.

[73] A. W. Luis and M. Bertin. Digital music consumption on the internet: Evidence from clickstream data. *European Commission Institute for Prospective Technological Studies*, 2013.

[74] M.A. Carrier. Sopa, pipa, acta, tpp: An alphabet soup of innovation-stifling copyright legislation and agreements. *Nw. J. Tech. & Intell. Prop.*, 11:21–163, 2013.

[75] Stratis Ioannidis and Peter Marbach. On the design of hybrid peer-to-peer systems. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):157–168, 2008.

[76] Jem E Berkes. Decentralized peer-to-peer network architecture: Gnutella and freenet. *University of Manitoba Winnipeg, Manitoba, Canada*, 2003.

[77] Joseph Lewthwaite. Frostwire P2P forensic examinations. *Digital Investigation*, (0):–, 2013.

[78] A. Hannaway and M-T. Kechadi. An analysis of the scale and distribution of copyrighted material on the gnutella network. In *Proceedings of the International Conference on Information Security and Privacy*, Orlando, USA, 2009.

[79] Andrew Kalafut, Abhinav Acharya, and Minaxi Gupta. A study of malware in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 327–332, New York, NY, USA, 2006. ACM.

[80] Oliver Heckmann, Axel Bock, Andreas Mauthe, Ralf Steinmetz, and Multimedia Kommunikation KOM. The edonkey file-sharing network. *GI Jahrestagung (2)*, 51:224–228, 2004.

[81] The BitTorrent Protocol Specification. http://www.bittorrent.org/beps/bep0003.html, January, 2010.

[82] D. Erman. *Bittorrent traffic measurements and models*. Blekinge Institute of Technology, Sweden, 2005.

[83] BitTorrent Inc. *http://www.bittorrent.com/*. June 2013.

[84] Marios Iliofotou, Georgos Siganos, Xiaoyuan Yang, and Pablo Rodriguez. Comparing bittorrent clients in the wild: The case of download speed. In *9th International Workshop on Peer-to-Peer Systems (IPTPS 10)*, 2010.

[85] P. Dhungel, D. Wu, B. Schonhorst, and K.W. Ross. A measurement study of attacks on bittorrent leechers. In *7th International Workshop on Peer-to-Peer Systems*, 2008.

[86] Jian Liang, Rakesh Kumar, Yongjian Xi, and Keith W Ross. Pollution in P2P file sharing systems. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1174–1185. IEEE, 2005.

[87] Stefan Saroiu, Krishna P Gummadi, and Steven D Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts. *Multimedia systems*, 9(2):170–184, 2003.

[88] Olfa Nasraoui, Deborah W Keeling, Adel Elmaghraby, George Higgins, and Michael Losavio. Node-based probing and monitoring to investigate use of peer-to-peer technologies for distribution of contraband material. In *Systematic Approaches to Digital Forensic Engineering, 2008. SADFE'08. Third International Workshop on*, pages 135–140. IEEE, 2008.

[89] Erling Wold. System for identifying content of digital data, February 7 2012. US Patent 8,112,818.

[90] Tor Project. Anonymity online, January 2010.

[91] I2P. I2p - the anonymous network, January 2010.

[92] Geon II Heo, Nam Hun Kim, Ah Ra Jo, Sae In Choi, and Won Hyung Park. A study on traceback of illegal users using anonymity technology in bittorrent. In *IT Convergence and Security 2012*, pages 169–175. Springer Netherlands, 2013.

[93] Yossi Gilad and Amir Herzberg. Spying in the dark: Tcp and tor traffic analysis. In *Privacy Enhancing Technologies*, pages 100–119. Springer, 2012.

[94] Thomas Karagiannis, Andre Broido, Nevil Brownlee, Kimberly C Claffy, and Michalis Faloutsos. Is P2P dying or just hiding?[P2P traffic measurement]. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, volume 3, pages 1532–1538. IEEE, 2004.

[95] Xi Chen, Kai Lin, Biao Wang, and Zhe Yang. Active measurements on bittorrent and emule ecosystem over the internet. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 126–129, April.

[96] Maxmind Inc. iblocklists, July 2013.

[97] Georgos Siganos, Josep Pujol, and Pablo Rodriguez. Monitoring the bittorrent monitors: A bird's eye view. *Passive and Active Network Measurement*, pages 175–184, 2009.

[98] Seungwon Shin, Jaeyeon Jung, and Hari Balakrishnan. Malware prevalence in the kazaa file-sharing network. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 333–338. ACM, 2006.

[99] Andrew Kalafut, Abhinav Acharya, and Minaxi Gupta. A study of malware in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 327–332. ACM, 2006.

[100] CJ Mielke and Hsinchun Chen. Botnets, and the cybercriminal underground. In *Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on*, pages 206–211. IEEE, 2008.

[101] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[102] Bruce Schneier. Inside risks: The trojan horse race. *Communications of the ACM*, 42(9):128, 1999.

[103] Abiola Abimbola, David Gresty, and Qi Shi. Subseven's honey pot program. *Network Security*, 2002(7):10–14, 2002.

[104] R.B. Jaiswal and S. Bajgude. Botnet technology. In *3rd International Conference on Emerging Trends in Computer and Image Processing (ICETCIP'2013)*, pages 169–175, January 2013.

[105] Dyndns.org, July 2013.

[106] No-ip.com, July 2013.

[107] Kim-Kwang Raymond Choo. Cloud computing: challenges and future directions. *Trends and Issues in Crime and Criminal Justice*, 400:1–6, 2010.

[108] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th conference on Security symposium*, pages 139–154, Berkeley, CA, USA, 2008. USENIX Association.

[109] M. Scanlon and M-T. Kechadi. Peer-to-Peer Botnet Investigation: A Review. *Future Information Technology, Application, and Service*, pages 231–238, 2012.

[110] Jochen Dinger and Oliver Waldhorst. Decentralized bootstrapping of P2P systems: A practical view. *NETWORKING 2009*, pages 703–715, 2009.

[111] Ghulam Memon, Jun Li, and Reza Rejaie. Tsunami: A parasitic, indestructible botnet on kad. *Peer-to-Peer Networking and Applications*, 2013.

[112] Dae il Jang, Minsoo Kim, Hyun chul Jung, and Bong-Nam Noh. Analysis of HTTP2P botnet: case study waledac. In *Communications (MICC), 2009 IEEE 9th Malaysia International Conference on*, pages 409 –412, dec. 2009.

[113] David Dittrich and Sven Dietrich. Discovery techniques for P2P botnets. 2008.

[114] Jose Nazario. Blackenergy ddos bot analysis. *Arbor*, 2007.

[115] Maryam Feily, Alireza Shahrestani, and Sureswaran Ramadass. A survey of botnet and botnet detection. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on*, pages 268–273. IEEE, 2009.

[116] Shaojun Zhang. Conversation-based P2P botnet detection with decision fusion. Master's thesis, Graduate Academic Unit of Computer Science, 2013.

[117] Junjie Zhang. *Effective and Scalable Botnet Detection in Network Traffic*. PhD thesis, College of Computing, Georgia Institute of Technology, July 2012.

[118] D. Dittrich and S. Dietrich. Discovery techniques for P2P botnets. *Stevens Institute of Technology CS Technical Report 2008*, 4, 2008.

[119] David Barroso. Botnets-the silent threat. *European Network and Information Security Agency (ENISA)*, 15:171, 2007.

[120] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE*, 9(3):49–51, 2011.

[121] R. Schoof and R. Koning. Detecting peer-to-peer botnets. *University of Amsterdam*, 2007.

[122] Zhen Li, Qi Liao, and Aaron Striegel. Botnet economics: uncertainty matters. *Managing Information Risk and the Economics of Security*, pages 245–267, 2009.

[123] Brett Stone-Gross, Thorsten Holz, Gianluca Stringhini, and Giovanni Vigna. The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[124] Dancho Danchev. Study finds the average price for renting a botnet. *ZDNet.com*, 26, May 2010.

[125] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647. ACM, 2009.

[126] Sufian Hameed. Leveraging email based social networks to prevent spam: Framework, system design and evaluation. 2012.

[127] Justin M Rao and David H Reiley. The economics of spam. *The Journal of Economic Perspectives*, 26(3):87–110, 2012.

[128] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, Vern Paxson, and Stefan Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 3–14, 2008.

[129] Cormac Herley and Dinei Florêncio. Nobody sells gold for the price of silver: Dishonesty, uncertainty and the underground economy. *Economics of Information Security and Privacy*, pages 33–53, 2010.

[130] Steve Mansfield-Devine. Anonymous: serious threat or mere annoyance? *Network Security*, 2011(1):4–10, 2011.

[131] Karim El Defrawy, Minas Gjoka, and Athina Markopoulou. Bottorrent: misusing bittorrent to launch ddos attacks. In *Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*, pages 1–6. USENIX Association, 2007.

[132] Edward G Amoroso and Edward Amoroso. *Cyber attacks: protecting national infrastructure*. Butterworth-Heinemann, 2012.

[133] D. Dittrich, F. Leder, and T. Werner. A case study in ethical decision making regarding remote mitigation of botnets. *Financial Cryptography and Data Security*, pages 216–230, 2010.

[134] Aiko Pras, Anna Sperotto, Giovane Moura, Idilio Drago, Rafael Barbosa, Ramin Sadre, Ricardo Schmidt, and Rick Hofstede. Attacks "anonymous" wikileaks proponents not anonymous. 2010.

[135] Kenneth C Wilbur and Yi Zhu. Click fraud. *Marketing Science*, 28(2):293–308, 2009.

[136] Alan Zeichick. Reality of botnet cyberwarfare. *netWorker*, 13(3):5–9, 2009.

[137] Raymond C Parks and David P Duggan. Principles of cyberwarfare. *Security & Privacy, IEEE*, 9(5):30–35, 2011.

[138] J. Vania, A. Meniya, and HB Jethva. A review on botnet and detection technique. *International Journal of Computer Trends and Technology*, 4, January 2013.

[139] Antti Nummipuro. Detecting P2P-controlled bots on the host. In *Seminar on Network Security, Espoo, Helsinki*, 2007.

[140] Cliff C Zou and Ryan Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 199–208. IEEE, 2006.

[141] Sanjay Goel, Adnan Baykal, and Damira Pon. Botnets: the anatomy of a case. *Journal of Information Systems Security*, 2006.

[142] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 5–5, Berkeley, CA, USA, 2007. USENIX Association.

[143] B.B.H. Kang, E. Chan-Tin, C.P. Lee, J. Tyra, H.J. Kang, C. Nunnery, Z. Wadler, G. Sinclair, N. Hopper, D. Dagon, et al. Towards complete node enumeration in a peer-to-peer botnet. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 23–34. ACM, 2009.

[144] B. Stone-Gross, M. Cova, B. Gilbert, R. Kemmerer, C. Kruegel, and G. Vigna. Analysis of a botnet takeover. *Security & Privacy, IEEE*, 9(1):64–72, 2011.

[145] Reinier Schoof and Ralph Koning. Detecting peer-to-peer botnets. Technical report, System and Network Engineering, University of Amsterdam, February 2007.

[146] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets: overview and case study. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[147] Nelly Marylise Mukamurenzi. Storm worm: A P2P botnet. Master's thesis, Department of Telematics, Norwegian University of Science and Technology, 2008.

[148] Brent Byung, Hoon Kang, Eric Chan-Tin, Christopher P. Lee, James Tyra, Hun Jeong Kang, Chris Nunnery, Zachariah Wadler, Greg Sinclair, Nicholas Hopper, David Dagon, and Yongdae Kim. Towards complete node enumeration in a peer-to-peer botnet. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 23–34, New York, NY, USA, 2009. ACM.

[149] G. Sinclair, C. Nunnery, and B.B.-H. Kang. The waledac protocol: The how and why. In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pages 69 –77, oct. 2009.

[150] Blackberry, android users targeted by new zeus trojan, July 2013.

[151] Christian Rossow, Christian Dietrich, and Herbert Bos. Large-scale analysis of malware downloaders. In *Proceedings of 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, 2012.

[152] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang. On the analysis of the zeus botnet crimeware toolkit. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 31 –38, aug. 2010.

[153] David E Sanger. Obama order sped up wave of cyberattacks against iran. *The New York Times*, 1, 2012.

[154] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 2011.

[155] Michael Conrad and Hans-Joachim Hof. A generic, self-organizing, and distributed bootstrap service for peer-to-peer networks. *Self-Organizing Systems*, pages 59–72, 2007.

[156] Robert R. Schaller. Moore's law: past, present, and future. *IEEE Spectr.*, 34(6):52–59, 1997.

[157] Steve McCanne, Craig Leres, and Van Jacobson. Libpcap, June 2012.

[158] Maxmind Inc. Geolite country database, July 2013.

[159] Kristina Chodorow. *MongoDB: the definitive guide*. O'Reilly, 2013.

[160] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. Ip geolocation databases: unreliable? *ACM SIGCOMM Computer Communication Review*, 41(2):53–56, 2011.

[161] Marco Canini, Wei Li, and Andrew W Moore. Toward the identification of anonymous web proxies. PAM, 2009.

[162] Ruei-Min Lin, Yi-Chun Chou, and Kuan-Ta Chen. Stepping stone detection at the server side. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 964–969. IEEE, 2011.

[163] P. Sealey. Remote forensics. *Digital Investigation*, 1(4):261–265, Febuary 2004.

[164] Kelly A Cole, Ramindu L Silva, and Richard P Mislan. All bot net: A need for smartphone P2P awareness. In *Digital Forensics and Cyber Crime*, pages 36–46. Springer, 2012.