9-4-2014

An Analysis of (Bad) Behavior in Online Video Games

Jeremy Blackburn
University of South Florida, jhblackb@mail.usf.edu

An Analysis of (Bad) Behavior in Online Video Games

by

Jeremy Blackburn

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Adriana Iamnitchi, Ph.D.
John Skvoretz, Ph.D.
Cristian Borcea, Ph.D.
Jay Ligatti, Ph.D.
Ken Christensen, Ph.D.
Sanjukta Bhanja, Ph.D.

Date of Approval:
September 4, 2014

Keywords: social network analysis, toxic behavior, cheating, big data, contagion

Dedication

To my friend Brandon Lorenz, with whom nearly 20 years ago I forged a friendship around online games giving me first hand experience with much of the behavior we study in this dissertation. To my friend Jon Greene, with whom I spent (too) many hours embedded in the Slaughter House game server we study in this dissertation. To my brother, Dr. Bryan Blackburn, for activating sibling rivalry as a force towards completing this dissertation. To my parents Bob and Janet, for not only believing in me, but unwittingly contributing to this dissertation by letting me spend all those hours gaming with Brandon. To my daughter Ruby, for giving me a whole new perspective on the world. To my wife Lauren, for putting up with me and being my best friend.

Acknowledgments

First, I thank my lab mates during my time in the Distributed Systems Group at USF: Dr. Nicolas Kourtellis, Xiang Zuo, and Imrul Kayes. Nicolas, in particular, was instrumental in my early work in the lab, as well as a major contributor on the analysis of cheaters in the Steam Community. I've also spent many hours discussing problems with both Xiang and Imrul, and appreciate the thought provoking conversations we've had.

I thank two of my collaborators who coauthored some of the work in this dissertation with me: Prof. Matei Ripeanu who worked with me from the beginning on the analysis of cheaters, and Dr. Haewoon Kwak who coauthored the work on toxic behavior with me.

I would like to thank my committee members, Professor John Skvoretz, Professor Cristian Borcea, Professor Jay Ligatti, Professor Ken Christensen, Professor Sanjukta Bhanja, and Professor Adriana Iamnitchi. In particular, Prof. Christensen, who oversaw my Masters dissertation and started me down this path by supervising undergraduate research work. Finally, I give my utmost gratitude to Anda, for advising and helping me grow as a scientist, for encouraging me to pursue my ideas, and pushing me to submit to the best venues possible. Without Anda's guidance, this dissertation would not have been possible.

Table of Contents

List of Tables

List of Figures

Abstract

This dissertation studies bad behavior at large-scale using data traces from online video games. Video games provide a natural laboratory for exploring bad behavior due to their popularity, explicitly defined (programmed) rules, and a competitive nature that provides motivation for bad behavior. More specifically, we look at two forms of bad behavior: cheating and toxic behavior.

Cheating is most simply defined as breaking the rules of the game to give one player an edge over another. In video games, cheating is most often accomplished using programs, or "hacks," that circumvent the rules implemented by game code. Cheating is a threat to the gaming industry in that it diminishes the enjoyment of fair players, siphons off money that is paid to cheat creators, and requires investment in anti-cheat technologies.

Toxic behavior is a more nebulously defined term, but can be thought of as actions that violate social norms, especially those that harm other members of the society. Toxic behavior ranges from insults or harassment of players (which has clear parallels to the real world) to domain specific instances such as repeatedly "suiciding" to help an enemy team. While toxic behavior has clear parallels to bad behavior in other online domains, e.g., cyberbullying, if gone unchecked it has the potential to "kill" a game by driving away its players.

We first present a distributed architecture and reference implementation for the collection and analysis of large-scale social data. Using this implementation we then study the social structure of over 10 million gamers collected from a planetary scale Online Social Network, about 720 thousand of whom have been labeled cheaters, finding a significant correlation between social structure and the probability of partaking in cheating behavior. We additionally collect over half a billion daily observations of the cheating status of these gamers. Using about 10 months of detailed server logs from a community owned and operated game server we next analyze how relationships in the aforementioned online social network are backed by in-game interactions. Next, we use the insights gained and find evidence for a contagion process underlying the spread of cheating behavior and perform a data driven simulation using mathematical models for contagion. Finally, we build a model using millions of crowdsourced decisions for predicting toxic behavior in online games.

To the best of our knowledge, this dissertation presents the largest study of bad behavior to date. Our findings confirm theories about cheating and unethical behavior that have previously remained untested outside of controlled laboratory experiments or only with small, survey based studies. We find that the intensity of interactions between players is a predictor of a future relationship forming. We provide statistically significant evidence for cheating as a contagion. Finally, we extract insights from our model for detecting toxic behavior on how human reviewers perceive the presence and severity of bad behavior.

Chapter 1: Introduction

The past decade has seen an explosion of computer mediated social interactions. Online social networks such as Facebook have enabled billions of people to communicate and share information with their friends. Social media applications like Twitter have become primary sources for the spreading of news. Multiplayer video games are poised to become the largest segment of the entertainment industry.

While users of these services benefit in a variety of ways, the huge amount of data they generate also enables the study of social phenomena at previously unheard of scales. For example, empirical studies in the social sciences tend to involve sample sizes of a few thousand individuals at most, whereas social network systems expose the relationships and interactions between millions, and in some cases billions, of individuals. Further, the digital nature of these services means that researchers can eschew the use of surveys in favor of hard ground truths.

One area of research that is particularly enabled is the study of bad behavior. In this dissertation, through social network analysis and machine learning techniques, we quantify and model bad behavior in online video games, leading to suggestions for the detection and prevention of this behavior as well as insights into unethical behavior at large.

Studying bad behavior in online games can serve to better understand the behavior of individuals that abuse the shared social space in large-scale non-hierarchical communities.

In online social networks, for example, such individuals abuse available, legal tools, like communication or tagging features, for marketing gains, political activism, or cyberbullying. Taken to the extreme, such behaviors lead to the tragedy of the commons: all game players become cheaters and then abandon the game, or corruption escalates and chaos ensues.

The remainder of this chapter provides background on the problems and methodology this dissertation addresses. In Section 1.1, we give a brief introduction to social network analysis, which we make heavy use of in this dissertation. Section 1.2 discusses the use of video games as living labs for social research. Cheating and toxic behavior, the two unethical behaviors we study in this dissertation, and their impact are introduced in Section 1.3. Finally, in Section 1.4 we summarize and provide an outline for the remainder of this dissertation.

1.1 Social Network Analysis

Much of the work in this dissertation is concerned with social network data. A social network, also referred to as a social graph, is simply a collection of *vertices* which represent people, also called *actors*, and *edges* representing the social relationships that connect them. More formally, a social network is a graph $G(V, E)$, where a vertex v exists in the graph if $v \in V$, and an edge $e(u, v)$ exists if there is a relationship between vertices u and v. A graph can be either *directed* or *undirected*. If the graph is directed, $e(u, v)$ is a different edge than $e(v, u)$.

Social networks have been studied in sociology for several decades. They provide researchers with a stronger data point than simply looking at various attributes aggregated across

individuals: the relationship between actors. The study of social networks has proven successful at providing quantitative explanations of phenomena.

For example, perhaps the most well known social network experiments, by Milgrim, tested how information could spread between people [TM69]. The experiments sent a package to a randomly selected source individual, asking that they do their best to ensure the package was received by a specific target person located in Boston. If the source did not know the target on a first name basis, she would instead forward it to a go between, whom she knew on a first name basis, that might have a chance at knowing the target. The result of this experiment was that the package arrived after passing through a relatively small number of go-betweens. I.e., Americans seemed to form a small world network, with around "six degrees of separation"[1].

While social networks have proved fertile for scientific discovery, more recently, they have become important to the computing world. Social applications like Facebook have become ubiquitous, accessed by desktop computers, mobile phones, TVs, and even cars. They have changed the way news is delivered [KLPM10], the way we do business [SR03], and the way we form communities [bE07]. These applications require special concern on their design in terms of how to distribute computation and storage, privacy implications, as well as user and application programming interfaces, among others. Because of their ubiquity and popularity, they also serve as excellent platforms for empirical study of planetary scale social networks.

Social network analysis is the study of structural properties of social networks. The structure of a network has proven to be a robust method for drawing insights into underlying social phenomena. For example, the strength of a relationship can be estimated by looking

[1]This phrase was not coined (or used) by Milgram, but has become popularly associated with the small world theory.

at how much overlap there is between the two individual's friends [OSH+07]. These insights help us learn about how other people influence, and are influenced by, our behavior.

1.2 Video Games: Living Labs for Social Research

In the past 30 years or so, video games have transitioned from being seen as a children's activity to becoming a huge chunk of the entertainment industry. What started as simple games like PONG has grown to become a billion dollar industry with hundreds of millions of people playing together over the Internet, forming planetary scale social networks. Considering the time and financial investment people put into games, the research community has shown increasing interest in studying games from a variety of perspectives.

Studying video games as a scientific endeavor can have huge impact for the industry. Quantification and empirical evidence can inform game developers on design decisions, aid in the advancement of gaming technology, and provide new insight into the minds of gamers. This is important for several reasons. First, the gaming industry has driven significant amounts of innovation in computing technology [Gil10]. Second, gaming skill has been shown to be a predictor of real-life ability, as well as increasing certain skill sets. For example, skill in Nintendo Wii games has been shown to be a predictor of laparoscopic (minimally invasive) surgery skills [BASS+10]. Finally, as people are spending an increasing amount of their time in online games [Cro11], the development and crafting of games has a direct impact on a large number of social interactions.

Edward Castronova has put forth the theory that video games can serve as living laboratories allowing for novel social science research [Cas06]. The rules of the game world are not just explicitly known, they are in fact completely *controlled* by the game developers.

This in turn allows us to study what amounts to real world behavior, but still have some well defined controls.

For example, economists face a difficult struggle when it comes to experimental methodology: it is nearly impossible to perform controlled experiments at scale. Further, although historical data is helpful for modeling purposes, there is a large degree of uncertainty when it comes to accuracy and completeness of this type of data. Video games offer an exciting alternative, however.

With millions of players participating in virtual economies, large scale studies become much more feasible. Because it is a virtual economy operating within the confines of the developer's programming, and since the data is digital, there are few questions about accuracy or completeness [CWC+09]. In fact, the abundance of data means that even more sophisticated analysis can occur. Thus, video game studies can provide real benefit to academic inquiry, and enable the answering of questions applicable outside the virtual world.

1.3 Bad Behavior in Online Games

The popularity of online gaming supports a billion-dollar industry, but also a vigorous cheat code development community that facilitates unethical in-game behavior. "Cheats" are software components that implement game rule violations, such as seeing through walls or automatically targeting a moving character. It has been recently estimated that cheat code developers generate between $\$15,000$ and $\$50,000$ per month from one class of cheats for a particular game alone [APB11]. For some cheaters, the motivation is monetary: virtual goods are worth real-world money on eBay, and online game economies pro-

vide a lucrative opportunity for cyber criminals [KAW+10, KCWC07]. For others, the motivation is simply a competitive advantage and the desire to win [NRCS10]. And finally, some cheaters simply want to have fun and advance to a higher level in the game without investing tedious effort [Con07].

In all cultures, players resent the unethical behavior that breaks the rules of the game: "The rules of a game are absolutely binding [...] As soon as the rules are transgressed, the whole play-world collapses. The game is over [Hui50]". Online gamers are no different, judging by anecdotal evidence, vitriolic comments against cheaters on gaming blogs, and the resources invested by game developers to contain and punish cheating (typically through play restrictions).

Cheating is seen by the game development and distribution industry as both a monetary and a public relations problem [Con07] and, consequently, significant resources are invested to contain it. For example, *Steam*, the largest digital distribution channel for PC games, employs the Valve Anti-Cheat System (VAC) that detects cheats and marks the corresponding user's profile with a permanent, publicly visible (regardless of privacy setting), red, "ban(s) on record". Game servers can be configured to be VAC-secured and reject players with a VAC-ban on record matching the family of games that the server supports. The overwhelming majority of servers available in the Steam server browser as of October 2011 are VAC-secured. For example, out of the 4,234 Team Fortress 2 servers available on October 12, 2011, 4,200 were VAC-secured. Of the 34 servers that were not VAC-secured, 26 were owned and administrated by a competitive gaming league that operates its own anti-cheat system. Like many gaming environments, Steam allows its members to declare social relationships and connect themselves to *Steam Community*, an online social network.

Dishonesty, prevalent in society [Ari12], permeates the online gaming world as well, raising technical, legal and social challenges. Understanding cheaters' position in the social network that connects online gamers is relevant not only for evaluating and reasoning about anti-cheat actions and policies in gaming environments, but also for studying dishonest behavior in society at large.

For example, a study of large-scale cheating behavior can provide evidence to validate theories from social sciences and psychology on the nature of unethical behavior in general [GAA09]. Studying a gaming network is particularly interesting because of the *competitive* nature of many multiplayer games, that has parallels in the real world, possibly describing corruption mechanisms in cases such as Enron, where "internal [group] competition could set the stage for the diffusion of 'widespread unethical behavior'" [KOS08].

Another prevalent cheating phenomenon is in academia, from plagiarizing in student assignments to falsifying data in research. Research into academic cheating has indicated the presence of a network effect. For example, the acceptance of a single high school cheater into a United States military service academy (typically thought of as bastions of honor and integrity) has been shown to cause a statistically significant 0.37 to 0.47 additional students to cheat [CMW08]. A study of 158 private universities [RK09] showed that observing other undergraduate students cheat was strongly correlated with one's own cheating behavior. What has been lacking, for the most part, is an empirical investigation into how cheating behavior diffuses through the relationships of a social network.

Another area where understanding cheating in games is relevant is gamification. Gamification attempts to import gameplay mechanics to otherwise non-gaming environments, with lofty goals such as increasing engagement, participation, and even performance. It

has become a hot topic in both science [TMD12, WH12, LZ12, LC11, DSN+11, HL10] and industry [Inc12b, Inc12a, Gar11], emphasizing even more the importance of understanding gaming, and the deviant behavior associated with it. As gamification becomes increasingly ubiquitous, the threat of cheating will become more prominent, and its effects more profound.

In fact, Foursquare, one of the most popular location-based social networks and an early innovator in gamification, has experienced "pervasive" cheating [Gla11]. In Foursquare, users "check-in" to physical locations, receiving points for various tasks. Badges are given to encourage service usage (e.g., the "Superstar" badge given for checking into 50 different venues), for various achievements (e.g., the "Douchebag" badge is given for checking in to 25 locations with a "douchebag" tag), and have been converted into a revenue stream by charging for sponsored "Partner Badges" which are often accompanied by special offers at a venue. Cheating not only diminishes the achievements of fair players, but poses a direct threat to Foursquare's business model as cheaters circumvent the rules set up by Foursquare's paying partners for badges and special offers.

Cheating in Foursquare is simple: lie about your location. By falsifying check-ins, a user can gain badges and honorifics that he would not attain legitimately. Effort has been invested into the detection of cheaters on Foursquare [PKZ12, HLR11], but not to understanding how cheating propagates. Although useful, it seems unlikely that detection mechanisms will transfer to other gamified systems. While methods for protecting against and detecting cheaters tend to be domain specific, the process by which the cheating behavior spreads is unlikely to differ in any fundamental sense, and thus remain applicable to the entire spectrum of gamified systems.

Toxic behavior is another type of unethical behavior that is prevalent in online games and the real world. While cheating is relatively well defined and easy to understand, toxic behavior is a bit more ambiguous. Toxic behavior has been an area of focus in management research, where it has been been noted to act like a "poison" and "contaminate" individuals and entire organizations, causing "emotional pain," among other unwanted consequences [Gol08]. Thus, things like bullying, deliberate undermining of co-workers, and abusive leadership have been identified as manifestations of toxic behavior. As in the real world, toxic behavior can permeate a gaming community, lowering player enjoyment and ultimately make people stop playing.

One confounding issue with the study of toxic behavior is that perceptions vary between individuals [Pel10]. For example, players of an "unorthodox" basketball coach who exhibited abusive behavior when they performed poorly showed no consensus on their opinion about his behavior. For some, he was the best coach they had played for, while others were scared or angered.

Because the rules of gaming are more codified, and because we can extract a huge amount of precise data, it opens up an opportunity for the quantitative study of toxic behavior. Like cheating, a quantitative analysis of toxicity in games can produce insights not only into gamer behavior, but also provide useful information for studies in other domains.

1.4 Dissertation Outline

This dissertation is an exploration of (bad) behavior in online video games. Using tools from social network analysis and machine learning, we study two specific forms of bad behavior: cheating and toxic behavior. We collect a dataset of over 10 million gamers from

the Steam Community online social network, and in conjunction with a 10 month data trace from a community owned and operated Team Fortress 2 (TF2) game server study cheaters and how cheating behavior spreads. Additionally, we use crowd sourced decisions on several million instances of toxic behavior in the League of Legends (LoL) to analyze and model toxic behavior.

In summary, this dissertation makes the following contributions:

1. An architecture for the collection of longitudinal social network data.
2. A prototype implementation of this architecture.
3. A social network analysis of (cheaters in) the steam community.
4. An analysis of in-game interactions from a popular server.
5. Evidence for cheating as a contagion.
6. A data driven simulation of cheating as contagious behavior.
7. A model for predicting crowdsourced decisions on toxic behavior.

The remainder of this dissertation is organized as follows. Chapter 2 reviews related work that this dissertation builds on. Our architecture and reference implementation are presented in Chapter 3. The datasets we study are discussed in Chapter 4. In Chapter 5 we perform a social network analysis of gamers in the Steam Community online social network, with a focus on the position of cheaters. We explore the in-game interactions backing the declared relationships in Steam Community in Chapter 6. Chapter 7 we find evidence for cheating as a contagion process and perform a data driven analysis of a contagion model. In Chapter 8 we produce a model for predicting crowdsourced decisions on toxic behavior. Finally, we conclude and discuss our findings in Chapter 9.

Chapter 2: Related Work

In this chapter, we present previous research that this dissertation builds upon. First, we discuss research into collecting and analyzing online social network data. Next, we discuss using games for scientific research. We then discuss work related to cheating and unethical behavior. Finallly, we present previous work on toxic behavior.

2.1 Collecting and Analyzing Online Social Network Data

Chau et al. appear to have published the first work on parallel crawlers for online social networks [CPWF07]. They describe a system that uses a breadth first search (BFS) and a centralized queue to assign users to a set of independent crawler tasks. Chau et al.'s system was not designed to perform longitudinal data collection, and thus lacks a scheduler, an analytics engine, and thoughts on how to deal with a growing dataset over time. We address these concerns via the architecture presented in Chapter 3.

Willinger et al. argued that the reliance on static datasets by OSN researchers was detrimental to understanding the real processes that govern human interaction [WRT+10]. By taking measurements at various points in time from a constructed "toy" graph with dynamic properties, they illustrate that overlooking this dynamism can lead to misleading or

downright wrong results. Ultimately, they propose a multi-scale approach whereby graphs are analyzed at various granularities of spatial and temporal resolution.

Wilson et al. [WSPZ12] questioned the meaning of friendships in declared OSNs by examining interactions between users on Facebook. They define an interaction graph to be a subset of the Facebook social graph (since Facebook interactions are limited to individuals with declared friendship) where the two end points had n interactions over a time interval t, and show that users only interacted with a small subset of their friends, that interaction degree is not correlated with social degree, and that small-world clustering decreased as the interaction graph becomes more restrictive. One aspect they leave open for future work is the construction of interaction graphs that are *not* a strict subset of the social graph. The work in Chapter 6 is a first step towards filling that gap since our interactions exist in a separate, although related, context than the OSN they back.

Finally, a preliminary report by Souravlias et al. presents an architecture [SKP12] with the same basic components as seen in Figure 3.1. The indication is that work towards an implementation is still being performed, but we suspect they will face many of the same challenges outlined in Chapter 3.

2.2 Game Studies

Gaming studies have generally fallen into two categories: 1) technological and 2) behavioral. Technological gaming studies have characterized network traffic due to gaming, resource provisioning, work load prediction, and player churn in online games [CFS$^+$10, CLW03, FCFW02, FCFW05, FF03]. While the actual cheating in online games, especially the cheating that is most likely to occur within our dataset, is accomplished via technical

means, e.g., the modification of game code, the present work is primarily focused on social aspects. Thus, we limit the majority of our discussion on related work to the study of gamers' behavior.

Before we begin in earnest, it is important to discuss how the study of online games has arisen as a legitimate scientific endeavor. Edward Castronova appears to have made one of the first comprehensive arguments for viewing online games as large scale natural experiments [Cas06]. In a nutshell, his thesis is that the scale, breadth, and depth of online games results in genuine social interactions, providing detailed, precise, and accurate traces at the *society* level. While services like Steam Community (Section 4.1) and games like League of Legends (4.3) have tens of millions of active users, even the single server dataset we make use of (Section 4.2.1) involves over 30,000 people, much too many to study in the lab.

As Castronova puts it, real-world society can be viewed as a large game: there are choices to be made, the outcome is uncertain, and there are competing interests. Thus, as theories of human interaction are portable across cultures, for example, the "invisible hand" in the economies of Ancient Greece and today, they should also hold true across large games. In other words, we can empirically test social science theories at unprecedented scales in a completely specified system under controlled conditions.

Using two case studies, Castronova shows that coordination games, which have been theorized to be a driving force behind societal interactions, are present in online games. Further, because the games themselves had replication (via multiple servers, and thus societies, running in parallel), statistically significant coordination effects were found. Thus, the success is not (just)that coordination effects were present, but that he found the evidence *empirically*.

In another work, Castronova et al. [CWC+09] use traces of virtual goods transactions to measure whether there is a mapping principal for economic theories. In other words, do virtual world denizens operate in the same way as real world economies? Their findings indicate yes. They are able to quantitatively measure several economic indicators at large scale due to the accuracy and completeness of digital traces. Like the work presented in this dissertation, having access to detailed data eliminates many of the concerns of traditional survey based studies.

Although Steam Community, the OSN of Steam users that we study throughout this dissertation, in particular has not been studied before, other social networks of online gamers have been addressed in recent studies. Szell and Thurner [ST10] provide a detailed analysis of the social interactions between players in Pardus, a web-based Massively Multiplayer Online Game. By studying the socio-economic aspects of players, they provide additional evidence that communities of gamers can serve as a proxy for real world behavior.

Xu et al. interviewed 14 Halo 3 players to study the meaning of relationships within an online gaming context [XCS+11]. They found evidence of in-game relationships supported by real-world relationships, triadic closure of relationships making use of both real and virtual relationships as a bridge, and in-game interactions strengthening ties in the real world. They further found evidence of social control as a tool for managing deviant behavior. Mason and Clauset investigated the behavior of Halo: Reach (the 2010 sequel to Halo 3) players by combining gameplay logs with pyschometrics and a social network constructed from survey data [MC13]. They find that gamers preferred to play with friends, that teaming up with friends increased performance, and that social ties fundamentally affect the environment of competition.

A major difference with our work is the use of surveys vs. an OSN as ground truth for a friendship existing. Considered together, our work and theirs, particularly Mason and Clauset's results that playing together is a useful proxy for friendship and our results that declared friends have orders of magnitude more interactions than non-declared friends can be taken as evidence that declared relationships in gaming related OSNs might very well represent real "friendships."

There is an additional subtle, yet important difference between these works though: the mechanism for finding play partners. In Halo, the primary mechanism is a skill-based match-making service [HMG07], which places groups of players of similar skill into a peer-to-peer gaming session. To play with the same teammates, players must explicitly choose to "party up", and anecdotal evidence suggests that most players back out of the party up option after games with random players. In contrast, the mechanism in TF2 relies on players explicitly choosing a particular community owned and operated server, each with their own unique personalities and atmospheres, for play. This makes the selection of a virtual environment an analogue to the selection of a real world environment. For example, the frequenting of a particular pool hall, chosen not just for the competition but also for the camaraderie exhibited by the community. This easily accessible metaphor hints at the applicability of our results to real world scenarios.

Shen and Iosup [SI11] analyzed long term traces of the Xfire, a gaming network similar to Steam Community. They propose an observational study framework where repeated samplings of data are used to form a more complete picture of gamers' activities. Focusing on the characteristics of gamers, they investigated the time players spend in game, the number of games they play, the production and consumption of user generated content, and the social structure of gamers. The most notable differences with our study are our

characterization of cheaters in the gaming community and the data collection technique (sampling in [SI11] vs. exhaustive crawl in our study, which enables more in-depth social network analysis). One notable difference between their study and ours is the size of the datasets. They used a bootstrap method that identified about 65,000 users, later repeatedly sampling these users and a random sampling of their friends for a total of about 545,000 users.

In contrast, as explained in Section 4.1, we captured what we believe to be the vast majority of Steam Community users (over 10 million) at the time of our crawl via a distributed breadth first search, allowing us to make stronger claims about the social network of gamers, our primary interest. Although an early entry in the genre of gaming services, Xfire differs from Steam Community in several important ways. Steam Community is a piece of the larger Steam platform under the control of Valve. In particular, we note the vertical integration of the Steam store, Steam content delivery system, Steam Community, and Steamworks. Because of this tight integration, many newer games *require* a Steam account to play and thus many more active players.

While Xfire averages about 150,000 and peaks at 165,000 online users, Steam Community has between 3 and 6 million online users at any given time [Val12b]. Unsurprisingly, the integrations of these offerings has catapulted Steam (and Steam Community) into the forefront of gaming services, while Xfire has lagged behind. Next, Xfire has a maximum limit of 1,000 friends, which is quite a bit larger than Steam Community's limit of 250. Shen and Iousup discovered a power-law like degree distribution in Xfire, unlike what we find in Steam Community (Chapter 5), and the much higher limit on friends might be the reason for this difference. Finally, and most important for this work, Steam Community makes the cheating flag assigned by the VAC service publicly visible.

2.3 Cheating and Unethical Behavior

Cheating and unethical behavior has been studied mostly in psychology and sociology. In this section, we look at previous research on cheating and unethical behavior in two categories: 1) cheating in general, and 2) cheating in video games specifically.

2.3.1 Cheating in General

Gino et al. performed a controlled lab experiment to test the effects of "one bad apple" with respect to cheating. The experiments made use of several groups of university student volunteers. In all experiments, students were given a math test to take. In certain setups, students self reported the number of questions they successfully answered. For certain groups a shredder was made available to destroy their copy of the test. The researchers then further introduced an actor to certain groups. This actor made it very clear that he was cheating on the test by turning it in almost immediately and shredding his original work. I.e., these groups were exposed to cheating behavior.

The diffusion of (un)ethical behavior in social networks has been a topic of interest in business and management in particular. Kulik et al. proposed a theory that competitive environments can act as a catalyst for the spread of unethical behavior, using Enron as a case study [KOS08]. Enron's internal corporate structure was built around the concept of "stacking", where employees would be ranked on their performance and split along percentiles into different groups. Those in the top percentile group would receive commensurate benefits, while those in the lowest 15% were placed into an isolated area together. Those in the lower 15% were fired within a few weeks if they were unable to find a new job

within Enron, which was a near certainty due to their publicly known poor performance. Ultimately, employees could be divided into two groups: winners and losers. This intense competition lead some employees to cheat: e.g., inflating the projected profit of a proposal. An honest employee could not compete, became a loser, and was summarily fired. After observing the gains made by cheating co-workers, Kulik, O'Fallon, and Salimanh argue the behavior would be emulated by otherwise ethical employees.

Kedia et al. [KKR10] studied contagion in corporate misconduct via the manipulation of earnings statements. Their primary findings indicate that the likelihood of cheating on a report increased as the fraction of employees who had previously cheated increased. In particular, they a significantly higher chance of "aggressive accounting policies" if the fraction of cheaters in a given industry or local contacts that had cheated is higher. In other words, their findings support the theory of unethical behavior as a contagion.

As mentioned in Chapter 1, Kulik et al. [RK09] and Carrel et al. [CMW08] studied cheating in academic environments. Rettinger et al. studied cheating in US military academies. Both of these studies were survey based. In each case, they found evidence of contagion in academic cheating.

2.3.2 Cheating in Online Games

Although we do not study any cheat in particular, it is important to understand how different forms of cheating can be categorized and classified. Yan and Randell develop taxonomy for cheating across 3 dimensions: 1) what the cheat exploits, 2) what cheaters gain, and 3) who is cheating. They identify 15 common forms of cheating. Each category is well defined and comprehensive, but not disjoint; a given cheat might fall into more than one

category. While they concede that such a disjoint listing would be of great benefit, the creation of one has so far proven difficult. For example, consider a match making queue where players joining are not allowed to be in a party. Now imagine that two players can exploit such a system by joining the queue at the same time to guarantee their placement on the same team[1]. The cheaters in this case have a large tactical advantage over the other team due to, e.g., real-time voice communication over Skype. This could be placed in both the "cheating by collusion" and "cheating by abusing the game procedure". It could also arguably fall into the "timing cheating" category.

They divide what the cheat exploits into two categories, either a system design inadequacy or the people that play and operate the game. System design inadequacies are further divided into the game system and the underlying system (code). The subcategories of consequences of cheating are based on traditional computer security classifications. Confidentiality is not leaking information to unauthorized users. Integrity is ensuring that information is altered by an unauthorized party. Availability, or making sure that information is not withheld by unauthorized parties. Finally, authenticity provides identity assurances not reliant on the machine a user connects from.

Wu and Chen present a social cognitive theory based model for cheating in online games [WC13]. Social-cognitive theory is a framework for understanding human behavior that treats human behavior as as a function of both self and outside influences. In other words, social environment and personal beliefs interact to produce a given behavior. They devise 10 hypotheses on the relationships between social and personal influence, for example, that the more often people cheat around an individual, the more said individual cheats.

[1]This is a scenario that has appeared in the wild.

They first performed a preliminary study of 63 gamers to inform the creation of a survey. The resulting survey was composed of 27 Likert-scale questions across 5 theorized constructs: game cheating, social effects, cheating self-efficacy, cheating evaluation, and attitudes towards cheating behavior. They then build a structured equation model based on data collected from 1,666 respondents.

A structured equation model can be viewed as a graph, where nodes are measured variables and edges between them represent unmeasured (yet hypothesized) variables. The model they fit showed that social effects had a significant effect on cheating behavior, as well as the valuation of cheating behavior. Additionally, the valuation of cheating had a significant effect on attitude towards cheating, which itself had a significant effect on cheating behavior itself. In other words, the more an individual is exposed to, the more likely they are to cheat, the more likely they are to find value in cheating, and the more their general attitude towards cheating will shift.

While there has been a decent amount of attention on technical aspects of cheating, primarily in the detection and prevention of 3rd party cheat software [MDF11, KTCB05, BCR11], the social aspects of cheating in games remains relatively unexplored.

Dumitrica [Dum11] examines *Neopets*, a web based social game. She describes a process by which gamers, who naturally seek ways to increase their "gaming capital", are tempted to cheat, and argues that a cheating culture emerges, where social values are used to understand and evaluate the ethical questions of cheating.

Nazir et al. study fake profiles created to gain an advantage in [NRCS10]. Though the evaluation of behavior of player accounts within the social game Fighters' Club (FC), they are able to predict with high accuracy whether a profile is fake. Users in FC cheat by performing what is essentially a Sybil attack, since a player's power in FC is directly

proportional to the number of declared friends that are also players. Cheaters in Steam Community, however, do not explicitly gain an advantage by altering the structure of the social graph. Instead, they are attacking out of band game rule implementations.

Vazquez and Consalvo performed a survey based study on the perception and practices of cheating in social network games similar to FC [VC13]. They answer three research questions: 1) *how* players cheat in social network games, 2) what cheating related practices they partake in, and 3) how does the perception of cheating differ between individuals and across genres and gaming platforms. They find a variety of mechanisms by which players cheat in social network games, ranging from account sharing to spaming their friends lists. They also note that their respondents differed as to their opinions on what behavior constitutes cheating. Finally, they determined that there was a difference in cheating behavior, and particularly how it is perceived, depending on the genre or platform of the game. The study's results show a different alignment of moral compasses with respect to cheating in social network games vs. multiplayer games on consoles or PC: about 35% of respondents said that cheating across these different platforms was "different." This finding is particularly relevant to the current work.

Because the datasets used in this dissertation are drawn from PC gamers, the results might not be completely portable to other game platforms. From samples of open ended questions on the survey, we conjecture that many of the respondents who who viewed cheating as different might not be familiar with modern mulitplayer gaming. For example, one respondent noted that cheating on PC or console games did not hurt any other players, indicating a lack of knowledge of the popularity of multiplayer games That said, about 65% of respondents viewed the cheating behavior the same. Unfortunately, their study was

primarily qualitative and they were unable to draw many statistically significant conclusions.

"Gold farmers" are cheaters that make black-market exchanges of real world currency for virtual goods outside of sanctioned, in-game, trade mechanisms. By examining social networks constructed from database dumps of banned *EverQuest II* (EQ2) players, Keegan et al. [KAW+10] found gold farmers exhibit different connectivity and assortativity than both their intermediaries and normal players, and are similar to real-world drug trafficking networks. Ahmad et al. [AKS+11] further examined trade networks of gold farmers and propose models for deviant behavior prediction.

The dataset they used dataset differs from ours in both motivation for cheating, and the method of punishing cheaters. No clear financial motivation for cheating exists in the majority of games played by the majority of Steam Community players. Although the growing eSports industry has made gaming a possible profession, and tournaments with real world prizes are seen as a stepping stone for amateurs to go professional, most gameplay time is not spent with prize money on the line. While trafficking of virtual goods most certainly has implications for the monetization of the Free-to-Play model that has been recently adopted by many games, VAC bans apply specifically to *cheating*. Additionally, while cheaters in EQ2 have their accounts permanently disabled, cheaters in Steam Community are only restricted from playing the particular game they were caught cheating in on VAC-secured servers, as explained in Section 4.1.1.

Similar to business environments like Enron [KOS08] the overwhelming majority of multiplayer games have clear winners and losers. In multiplayer FPSs like Halo and TF2, performance is also made public, as it was in Enron. At minimum, these games tend to have an in-game scoreboard displaying performance related statistics for each player (such

as points, or the number of kills and deaths in the given gaming session), with the highest performing players at the top of the scoreboard's visual representation, and the lowest at the bottom. In addition to the in-game scoreboard, most games make world-wide and friends leaderboards, as well as more detailed life-time statistics such as accuracy, number of games won, and a variety of game-specific statistics available.

Although no one is fired for poor performance in a game, there are consequences for poor performance besides the social stigma of being a "newb". For example, the TrueSkill [HMG07], algorithm used by Halo, ranks players and places similarly skilled individuals in the same game. The rank of a player is visible in the in-game UI for Halo 3, removed from the UI of its sequel Halo: Reach (studied by Mason and Clauset [MC13]), and will eventually be made available on the stats tracking website for the latest incarnation, 2012's Halo 4 [2], but it affects matchmaking none the less. A particularly relevant finding from discoveries in [HMG07] is that the deployment of TrueSkill led to an increased frequency of cheating. This form of cheating, known as "boosting," has sprung up where players boost their statistics using various unscrupulous methods. There is even at least one service that claims to employ professional gamers to boost accounts, later selling them for around $50 a piece [Hal12]. The competitive aspects shared by gaming and business indicates they share same process of unethical behavior's spread. Thus, findings from this work can help explain how unethical behavior spread through an organization like Enron.

Finally, we note that to the best of our knowledge, our work is the largest scale study of cheaters in a gaming social network. We discovered over double the amount of cheaters as there were players in [NRCS10], and multiple orders of magnitude more cheaters than players in [KAW+10, AKS+11].

[2]The ranking system has become so important to players that developers are postponing the release until they can properly tune it, much to the chagrin of a sometimes rabid fan base hungry for a metric by which to gauge their standing in the highly competitive community.

2.4 Toxic and Anti-social Behavior

Toxic behavior in online games is a form of cyberbullying, defined as repetitive intentional behavior to harm others through electronic channels [SMC+08]. Computer mediated communication through electronic channels without face-to-face communication lacks social psychological influences and can lead to more hostile interaction [HM08]. In psychology and education, offline bullying has been actively researched [Olw96], and offers a theoretical background to understand cyberbullying.

Griefing is a term describing cyberbullying in online gaming, and those who enjoy the act of disrupting other players are called grief players [FK04, MB00]. Griefers make other players annoyed and feel fatigued. Sometimes victims even leave the game [MPK03], exhibiting toxic behavior themselves to escape beratement.

Although griefing is intuitively understood, its boundary is unclear [CCLM09, FK04, LS05, MPK03] because customs, rules, or ethics can be different across games [WR05]. In addition, the perception of grief behavior is unique across individuals. As a consequence, even griefers themselves may not recognize what they have done as griefing [LS05]. This inherent vagueness makes it hard to understand grief behavior.

A few studies have characterized grief playing. Foo and Koivisto divide grief behavior into four categories: harassment, power imposition, scamming, and greed play [FK04]. They focus on the intention of behavior and distinguish griefing from *greed playing* because motivation behind greed playing is usually for the win instead of disrupting other players. Barnett discovers that griefing is one of factors provoking anger in World of Warcraft by survey of 33 participants [BCF10]. Chen et al. correlate personality and grief playing. They reveal that players who enjoy the anonymous experience tend to like the motivation

of grief playing, such as "I like the experience of wanting to feel powerful while playing online games" [CDN09].

Chapter 3: ARKS: An Architecture for the Collection and Analysis of Social Network Data

Unlike controlled laboratory experiments, researchers wishing to conduct empirical "in the wild" studies face several unique challenges in collecting and analyzing their data:

1. Poorly documented, incomplete, or broken APIs exposed by data sources.
2. "Moving target" concerns with the data that is exposed.
3. Changing needs in both data collection and analysis as understanding of a research problem matures.
4. New sources of data becoming available over time.

To address the above challenges, we devised an architecture [BI13b] that met the following three requirements:

- Requirement 1: Decoupled architecture.

Decoupling of data collection from extraction and analysis enables our platform to be useful for studies of dynamic networks relying on volatile data sources. Most solutions employed in one-time social network analyses address data collection, extraction, and analysis as a monolithic component: the data analysis requirements inform data extraction,

inevitably leading to multiple iterations of data collection or limited knowledge extraction. While decoupling data collection from extraction and analysis makes greater demands on storage, it allows for (a) successive iterations of data extractions as informed by data analysis results with minimum costs; (b) transparent collection of newly introduced attributes embedded within the data source independent of the data extraction or processing mechanisms; and (c) independent insertion or removal of new sources of data. This loosely coupled architecture has the implicit benefits of fault tolerance and scalabilit

- Requirement 2: Versioning and provenance.

Longitudinal studies require revisiting the same data source over time to compare changes in the network. At minimum, this requires time stamping of data. More importantly, however, in light of the rapid growth experienced by social applications and the ever increasing ubiquitousness of Internet connected mobile devices, the format of data sources are likely to change over time, necessitating adaptations in data extraction code. In addition to data provenance, we are exploring the usage of existing distributed version control systems and continuous integration tools to provide knowledge of process provenance.

- Requirement 3: Multigraph, ego-net based data model.

Representing the fusion of multiple sources of social data for a single ego involves several design challenges. First, we observe that an undirected graph cannot adequately model contexts where relationships and interactions are not necessarily reciprocal in nature. Second, from a sociological perspective, even reciprocal relationships are usually not symmetric, calling for the ability to place a weight on an edge [Wel88]. Third, simple graphs are unable to capture ego's ties *across* disparate social networks, and thus are insufficient

for answering questions about behaviors that span application domains. For example, consider a study on diffusion, focusing on free form reviews given by gamers. Gameplay statics available via the Steam Community application, comments from YouTube, and participation in relevant communities such as `gaming.reddit.com` can be combined to form a more complete view of influence and exposure.

Finally, the abstraction of a multigraph stored as ego-nets allows for holes in the observational record to be patched in certain cases. Consider a situation where a change in state triggers the crawling of a user (see: Section 3.4.1). In this scenario, information about the newly crawled user might be inferable from information collected on previous users, and vice versa, *without altering the provenance record*.

For example, say users A and B were crawled at times t_i and t_{i+n}, respectively. In A's record no relationship between the two exists, yet in B's one does. We can thus *infer* (at least partially) A's neighborhood at t_{i+n} without actually collecting any additional information. We can also infer information about B's neighborhood prior to having made any direct observations. Most importantly, the inference maintains the observational record; storing the graph as, e.g., an edge list, would eliminate the distinction between inference and direct observation, especially considering that the overwhelming majority of crawled datasets are not direct observations of the graph, but rather multiple observations of separate ego-nets.

A high level overview of our architecture can be seen in Figure 3.1. The scheduler (labeled 1) is responsible for issuing requests to crawl a particular user. The crawlers (2) receive the requests issued by the scheduler and retrieve raw data from various data sources. The output from the data collection sub-system is received by the storage process (3), which coalesces crawler output, for example the output from the Steam crawler and the YouTube

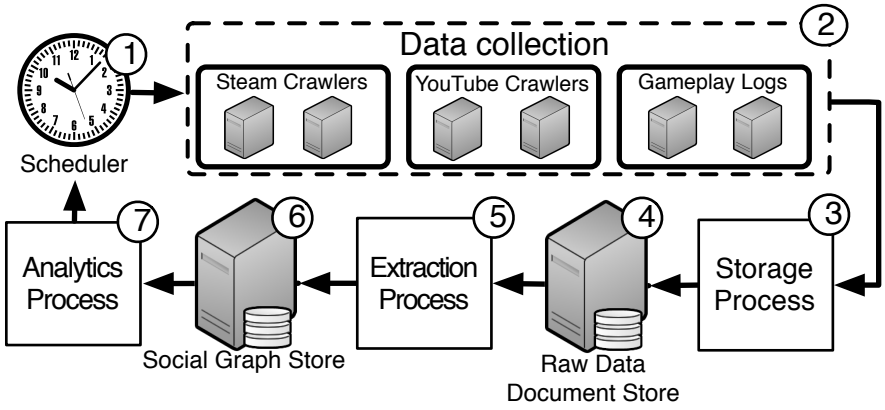

Figure 3.1: A high level view of our architecture.

crawler for a given user, and generates provenance metadata. Once coalesced, the storage process commits the raw data to the raw data document store (4). The extraction process (5) extracts meaningful social information, and stores it, along with additional provenance metadata, in the social graph store (6). Finally, the analytics process (7) operates on the structure exposed by the social graph.

With the above in mind, the remainder of this chapter describes an implementation of this architecture that meets the needs of social data scientists.

3.1 Scheduler

The scheduler is responsible for issuing crawl requests to crawlers. It separates deciding *what* to crawl from the actual process of crawling. This makes it particularly easy to change crawling behavior due to both research and technical requirements. For example, the scheduler can respect rate limiting by controlling the requests it makes to crawlers. From a more research oriented point of view, the scheduler allows for dynamic decisions on who to crawl, and how often to crawl.

The only requirements the architecture exposes on the scheduler is the ability to issue requests to crawlers and the ability to update schedules. Our scheduler implementation is written in Ruby and makes use of the rufus-scheduler gem to achieve a cron-like interface.

3.2 Data Collection

The initial implementation of the data collection component was used to crawl Steam Community (Section 4.1) between March 16th and April 3rd, 2011. Although there was a fledgling API exposed, it did not allow for querying of friends lists at the time. Thus, we used unmetered, consumable XML on the `steamcommunity.com` web site. The crawler collected user profiles starting from a randomly generated set of SteamIDs and following the friendship relationships declared in user profiles. To seed our crawler, we generated 100,000 random SteamIDs within the key space (64-bit identifiers with a common prefix that reduced the ID space to less than 10^9 possible IDs), of which 6,445 matched configured profiles.

The crawling was executed via a distributed breadth first search, and the implementation was built around Amazon Web Services. Each of the initial seed SteamIDs was pushed onto an Amazon Simple Queue Service (SQS) queue. Each crawler process popped one SteamID off this queue and retrieved the corresponding profile data via a modified version of the Steam Condenser library. The profile data of the crawled user was stored in a database and any newly discovered users (i.e., friends that were previously unseen) were added to the SQS queue. Crawling proceeded until there were no items remaining in the queue. Using RightScale, Inc's cloud computing management platform to automatically scale the crawl according to the number of items in the SQS queue, we ended up running

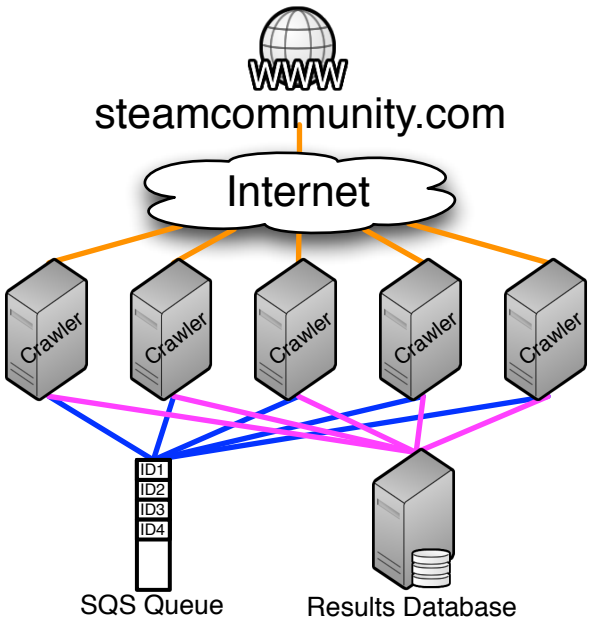

Figure 3.2: A high level overview of the crawler used to collect the static Steam Community data set.

up to six Amazon "c1.medium" EC2 instances executing up to 15 crawler processes each. A high level overview of the crawler implementation can be seen in Figure 3.2.

We built a second implementation that took full advantage of the scheduler to dynamically gather observations on over 9 million users every day. This implementation includes two data collection subsystems: 1) a status flag observer, and 2) a social neighborhood observer.

The status observer is scheduled to run for every known user in the system once per day. The scheduler creates one request per 100 known users per day, which is then placed onto a job queue. Using the `Resque` job queuing library, one of a number of distributed crawlers executes the job. Accessing the 3rd party data takes place over HTTP. The results from the 3rd party data source are then placed into another job queue for processing.

The second crawler retrieves friends lists for selected users. There is one particular domain specific concern making this crawler a bit more complicated. The issue is that the Web API is limited to 100,000 calls per day. Luckily, the ban status API allows for the querying of 100 users' ban statuses in a single call. Although our dataset grew slowly as friends were added from monitored users' friends lists, we were still left with several thousand unused API calls per day.

Unfortunately, the friends list API only returns a single user's friends list. To limit our calls, only the initial query of a user's neighborhood made use of the API and subsequent queries make use of the unmetered XML interface to Steam Community. A consequence of this is that the resolution of the `friends_since` field is reduced to 24 hours on all but the first observation of a user's neighborhood.

3.3 Data Storage

There has been some exploration of various database solutions suitable for social data storage [RBR11], each with their own pros and cons. For our purposes, we settled on MongoDB [1]. The choice of MongoDB was motivated by several factors:

1. Easy to understand ad-hoc data modeling.
2. Familiar SQL-esque querying and RDBMS-style indexing.
3. Promises of high availability and performant scaling.

[1] `http://www.mongodb.org`

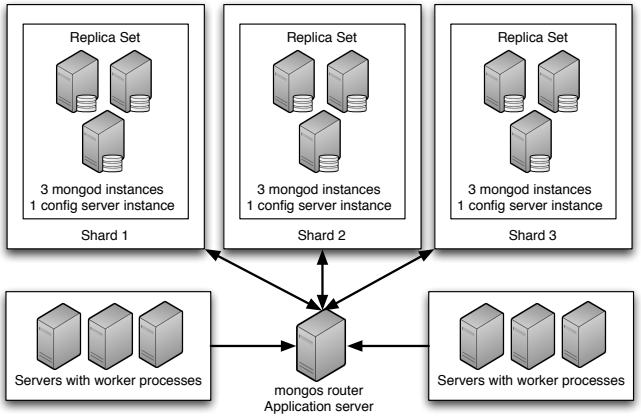

Figure 3.3: An overview of the MongoDB cluster for our collection system.

MongoDB is a NoSQL [HHLD11] document store. Generally speaking, the NoSQL name is a bit of a misnomer. SQL is the query language typically used by Relational Database Management Systems (RDBMs). RDBMs tend to be build around the concept of tables and rows with data normalization. For example, there might be a customers table that has columns for a customers first name and last name, as well as a link to another table that stores order information.

NoSQL document stores like MongoDB, on the other hand, tend to avoid normalization and instead favor an approach where a single document (e.g., a customer) contains all information relevant for that document (e.g., the customer's orders). There are several advantages to this type of scheme, but, for our purposes, a document store makes it easy to respond to changes in source data due to its schemaless nature. In other words, we do not have to normalize our records with defined tables and columns, but rather the structure of the record is contained within the document itself.

From a performance standpoint, the benefit to this has to do with the distribution of data throughout a multi-machine cluster. A document store like MongoDB has implicit data locality. It also avoids difficulties with distributing normalized data.

```
1  neighborhood_observation = {
2    user_id:string,
3    observed_at:timestamp,
4    plus:[friend_observations] <optional>,
5    minus:[friend_observations] <optional>
6  }
7  friend_observation = {
8    friend_id:string,
9    relationship:string,
10   friend_since:timestamp <optional>
11 }
```

Figure 3.4: JSON datastructure for storing neighborhood observations.

There were some issues related to the specific domain that are not addressed in the architecture described earlier. First, the volume of data we were collecting hinted at the need for *delta based storage*. Even though the per-user data returned for each ban status query was small, in aggregate it added up fast, and so we implemented a simple delta storage scheme that found the last known value of each flag and stored the new value if it had changed [2]. A side effect of this was that any additional ban types added to the API were automatically integrated into our system [3].

The same basic approach was used for the storing of neighborhoods, but a more complex data structure was required, as seen in Figure 3.4.

The `user_id` field is a key representing the user this observation corresponds to, and the `observed_at` field is the time the observation was taken. The `plus` and `minus` fields contain the deltas computed from neighborhood observation, with the `plus` fields storing the edges that were added to this neighborhood, and the `minus` field those edges that were removed since the previous observation. `friend_observation`s are edges, and labeled with

[2] A timestamp recording that an observation was made is always stored.
[3] Two additional ban types, "community" and "trade", were discovered.

the type of a relationship (at the time of this writing there is only one type of edge on Steam, "friendship"), and the `friend_since` field is the timestamp that the friendship was created.

Because MongoDB uses JSON as a record format, fields that were not used in an observation (e.g., if the user's neighborhood had no change) were not stored. The `friend_since` field in the friend_observation is optional for two reasons: 1) when someone is "unfriended", there was no need to store the date the relationship was created; we already had it on record in a previous observation, and 2) some friendships do not have a `friend_since` value available from the API [4].

3.4 Analytics

While collecting and storing data is necessary for studying empirical social data, researchers must perform actual analysis to produce results. In addition to analysis oriented towards end results, i.e., the findings we present in later chapters, the data collection process itself is informed by the analysis process. In this section, we present the reference implementation of the analysis tools used in this dissertation.

3.4.1 Finding New Users to Crawl

A major piece of the functionality in this implementation is the dynamic monitoring of neighborhoods for newly marked cheaters. To this end, after ban status's have been crawled, they are analyzed for transitions from non-cheater to cheater. We say a user has transi-

[4] Steam Community did not record the date of friendship creation until 2009.

tioned when we have at least one observation for him, he has been marked as a cheater in the current observation, and he was a non-cheater in the previous observation.

When we detect a transitioned user, the analysis component schedules several new crawler jobs. First, an immediate crawl that makes use of the Steam WebAPI to get a timestamped list of relationships. Next, one neighborhood crawl is scheduled for the next 10 days. Every time a neighborhood is crawled, any newly discovered users are added to the list of users for daily ban status checks. Note that the transition detection described above ensures that we are not performing neighborhood scans for users with undetermined ban dates (i.e., users that we had not previously discovered, but were banned some time in the past).

3.4.2 Ad-hoc Programs

Many of the results in this work were generated using ad-hoc analysis programs. Generally speaking, these programs simply made use of the data that had been collected by the implementation, without providing any feedback to the scheduler. Analysis programs were written in Ruby, Python, and R. Had we required "real-time" updates to our results, we could have simply scheduled the ad-hoc programs to run on a regular basis [5].

[5] We did implement a few scheduled programs to monitor the state of the data collected, for example the number of cheaters detected per day

3.4.3 Elaine

Elaine is an implementation of the Pregel system proposed in [MAB+10]. Pregel implements the Bulk Synchronous Processing (BSP) model [Val90]. The BSP model is composed of distributed computation nodes which compute in parallel, a communication component to allow the processes to communicate, and a synchronization barrier, which suspends computation until some condition has been met.

Pregel is a realization of the BSP model for performing distributed computation on (large) graphs. Each vertex in the graph keeps track of its out edges (i.e., edges pointing to other vertices) and a value. Additionally, each vertex implements a function that performs the actual computation. Results are built up iteratively via repeated calls to the compute function. Vertices send and receive messages to each other such that a message sent at super step s is available in super step (and only super step) $s+1$. Because the programmer is primarily concerned with the computation that occurs at the vertex level, Pregel programs are also referred to as *vertex programs*.

A *coordinator* node in a Pregel cluster is responsible for any initialization, partitioning, and distribution of the graph. The coordinator distributes portions of the graph (as determined by a partitioner) to a set of distributed *worker* nodes. It also is responsible for the synchronization of super steps between workers. Worker nodes are in turn responsible for executing the computation function on the vertices they are responsible for.

Figure 3.5 shows a state machine view of the Pregel computation model. Each vertex can be in one of two states: active or inactive. If the vertex is active at step s, then it will perform computation. Vertex states can be handled in a domain specific way, but, generally

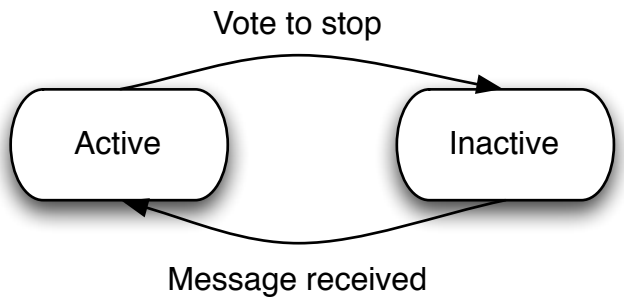

Figure 3.5: An overview of the Pregel computation model.

speaking, a vertex is active if it has not voted to stop or it has messages available at step s.

Elaine is an implementation of Pregel written in Ruby, and relies on the DCell actor-based distributed objects library for compute node communications. While Google's Pregel implementation is not publicly available, there are some public implementations [KAA+13]. We chose to create our own implementation for several reasons. First and foremost, we wished to expand our understanding of the challenges involved in a Pregel implementation. Next, we had particular interest in exploring the suitability of Ruby for large scale data processing. Finally, at the time of Elaine's inception, there was no Ruby implementation of a distributed Pregel, and thus we have the potential to provide a benefit to the Ruby ecosystem.

The heart of Elaine is in the exposure of three objects to remote processes, and the definition of a vertex program. The three objects, or services, are `Coordinator`, `Worker`, and `Postoffice`. With Elaine, we use the terminology *compute node* to represent a unique end point that an Elaine services reside. Any given compute node can be running one or more services, however, the assumption is that one compute node will expose the coordinator service while the remaining compute nodes each expose a worker and postoffice. It is im-

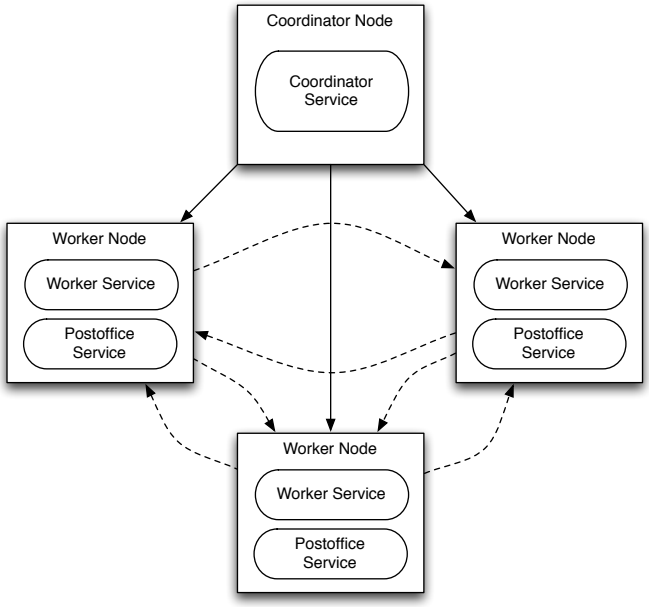

Figure 3.6: An overview of Elaine. The coordinator node controls the flow of execution. Worker nodes perform computation via the worker service. Messages are sent between to individual vertices via postoffice service (dashed lines).

portant to note that a compute node is a logical abstraction: a single machine can host multiple compute nodes, with each compute node running as its own process.

Within the compute nodes, services are managed as threads per the DCell library. The coordinator service is responsible for initializing a job, which includes partitioning and distributing a graph. It also controls the flow of a given program by instructing each worker to perform a super step. Each worker service is responsible for a given partition of the graph. It is responsible for invoking computation on each vertex in its partition. The post office service enables message passing between remote vertices. A high level overview of the Elaine architecture can be seen in Figure 3.6.

3.4.3.1 The Coordinator

As mentioned above, the coordinator is responsible for managing workers and any setup the vertex program, in addition to any setup that is necessary for the vertex program. To this end, Elaine exposes an API that provides some default functionality. The API is as follows:

- `register_worker` - Registers worker services.
- `graph=` - Allows for setting the graph of the coordinator.
- `zipcodes` - Generates a zipcodes hash that maps vertices to the worker they reside on.
- `distribute_graph` - Distributes the graph to worker nodes.
- `partition` - Partitions the graph.
- `init_job` - Initializes the job by partitioning the graph, generating zipcodes and distributing them to postoffices, and distributing the graph to worker nodes.
- `init_superstep` - Performs initialization necessary at the start of a super step.
- `superstep` - Performs a super step on each worker service in parallel.
- `vertex_values` - Gathers all vertex values from worker services.
- `aggregate` - Aggregates results from each worker service.

This functionality can be overriden by a programmer for any special needs.

3.4.3.2 Workers

The worker service is responsible for performing computation on any active vertices in its portion of the graph. The API exposed is:

- `add_vertex` - Adds a vertex to the portion of the graph this worker is responsible for.
- `add_vertices` - Adds multiple vertices to the portion of the graph this worker is responsible for.
- `init_graph` - Initializes an entier graph this worker is responsible for.
- `init_superstep` - Performs initialization necessary at the beginning of each super step.
- `superstep` - Executes a step for each active vertex in its portion of the graph, in parallel.
- `vertex_values` - Gathers and returns the values of each vertex in its portion of the graph.
- `aggregate` - Performs an aggregation function on the vertex values in its portion of the graph. Note: aggregation is a domain specific concern and thus programmers must supply an aggregate function if one is desired.

3.4.3.3 Postoffice

The post office handles message delivery to and receiving from remote computation nodes. Recall that vertices pass messages to each other. This message passing occurs asynchronously. I.e., a vertex does not wait for s sent message to be received by the destination before continuing computation. The post office enables this by essentially buffering all outgoing messages and then sending them in bulk to the post offices responsible for the destination vertices.

The post office exposes the following API:

- `zipcodes=` - Allows for setting the zipcodes mapping vertices to remote workers.
- `deliver` - Delivers a message from one vertex to another.
- `deliver_all` - Delivers all messages currently in this postoffice's outbox.
- `deliver_bulk` - Delivers bulk messages to remote postoffices.
- `receive_bulk` - Receives bulk messages from remote postoffices.
- `read_bulk` - Moves messages from vertex mailboxes to the vertex itself.
- `address` - Returns the compute node that a given vertex resides on.

There are some performance related decisions in the post office implementation. First, the aforementioned message buffering reduces the overall communication between computation nodes and enables the asychronous computation performed by vertices. Next, once the buffer limit is reached, vertex messages are compressed (by default using Zlib compression) and then sent across the wire in bulk. Finally, if the destination vertex resides on the same

compute node as the source, the post office will immediately place the message in the destination vertexes inbox for the next superstep.

3.4.3.4 Vertex Programs

Programmers define the actual computation process by deriving from the `Vertex` class. This class provides an API as well as some default functionality as follows:

- `deliver` - Delivers a message to another vertex via the postoffice service.
- `deliver_to_all_neighbors` - Delivers a message to all neighbors of this vertex.
- `compute` - Perform the computation for a step.
- `vote_to_stop` - Sets this vertex state to inactive.

As a simple example, consider the *single source shortest path* problem. The SSSP problem finds the length of the shortest path from a source vertex to every other reachable vertex in the graph. Consider the binary tree that appears in Figure 3.7. If we run SSSP with the source as vertex 0, the length of the shortest path of any other vertex is imply the level that vertex appears at. While it is easy to label the vertices in a binary tree in this way, Figure 3.8 provides a vertex program to solve the problem for any graph.

There are several relevant lines to understanding how a vertex program works. In line 5 from the constructor, all vertices besides the source vote to stop (become inactive). Although this is not entirely necessary due to the implementation of the `compute` method, it aids in clarity: we know that only the source vertex's `compute` will be run at superstep 1. In line 10, the vertex extracts the minimum value from any messages it was sent at the

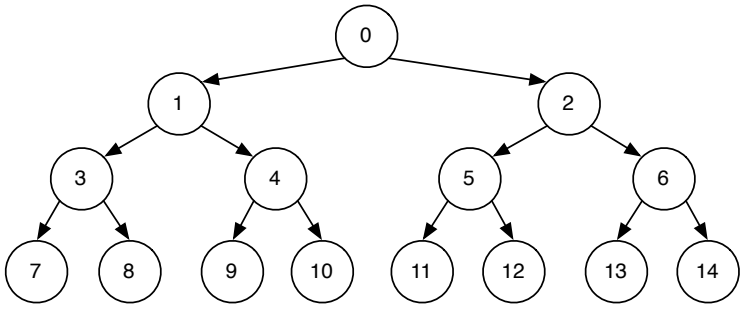

Figure 3.7: A binary tree.

```
1  class SSSPVertex < Elaine::Distributed::Vertex
2    def initialize(*args, graph_size: nil)
3      super(*args)
4      @value ||= {min_dist: Float::INFINITY}
5      vote_to_stop unless @value[:source]
6    end
7
8    def compute
9      min_dist = @value[:source] ? 0 : Float::INFINITY
10     min_dist = @messages.size > 0 ? @messages.min :
         min_dist
11
12     if @value[:min_dist].nil? || min_dist < @value[:
         min_dist]
13       @value[:min_dist] = min_dist
14       deliver_to_all_neighbors(@value[:min_dist] + 1)
15     end
16
17     vote_to_stop
18   end
19 end
```

Figure 3.8: A Pregel style vertex program for the single source shortest path problem.

previous superstep. In lines 13 and 14, if necessary, it updates its value (the minimum distance between itself and the source) and sends what amounts to its best guess at the minimum distance between each of its neighbors and the source vertex. Finally, in line 17 the vertex votes to stop.

3.4.3.5 Performance Analysis

Elaine is designed to scale to arbitrarily large graphs performing arbitrary computations. Because of this, it is somewhat difficult to measure Elaine's performance. However, since Elaine essentially provides the programmer with a message passing framework, it makes sense to examine how the number of messages sent affects running time.

The binary tree SSSP example (Listing 3.8) works as a good test case. The computational requirements are minimal, and as the number of messages in each superstep doubles, it provides an opportunity to observe message cascades.

Figure 3.9 plots the number of workers versus the cumulative superstep runtime for a SSSP on a binary tree of 1 million vertices. I.e., the running time of the vertex program, not counting graph initialization, worker provisioning, etc. Performance increases proportional to the number of workers, up until around 9 or 10 where it flattens. In other words, Elaine scales well.

There are a few caveats to note, however. First is that although we exhibit the same scaling properties as the original Pregel implementation (see Figures 7 and 8 in [MAB+10]), we believe the overall performance of our system is not as good. Pregel is written in C++ while Elaine is written in Ruby. Ruby is a much higher level language than C++, and

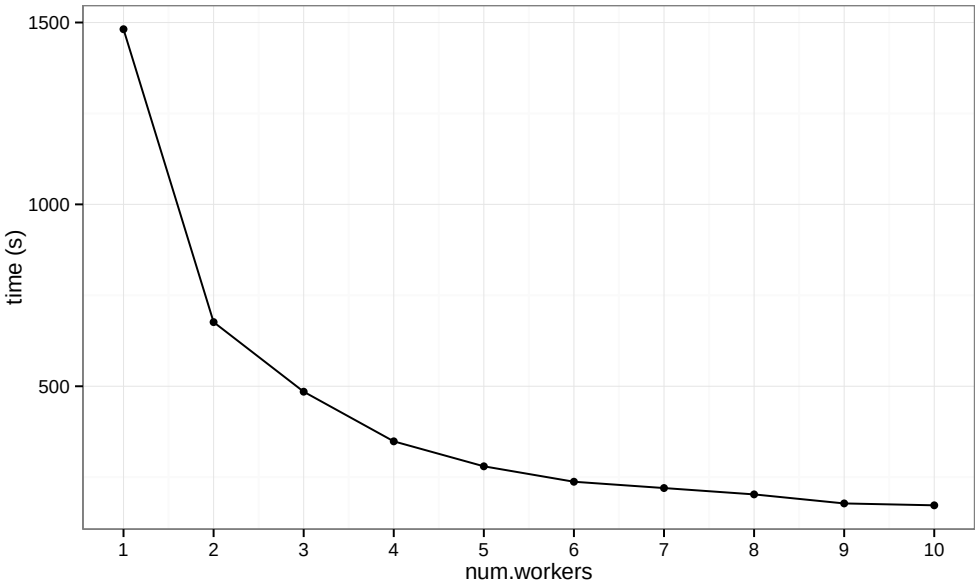

Figure 3.9: Total running time for the SSSP problem on a binary tree of 1 million vertices executed on varying amount of workers.

worse performance is expected. Additionally, we ran our tests using the JRuby virtual machine (VM) to take advantage of our multi-core cluster of machines. JRuby is, generally speaking, not as fast as the reference Ruby VM written in C, however, it takes advantage of hardware threads, which the standard VM does not. Because Elaine uses threads, JRuby is able to take full advantage of our multi-core machine cluster.

Figure 3.10 plots the individual running times of each superstep million vertex graph executed on varying number of workers. This plot makes the cascading aspect of the binary SSSP problem clear. Recall that the (maximum) number of messages sent at superstep s is 2^s. In other words, the number of messages sent doubles for each superstep. For the first 10 supersteps, the performance is approximately the same, regardless of how many workers are used. This is because there are very few messages to be sent. In total, there are $\sum_{s=1}^{10} 2^s = 2046$ messages sent through the first 10 steps. At step 11, there are $2^{11} = 2048$ messages sent. It is not until this point that message passing begins to exceed the

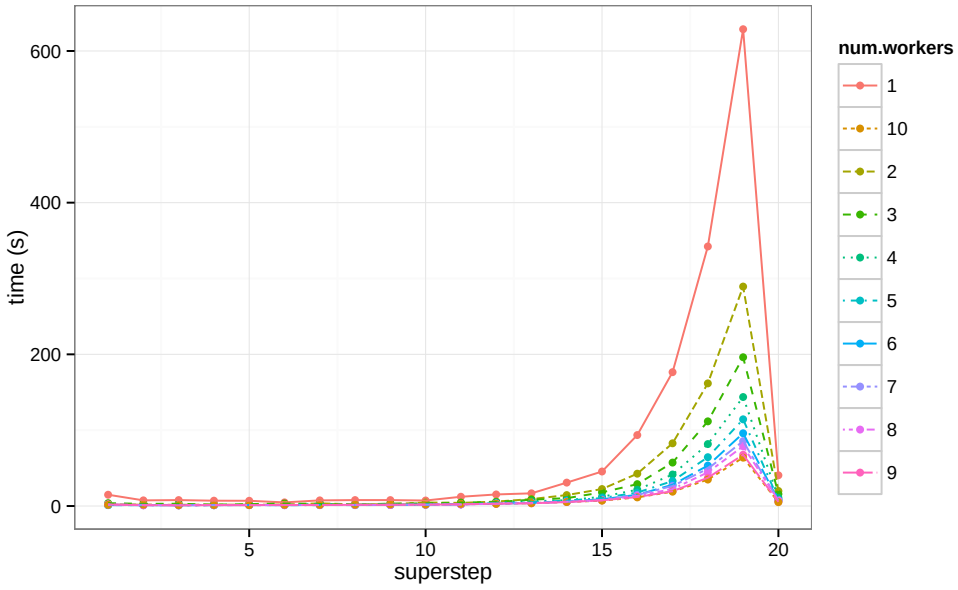

Figure 3.10: Running time per superstep for the SSSP problem on a binary tree of 1 million vertices executed on varying amount of workers.

overhead of initializing the superstep. As in Figure 3.9, we see that once we hit this point, individual superstep performance is proportional to the number of workers used. The drop off from step 19 to step 20 is due to a binary tree of 1 million vertices not being full (i.e., the last level does not have as many vertices as it could).

Next, to measure the scalability in terms of graph size, we run the SSSP on binary trees ranging in size from 1 to 10 million vertices on 10 workers. Figure 3.11 plots the results. The running time increases linearly, thus showing that Elaine scales not only in terms of workers, but also the size of the graph.

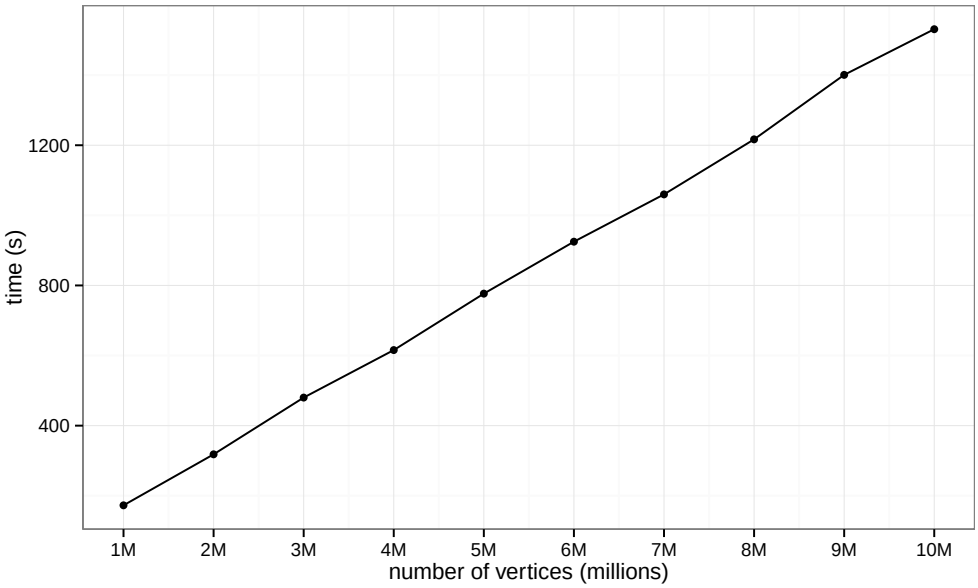

Figure 3.11: Total running time for SSSP on binary trees of varying size executed on 10 workers.

3.5 Summary

In this chapter we presented an architecture for the collection of longitudinal social data. We described the seven components of the architecture and the reasoning behind them. We then presented an implementation of this architecture. We demonstrated custom crawlers and a storage solution using MongoDB. We also discussed dynamic analysis that feeds into the scheduler, as well as a system for large scale graph analysis called Elaine. Finally, we presented measurements showing that Elaine scales both in the number of distributed workers as well as the size of the input graph.

Chapter 4: Datasets

[1]There are three major datasets used in this dissertation. The first is a planetary scale Online Social Network of gamers. The second is about 10 months worth of detailed in-game interactions from a community owned and operated game server located in California. The third dataset is crowdsourced decisions regarding toxic behavior in an online video game.

4.1 The Steam Community Online Social Network

Steam controls between 50% and 70% of the PC game digital download market [Sen11], and claimed over 40 million user accounts as of September 2012 [Val12a]. Steam is run by Valve, who also develops some of the most successful multiplayer first-person shooter (FPS) games.

For gamers, Steam provides a system to purchase, download, and play games. For game developers, Steam provides a content distribution network, storefront with a digital rights

[1]Much of the work in this chapter was first published in Proceedings of the 21st International Conference on World Wide Web, 2012 [BSK+12], ACM Transactions on Internet Technology, 2014 [BKS+14] and Proceedings of the 23rd International Conference on World Wide Web, 2014 [BK14]. Permission is included in Appendix B.

management solution, and a tightly integrated API for cloud storage, domain specific features like achievements, and a social service called Steam Community.

Games from a number of developers and publishers are available for purchase on Steam. A noteworthy segment is formed by the *multiplayer* FPS genre. In contrast to massively multiplayer online games, multiplayer FPSs usually take place in a relatively "small" environment, player actions generally do not affect the environment between sessions, and instead of one logical game world under the control of a single entity, there are often multiple individually-owned and operated servers. Because there is no central entity controlling game play and a large number of servers to choose from, the communities that form around individual servers are essential to the prolonged health of a particular game.

An initial dataset was collected by crawling the Steam Community website for user profiles and the social network represented by declared relationships between them. After a first round of analysis, we used the system described in Chapter 3 to make daily observations of the ban status of a large set of users (over 9 million), as well as daily observations of the neighborhoods of newly banned cheaters and a set of control users.

Steam Community is a social network comprised of Steam users, i.e., people who buy and play games on Steam. To have a Steam Community profile, one first needs to have a Steam account and take the additional step of configuring a profile. Users with a Steam account and no profile (and thus, not part of the Steam Community) can participate in all gaming activities, and can be befriended by other Steam users, but no direct information is available about them. Steam profiles are accessible in game via the Steam desktop and mobile clients, and are also available in a traditional web based format at http://steamcommunity.com.

A Steam profile includes a nickname, a privacy setting (public, private, friends only or in-game only), set of friends (identified by *SteamIDs*), group memberships, list of games

Table 4.1: Size of the *Static Steam Community* dataset.

Account	*All*	*Edges*	*Profiles*	*Public*	*Private*	*Friends only*	*With location*
All users	12,479,765	88,557,725	10,191,296	9,025,656	313,710	851,930	4,681,829
Cheaters	-	-	720,469	628,025	46,270	46,174	312,354

owned, gameplay statistics for the past two weeks, a user-selected geographical location, and a flag (VAC-ban, explained in more detail in the next section) that indicates whether the corresponding user has been algorithmically found cheating. We augmented the information for the VAC-banned players with a timestamp that signifies when the VAC ban was first observed (as explained next).

The initial implementation of the data collection component of the architecture described in Chapter 3 was used to crawl Steam Community between March 16th and April 3rd, 2011. Although there was a fledgling API exposed, it did not allow for querying of friends lists at the time. Thus, we used unmetered, consumable XML on the steamcommunity.com web site. The crawler collected user profiles starting from a randomly generated set of SteamIDs and following the friendship relationships declared in user profiles. To seed our crawler, we generated 100,000 random SteamIDs within the key space (64-bit identifiers with a common prefix that reduced the ID space to less than 10^9 possible IDs), of which 6,445 matched configured profiles.

From our initial $6,445$ seeds of user IDs, we discovered just about 12.5 million user accounts, of which 10.2 million had a profile configured (about 9 million public, 313 thousand private, and 852 thousand visible to friends only). There are 88.5 million undirected *friendship* edges and 1.5 million user-created groups. Of the users with public profiles, 4.7 million had a location set (one of 33,333 pre-defined locations), 3.2 million users with public profiles played at least one game in the two weeks prior to our crawl, and 720 thousand users are flagged as cheaters. Table 4.1 gives the exact numbers.

While Steam accounts are free to create, they are severely restricted until associated with a verifiable identity, for example from game purchases (via a credit card) or from a gift from a verified account. *Once associated with an account, game licenses (whether bought or received as a gift) are non-transferable.* This serves as a disincentive for users to abandon flagged accounts for new ones: abandoning an account means abandoning all game licenses associated with that account. Moreover, Sybil attacks become infeasible, as they would require monetary investments and/or a real-world identity for even the most trivial actions, such as initiating chats with other players.

4.1.1 Valve Anti Cheat System

Valve also provides the Valve Anti-Cheat (VAC) service that detects players who cheat and marks their profiles with a publicly visible, permanent *VAC ban*. Server operators can "VAC-secure" their servers: any player with a VAC ban for a given game can not play that game on VAC-secured servers (but they are allowed to play other games). In an effort to hinder the creators and distributors of cheats and hacks, most of the details of how VAC works are not made public.

Valve has publicly described VAC as sending periodic challenges to gamers' machine that, for example, execute otherwise unused game code and return a response [Kus10]. If the player's machine does not respond, then an alert of a possible cheat is registered. The detection itself operates similarly to anti-virus tools which examine changes in memory and signatures of known cheats. It is important to note that VAC is designed in a manner to leak as little information as possible to potential cheat creators: VAC is continuously updated and distributed piecemeal to obscure complete knowledge of the system. Addi-

tionally, VAC bans are not issued immediately upon cheat detection, but rather in delayed waves, as an additional attempt to slow an arms race between cheat creation and detection. Valve claims what amounts to a 0% false positive rate, and there are only 10 known instances where bans were incorrectly handed out (and eventually lifted). It is difficult to ascertain VAC's false negative rate, but we believe that nearly all cheats are eventually detected.

We collected historical data on when a cheating flag was first observed from a 3rd party service, vacbanned.com, that allows users to enter a SteamID into a search box to check whether or not that SteamID has been banned. If the account is banned, the date the ban was first observed is provided. We also re-crawled (between October 18th and October 29th 2011) all Steam profiles discovered during the first crawl without a VAC ban, to identify which non-cheaters had been flagged as cheaters since April 2011. Of these, $43,465$ now have a VAC ban on record.

Vacbanned.com had observed ban dates for 423,592 of the cheaters we discovered during our initial crawl. Figure 4.1 plots a *cumulative distribution functions* (CDF) of these ban observations over time. The CDF represents the probability of a random variable x having a value less than or equal to X. For example, about 35% of retrieved ban dates were observed by June, 2010. The earliest dates indicate users that were banned prior to December 29th, 2009. We combined the "banned-since" dates from our original crawl, `vacbanned.com`, and our re-crawl. In the case of a user profile having more than one ban date (due to the 3 sources), the earliest date was chosen. It is important to note that all ban dates were treated as "on or before" as opposed to a precise timestamp. This is because the ban dates are when the ban was first observed by a 3rd party (vacbanned.com or our crawler), not necessarily when it was applied by Valve.

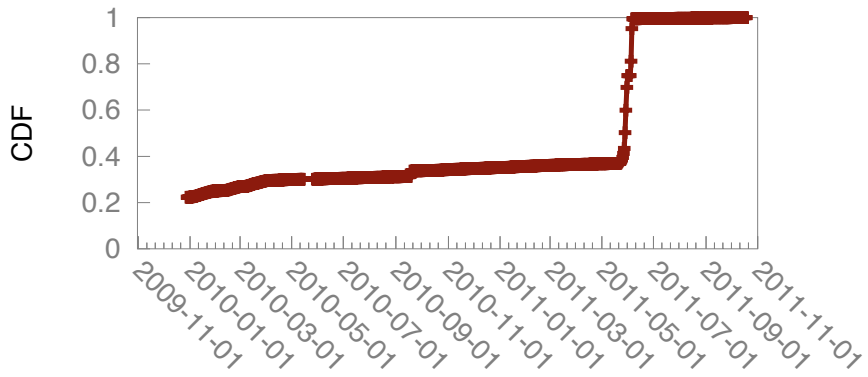

Figure 4.1: Historical VAC ban dates. The dates are collected from vacbanned.com. The date of discovery is on the x-axis, and the cumulative probability distribution of the number of discovered accounts is on the y-axis. The jump around end of May 2011 is probably due to an effort from the website to populate their database.

While the data collected from `vacbanned.com` allows us to perform a significant amount of analysis, the observations on VAC ban dates are course grain with accuracy and precision concerns due to the use of a 3rd party data base with unknown collection methodology. Additionally, it does not adequately capture changes in the structure of the Steam Community network over time. In particular, we are interested in two data fields: first, the time when a VAC ban was applied. And second, the dynamics of the relationships of a newly branded cheater soon after the VAC ban was applied.

Fortunately, two new Web API methods were made available after our initial crawl. One method provides access to timestamped friends lists and another provides the ban states (without dates) of up to 100 users at a time. With these API methods, we made daily observations of the ban status of users, as well as neighborhood observations of newly banned users and a control set of non-cheaters using the implementation described in chapter 3.

Ban status observations consist of the id of the observed user, the time stamp the observation was made, and a delta of the ban states. There are three different ban types: 1) *VACBanned*, 2) *CommunityBanned*, and 3) *EconomyBan*.

VACBanned is the boolean cheating flag described earlier. *CommunityBanned* is a boolean that we have been unable to find documentation about. However, we discovered that, unlike a VAC ban, it is not permanent: we observed users having a community ban applied, and later removed. We suspect it bars access to Steam Community features such as commenting on other users' profiles and sending new friend requests. The *EconomyBan* is related to a cross-game [2] virtual goods trading platform added to Steam in August 2011 and can take one of three string values: 1) *none*, 2) *probation*, and 3) *banned*. Along with this platform came cheating related to virtual assets [DC09] and many of the problems discussed in [KAW+10], as well as more traditional unscrupulous acts such as bait and switch scams [KCWC07]. Trade bans are displayed on a user's profile in a similar fashion to VAC bans, but, the three value system allows probated users to be reinstated. To the best of our knowledge, trade bans are applied *manually*, although individual games might very well employ algorithmic techniques to identify, e.g., gold farmers [AKS+11] and bring offenders to the attention of Valve. Although we captured information on all ban types, the remainder of this paper deals only with VAC bans. no documentation describing their use).

Friends list observations consist of the time when the relationship was declared in addition to the ID of the friend. We note that deleted relationships are not recorded by Steam Community, which is why we record the neighborhood of selected gamers on a daily basis.

On June 20, 2012 we began daily observations of an initial set of 9,025,656 public profiles from the static data set described above. In total, we collected over 525 million ban status observations for over 9 million gamers, with Table 4.2 giving the exact numbers. Figure 4.2 plots the cumulative number of bans per day, as well as the number of new

[2] Users can trade items from game A for items from game B, and even trade unactivated game licenses

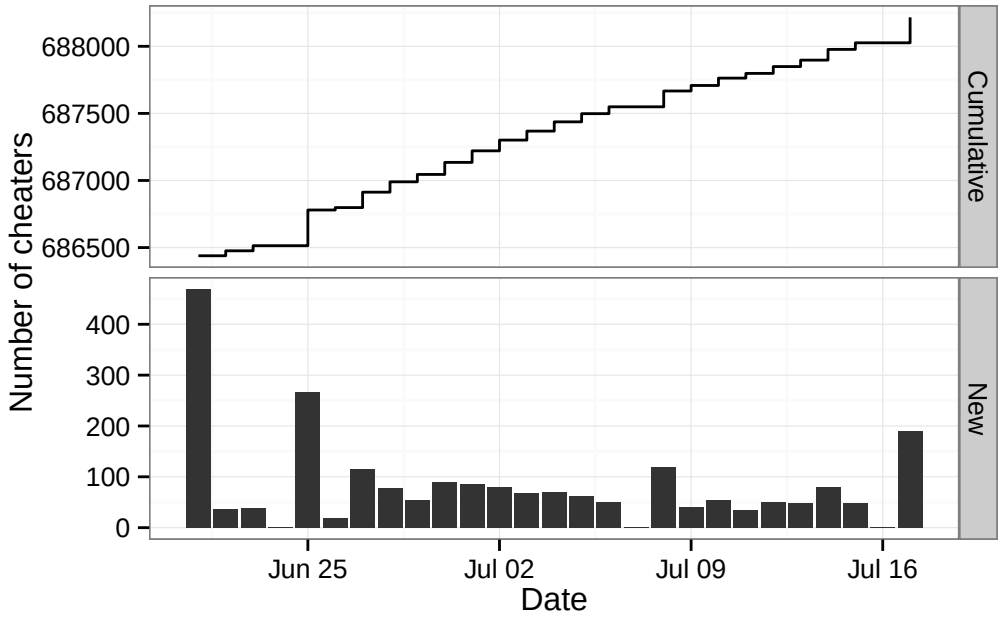

Figure 4.2: The number of cheaters in our system, per day from June 21 to July 17, 2012.

bans in the system from June 21, 2012 to July 17, 2012. We found an average of 83 users were flagged as cheaters per day, but, this number varied quite a bit: from 0 to over 400.

For monitoring the new cheaters' social neighborhood, any gamer that transitioned from a *VAC Banned = false* to *VAC Banned = true* state is treated specially. For these users, we begin a 10-day period of neighborhood observation where the friends list of the user is queried, and a delta stored once per day. Any friends of the user who do not already exist in the system are added, and will thus have their ban statuses recorded moving forward. In addition to users that transition from non-cheater to cheater, we also monitor the neighborhoods of a set of 10,000 randomly selected users (from the initial dataset) as a control group, 8,712 of which had public profiles. We call the combination of the control users and the newly discovered cheaters *monitored users*. We made 349,123 neighborhood observations of monitored users. Table 4.3 provides details on this dataset.

Table 4.2: Details of ban observations dataset.

Observations	*Users*	*VAC bans*	*Comm. bans*	*Eco. probations*	*Eco. bans*
525,427,853	9,124,454	701,448	11,701	423	313

Table 4.3: Details of neighborhood observations dataset.

Observations	*Public control group*	*Public cheaters*	*Edges*	*Total Nodes*
349,123	8,712	2,337	294,058	284,765

4.2 Team Fortress 2 In-game Interactions

Team Fortress 2 (TF2) is a team- and class-based, objective-oriented first person shooter game released in 2007. Game sessions in TF2 are hosted by individual servers, most often owned and operated by independent gaming communities. Gameplay pits two teams, Red and Blue, against each other on a variety of maps. We delineate an individual game as taking place on a single map. Some maps are symetrical, with both Red and Blue attempting to complete the same objective, and others are asymmetrical, with Blue attacking and Red defending the objective.

A player finds a server to play on via three primary mechanisms: 1) invitations to join a server from other players, 2) joining a server of a currently playing friend, and 3) the server browser. The server browser provides players with the ability to discover new servers based on a variety of criteria, including latency ("ping"), the map currently being played on the server, the number of players currently on the server, and tags supplied by the server operators. Valve has implemented a few features to improve the server browsing experience, including the ability to bookmark servers, a quick start feature, and a mechanism to blacklist servers that attempt to lie about their status (e.g., to make the server appear populated by real players when it is in fact populated by bots).

Once a player has located a satisfactory server, she can attempt to connect to it. If the server has an available slot, the connection will proceed. If there is no available slot (i.e., the server has reached its simultaneous player capacity) the server browser provides functionality to automatically connect when a slot becomes available. Players are free to join and leave a server at any time, and it is quite common for a game to start with a somewhat different set of players than it ends with.

Once players join a game they must choose a team. After choosing a team, players choose to play as one of 9 classes. Players are allowed to switch classes at any point of the game, and while certain team compositions might be more or less viable, players can choose a class independent of the choices made by the rest of their team. A key feature of TF2 is clearly defined difference in classes, even down to the silhouette of the models [MFE07], which promotes cooperative gameplay between players. A regular stream of free content updates (335 as of January 2013 [Val12c]) has kept TF2 popular since its release.

4.2.1 The Slaughterhouse Server

We obtained just over 10 months of gameplay traces (from April 1 to February 3, 2012) from "The Slaughterhouse" (SH), one of several TF2 servers operated by the "Brotherhood of Slaughter" (BoS) gaming community. The server, located in Los Angeles, California, hosts up to 30 players simultaneously, costs approximately $250.00 a month to operate, and is completely funded by donations from the BoS community.

SH has been customized with a variety settings. Of note is the *alltalk* setting, which broadcasts all voice communications to both teams, chosen by the BoS server administrators to foster a fun, social atmosphere, as opposed to a purely competitive environment. As voice

communication in games like TF2 both influences, and is influenced by, gameplay [TMW+12], a server like SH can produce an intense social gaming experience when filled with a talkative crowd.

The logs contain information such as gameplay events, in-game team and server-wide text chat, and map nomination and votes. They are in a special format that has been further customized by server plug-ins selected by the SH administrators. Each line in the log file is in a different format, and game-play events in particular have variable components. E.g., multiple people can capture a control point.

We extracted 12,621,543 gameplay events from the log files. Such events include, for example, one or more players capturing territory together, or two players on the same team working together to "kill" a player on the opposing team.

From the extracted events, we created an undirected *interaction graph* where an edge exists between two players if there was at least one event that involved both players. Each edge is annotated with a time series corresponding to the times of the extracted event between the players. In total there were 18,743,644 pairwise interactions, i.e., the sum of the length of all edges' annotated time series.

Because all TF2 players can be linked to a Steam account via the SteamID that appears in the log files, we also constructed a declared social network of players. Thus, the in-game interactions data set is composed of four networks: 1) Steam Friends, 2) Server Friends, 3) Interaction, and 4) Interacting Friends.

The *Steam Friends* network is the declared social networks of all players in our log files. It is important to note that this network includes gamers that did not actually appear in our log files. Even though they might be friends on Steam Community, they might

Table 4.4: Details of the SH interaction dataset.

Graph	*# Players*	*# Edges*
Steam Friends	33,546 (620,789 non-players)	1,038,133
Server Friends	22,099	50,522
Interaction	33,546	1,768,528
Interacting Friends	7,501	13,270

not have played together on this particular server; either during the period our log files capture or ever. The *Server Friends* network is simply the subgraph of the Steam Friends network that includes only players appearing in the log files, regardless of whether or not they had an interaction together. The *Interaction* network is a graph of players where an edge exists if the two players had an in-game interaction. Finally, the *Interacting Friends* network is the intersection of the Server Friends and Interaction networks. I.e., an edge exists between two vertices if they are declared friends *and* had an in-game interaction.

Table 4.4 presents the size of our dataset. There are 33,546 players on the server who are part of the Steam Community, involved in over 1 million relationships. Of them, 22,099 have 50,522 friendships where both friends played on the server. Of these, 7,701 friends interacted on the server during our observations, forming 13,270 interactive pairs.

4.3 The League of Legends Tribunal

The League of Legends (LoL) is a match-based team competition game developed by Riot Games. It is currently one of the most played online games in the world, and has a competitive eSports scene with weekly tournaments that are broadcast live around the world. Teams are most often composed of five players who are randomly matched together, and friends can also form pre-made teams.

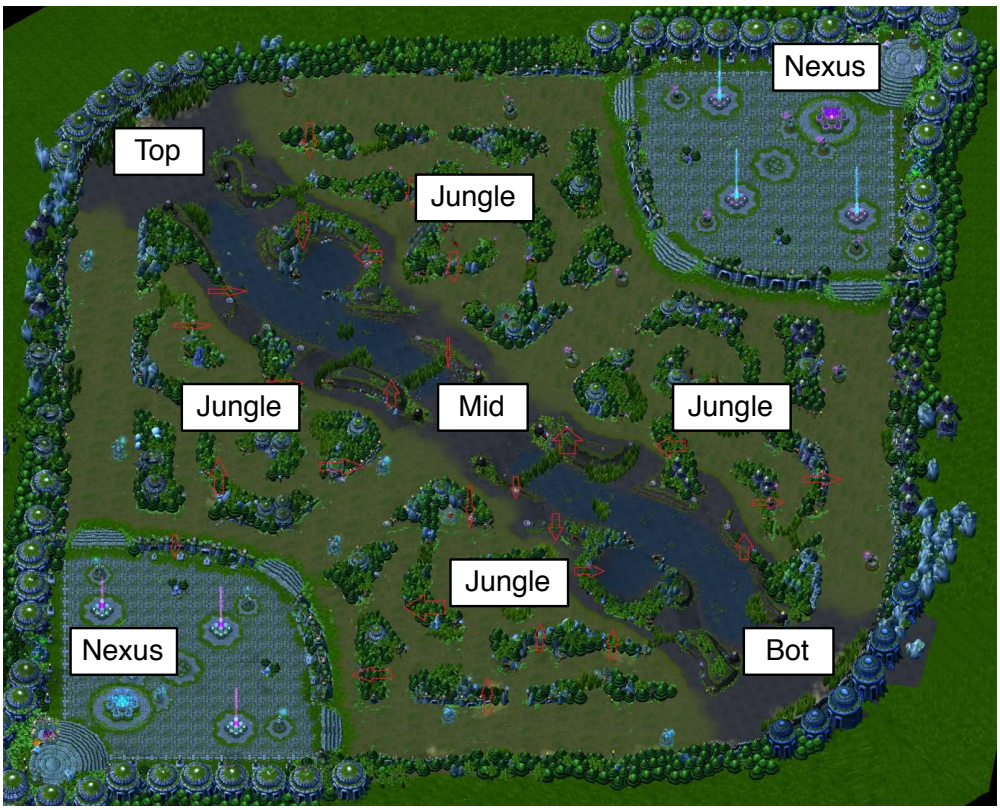

Figure 4.3: The League of Legends map, featuring 3 symmetric lanes, separated by a jungle. The goal of the game is to destroy the enemy team's Nexus by first destroying the towers in the lanes.

LoL features a symmetric map with three paths, or "lanes". The lanes are colloquially known as "top", "mid", and "bot" and have a "jungle" area between them. The goal of LoL is to penetrate the enemy team's central base, or "Nexus", and destroy it. Figure 4.3 illustrates the layout of the LoL map. To destroy the Nexus, players must first destroy towers in at least one lane. Typically, when a match starts players choose to go to one particular lane, often taking on different roles depending on the character they chose to play.

Players' in-game characters are called champions. Riot Games has released 115 champions as of September 2013. A weekly rotation of 10 champions is offered to all players, but they

can also be permanently purchased via points earned through play or real world money. Each champion has different strengths and thus naturally lend themselves to the different styles of play expected from a particular lane.

After a match, players can report toxic players in one of 10 predefined categories: assisting enemy team, intentional feeding, offensive language, verbal abuse, negative attitude, inappropriate name, spamming, unskilled player, refusing to communicate with team, and leaving the game/AFK [away from keyboard].

Intentional feeding means a player allowed themselves to be killed by the opposing team on purpose, and leaving the game is doing nothing but staying at the base for the entire game. To understand the motivation behind intentional feeding and assisting the enemy team, we present a common scenario when such toxic play happens. LoL matches do not have a time limit, instead continuing until one team's Nexus is destroyed. If a team is at a disadvantage, real or perceived, at some point during the match, they may give up by voting to surrender. However, surrendering is allowed only after 20 minutes have passed, and the vote requires at least four affirmatives to pass. If a surrender vote fails, players who voted for the surrender may lose interest and regard continuing the game as wasting time. Some of these players exhibit extreme behavior in response to the failed vote.

4.3.1 The Tribunal

To take the subjective perception of toxic behavior into account, Riot Games developed a crowdsourcing system to judge whether reported players should be punished, called the *Tribunal*. The verdict is determined by majority votes.

When a player is reported hundreds of times over dozens of matches, the reported player is brought to the Tribunal[3]. Riot Games randomly selects at most five reported matches [4] and aggregates them as a single case. In other words, one *case* includes up to 5 *matches*. Cases include detailed information for each match, such as the result of the match, reason and comments from reports, the entire chat log during the match, and the scoreboard, that reviewers use to decide whether that toxic player should be pardoned or punished, as seen in Figure 4.4. To protect privacy, players' handles are removed in the Tribunal, and no information about social connections are available. We note that our data set does not include reports of unskilled player, refusing to communicate with team, and leaving the game in our datasets, even though players are able to choose from the full set of 10 predefined categories[5].

The Tribunal also institutes mechanisms to maintain the quality of reviewers' tasks. One is a limit on the number of cases that reviewers can see a day. Next is a minimum duration for decision making, limiting mechanical clicks without careful consideration. Another is a skip feature for difficult cases.

After some time has passed, reviewers can see the crowdsourced decisions for the cases they reviewed. Both the final decision and the level of agreement are accessible, as seen in Figure 4.5. To encourage user participation, Riot Games adopts reviewers' accuracy score and ranking as gamification element.

Data (summarized in Table 4.5) was provided by [BK14] which collected over 1.46 million cases in the Tribunal, made up of 10 million user reports using distributed crawlers similar to those described in Chapter 3. Riot divides the world into several regions, each served

[3]http://tinyurl.com/stfunub4
[4]http://tinyurl.com/stfunub5
[5]To the best of our knowledge, this is intentional on the part of Riot.

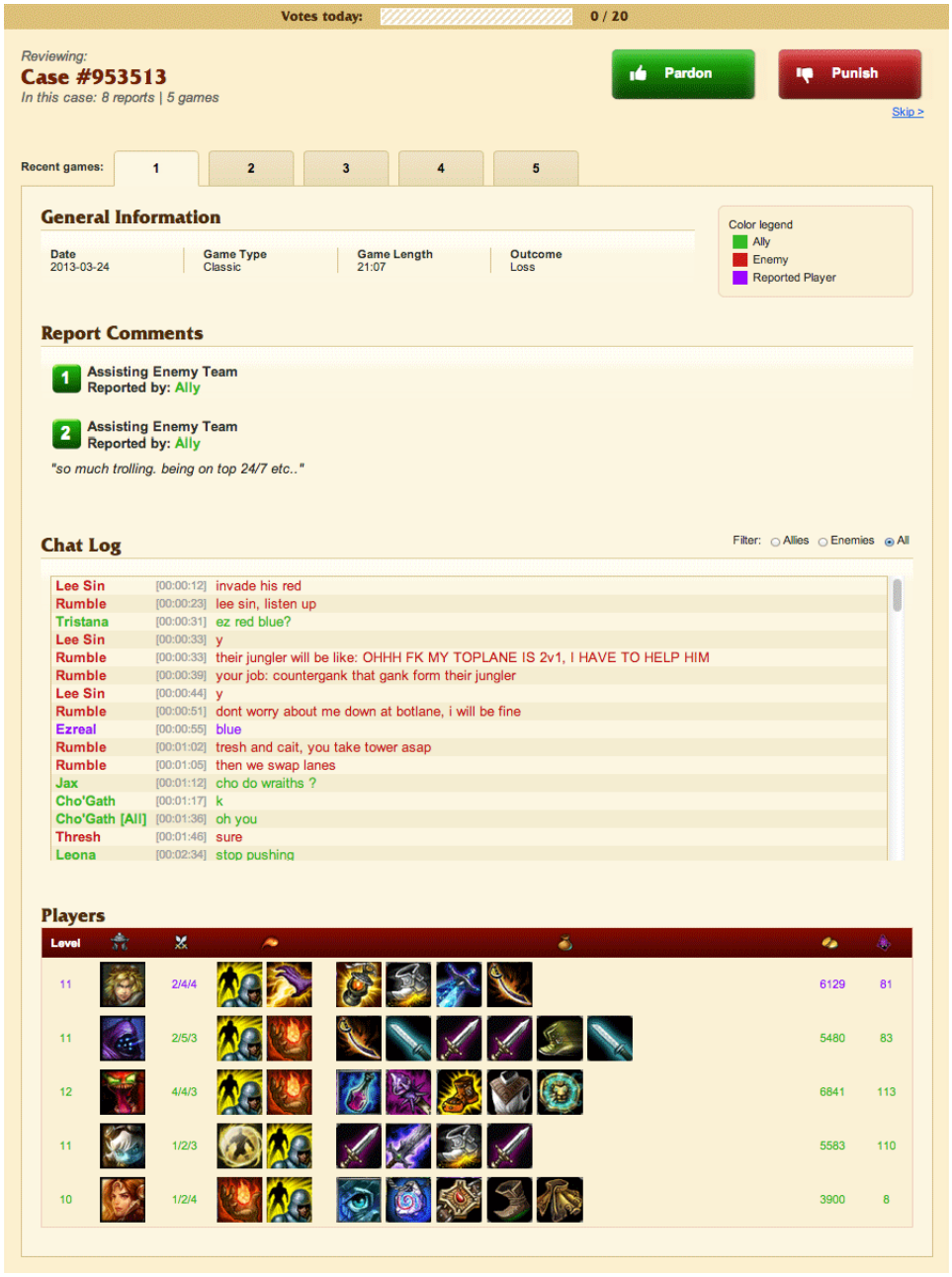

Figure 4.4: A tribunal case as presented to a reviewer.

Table 4.5: LoL Tribunal data set summary.

	EUW	NA	KR	Total
Cases	649,419	590,311	220,614	1,460,344
Matches	2,841,906	2,107,522	1,066,618	6,016,046
Reports	5,559,968	3,441,557	1,898,433	10,899,958

Figure 4.5: The decision of reviewed cases, available after some time.

by dedicated servers due to quality of service concerns. The data was collected from North America (NA), Western Europe (EUW), and South Korea (KR) servers. These regions were selected by considering the representativeness of cultural uniqueness and familiarity of the authors [BK14]. We reasonably assume most of the players connect to the servers in the region closest to them for the best user experience. This assumption is supported by observations from the Steam Community dataset (Section 5.4.1).

4.4 Summary

In this chapter we presented our data sets. We described the Steam Community and high level statistics of the over 10 million user profiles we collected. We additionally introduced a dynamic network with over 500 million daily observations of ban statuses and neighbor-

hood changes for a subset of Steam Community users. We then presented the data sets derived from 10 months of detailed gameplay logs on a community owned and operated server. Finally, we explained the LoL Tribunal and our dataset of 1.46 million cases on toxic behavior. In the next chapters, we analyze these data sets and discuss the implications of our findings.

Chapter 5: Social Network Analysis of Cheaters in Steam Community

[1]One line of thought in moral philosophy is that (un)ethical behavior of an individual is heavily influenced by his social ties [Par84]. Under this theory, cheaters should appear tightly connected to other cheaters in the social network. On the other hand, unlike in crime gangs [AKS+11], cheaters do not need to cooperate with each other to become more effective. Moreover, playing against other cheaters may not be particularly attractive as any advantage gained from cheating would be canceled out. These observations suggest that cheaters may be dispersed in the network, contradicting the first intuition.

To understand the position of cheaters in the social network, we raise four research questions, initially explored in [BSK+12, BSL+11, BKS+14]:

- Research Question 1: Are cheaters "social gamers"?

At its surface, cheating appears to be anti-social behavior: cheaters are violating social norms for their own benefit to the detriment of others. This might lead us to believe that cheaters themselves are anti-social. However, this dissertation is concerned with games which *rely* on social interactions between players. Thus, we use socio-gaming metrics to

[1]Much of the work in this chapter was first published in Proceedings of the 21st International Conference on World Wide Web, 2012 [BSK+12] and ACM Transactions on Internet Technology, 2014 [BKS+14]. Permission is included in Appendix B.

explore the relationship between game ownership, play time, and cheater status and find that cheaters and non-cheaters are quite similar.

- Research Question 2: What is the relationship between cheating status and friendships?

If cheaters are social gamers, then we might be expect them to have different socialization patterns. For example, just because a cheater is interested in the social aspect of gaming, does not necessarily mean that they are particularly engaged in the Steam Community OSN. We find that although cheaters are well embedded in the social network, they tend to associate with each other more than non-cheaters.

- Research Question 3: Are cheaters visibly penalized by other members of the social network?

While substantial evidence indicates that cheaters are generally frowned upon by the gaming community, the permanent and publicly visible cheater status in Steam Community allows us to quantify the reaction to cheaters. While VAC bans result in restricted play, do they have any observable effect on social standing? Observations on reactions to cheater status help us better understand how cheating impacts a community, as well as providing insight on possible mitigation strategies. By studying changes in neighborhood sizes, we find that cheaters face some social penalties for their sins.

- Research Question 4: What is the relationship between cheating status and gamer proximity?

Measuring how "close" users are gives us better understanding of network dynamics. For example, do cheaters tend to form ties with other users who are within short geographic distance of each other? Answering this research question can provide insight into cultural perceptions of cheating, as well as how cheaters form relationships with each other and fair users. We use three metrics for closeness and find that cheaters are quite close to each other with respect to geograpy and position in the social network.

5.1 Socio-gaming

While the majority of this chapter is concerned with how cheaters are positioned within Steam Community, understanding their behavior as gamers helps to better understand their interactions in the community. To this end, we analyze the number of games owned and hours played per game genre using the tags provided by the Steam Store to describe each game, and place games in the following categories: single-player, multi-player, single-player only, multi-player only, and co-op.

We use the categories "single-player" (the game can be played by a single human player) and "multi-player" (the game supports multiple human players) for two reasons: first, VAC bans only have an effect on multi-player games, and second, all games are tagged in at least one of these categories. Some games do not contain a single-player component at all. We classified these types of games as "multi-player only" if they were tagged as multi-player but not single-player, and those with no multi-player component are likewise classified as "single-player only". Finally, "co-op", or cooperative games, are loosely defined as multi-player games with a mechanic focusing on co-operative (as opposed to competitive) interaction between human players. For example, players might work together to defeat

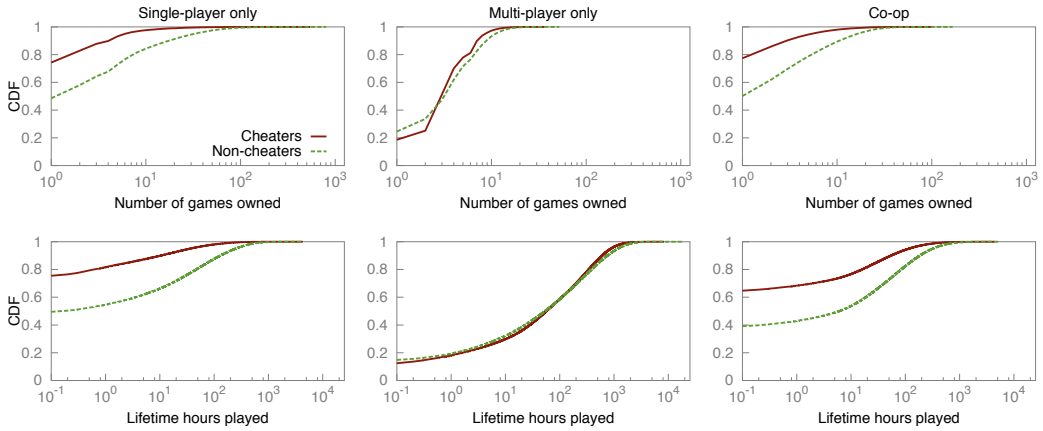

Figure 5.1: The number of games owned and lifetime hours per genre for cheaters and non-cheaters from the March 2011 crawl, and newly discovered cheaters from the October 2011 crawl.

a horde of computer controlled goblins [Tre11], or to excavate a landscape and build a city [Re-11].

Figure 5.1 plots the CDF of the number of games owned and the lifetime hours on record per game category for cheaters and non-cheaters in our initial crawl (static snapshot). Around 60% of cheaters and non-cheaters have less than 100 lifetime hours played in multi-player only games. For brevity, we only present the plots representing the ownership of single-player only, multi-player only, and co-op games.

These results lead to the following observations. First, they provide confirmation of gaming as a social-activity: gamers on Steam Community are far more likely to own more than one multi-player games than single-player games, even though there are over twice as many single-player games available on Steam. This trend is even clearer when considering single-player only games vs. multi-player only games.

Next, we observe that non-cheaters are more likely to own more games than cheaters in general. However, the difference in number of games owned between cheaters and non-

cheaters is significantly smaller for multi-player only games than for single-player only games. This provides an initial indication that cheaters are social gamers: even though they might not own as many games as a whole, they are as interested in multi-player games as non-cheaters are.

When considering the lifetime hours played per category, we see a similar story. Cheaters tend to play fewer hours of single-play only and co-op games than fair players. This is not entirely expected, as cheating can be motivated not only by competition with other players, but also for advancing in the game to access higher levels of fun [Dum11]. Because the co-op tag was later introduced, it is possible that not all co-op games are properly tagged: however, of the games tagged as co-op, cheaters tend to own fewer games and play fewer hours than non-cheaters.

The message of this analysis is that cheaters are most definitely social gamers: they favor multi-player games over single-player games for both purchase and play time. Specifically, cheaters are much less interested in games *without* a multi-player component. This strengthens the conclusions we can draw regarding further analysis on cheaters' social positioning.

5.2 Cheaters and their Friends

A common first step in network analysis is to examine degree distributions. By this, we mean looking at the probability of a user having a certain number of friends. One method of analysis involves comparisons of cumulative distribution functions. A common way to visualize this type of analysis is to plot a variant of the complementary cumulative

distribution function (CCDF) on a log-log axis. The CCDF represents the probability of a random variable x having a value greater than or equal to X[2].

Figures 5.2 and 5.3 presents the degree distribution for the Steam Community graph as a whole and for just cheater profiles, as well as for private, friends-only profiles, and for users users without a profile, respectively. For example, about 10% of Steam Community users have at least 20 friends. For users without a profile or with private profiles, edges in the graph are inferred based on the information from public profiles that declare the user as a friend. From the degree distributions we make two observations.

First, we discovered a hard limit of 250 friends (this limit has since been raised if a user links their Facebook account to their Steam Community account or fulfills other criteria). However, there are some users who have managed to circumvent this hard limit. One user in particular has nearly 400 friends, and through manual examination we observed this user's degree increasing by one or two friends every few days. Coincidentally, this profile also has a VAC ban on record.

Second, all categories plotted in Figures 5.2 and 5.3, with the exception of users with Steam accounts but no profiles, overlap. This means that the distribution of the number of friends cheaters have is about the same as the non-cheaters' distribution. It also highlights that attempting to hide connection information through *private* or *friends-only* profile privacy settings is mostly unsuccessful: the player's position in the social network is revealed by the privacy settings of his friends.

While cheaters are mostly indistinguishable from fair players using the node degree distribution, a more important question is whether their behavior shows network effects. In

[2]Some definitions of the CCDF are strictly greater than, however, degree distributions in social networks are often plotted using the greater than or equal to definition.

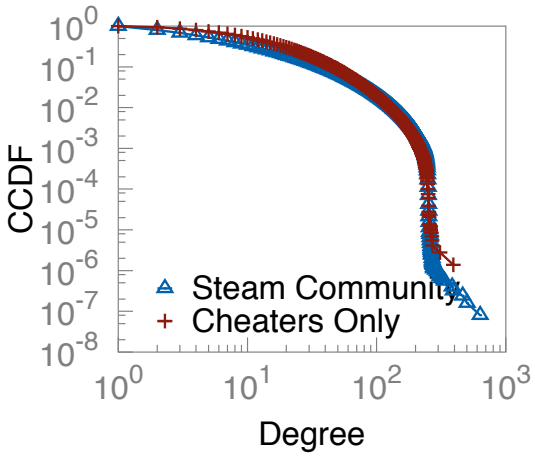

Figure 5.2: Degree distribution for all users and cheaters in Steam Community.

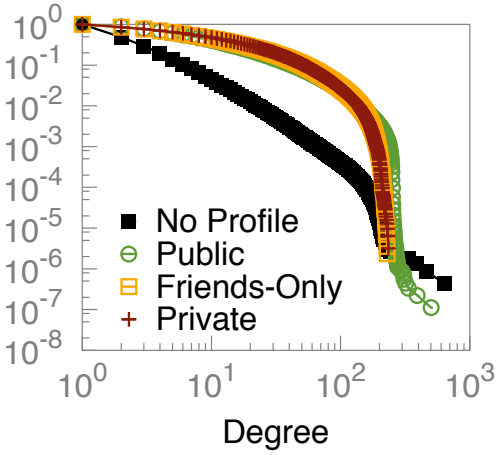

Figure 5.3: Degree distribution for users with public, private, friends only, and no profile in Steam Community.

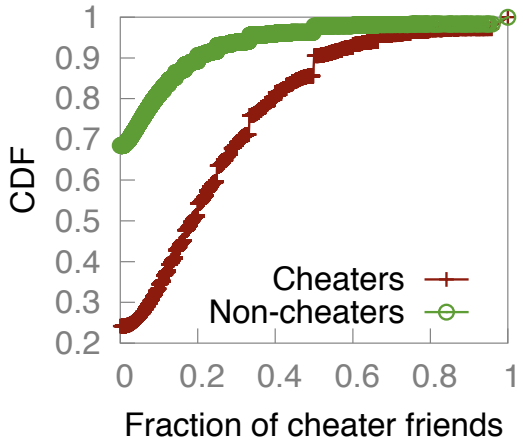

Figure 5.4: CDF of the fraction of cheaters' friends that are cheaters vs the fraction of non-cheaters' friends that are cheaters.

other words, are cheaters more likely to be friends with other cheaters than with non-cheaters? Figure 5.4 plots the CDF of the fraction of a player's friends who are cheaters. Figure 5.5 plots the CCDF of the number of cheater friends for both cheaters and non-cheaters. This figure is comparable to Figure 5.2, but displays only the size of the cheating neighborhood.

The picture that emerges from these two figures is a striking amount of homophily between cheaters: cheaters are more likely to be friends with other cheaters. While nearly 70% of non-cheaters have *no* friends that are cheaters, 70% of cheaters have at least 10% cheaters as their friends. About 15% of cheaters have over half of their friends other cheaters.

While the differentiation is visually apparent, we ensured it was statistically significant via two methods: 1) the two sample Kolmogorov-Smirnov (KS) test, and 2) a permutation test to verify that the two samples are drawn from different probability distributions. An explanation of the two methods appears in Appendix A. We find that the distributions are in fact different with $p_{ks} < 0.01$, $D = 0.4367$, $p_{permute} < 0.01$, $T = 969.0140$, and $p_{ks} < 0.01$, $D = 0.4752$, $p_{permute} < 0.01$, $T = 766.8699$ for Figures 5.4 and 5.5, respectively.

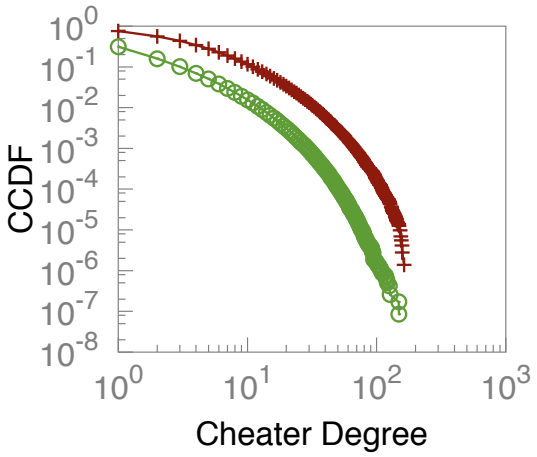

Figure 5.5: CCDF of the number of cheater friends, the "cheater degree", for both cheaters and non-cheaters.

Declared relationships in online social networks are inconsistently backed by actual interactions [WBS+09]. While gaming is a primarily interactive community, with the declared ties not always necessary for the primary purpose of the community, we verify that the number of declared friends translates into gaming interactions. To this end, Figures 5.6 and 5.7 plot the average number of games owned and hours played, respectively, in the two weeks prior to our March 2011 crawl as a function of social degree. There is a strong correlation between the degree of a user and the average number of hours played (0.70 for the whole Steam Community, 0.64 for cheaters). The figures show a positive correlation between the social degree of a gamer and the average number of games owned (0.38 Pearson correlation score) and a clear positive correlation for cheaters (0.63). In Chapter 6 we perform a more detailed analysis related to declared relationships and in-game interactions using the SH log files.

In other words, the investment in gaming (both monetary and time) increases with the number of friends for both cheaters and non-cheaters. Even though cheaters are involved in decidedly anti-social behavior, they still have a positive response to the social phenom-

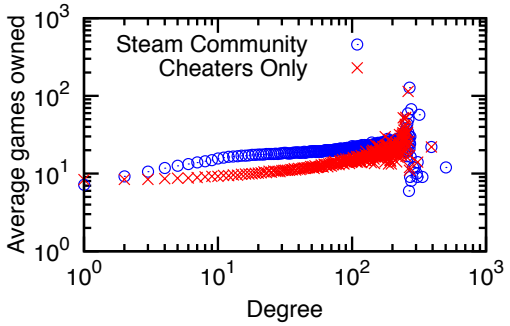

Figure 5.6: The number of games owned as a function of the number of friends.

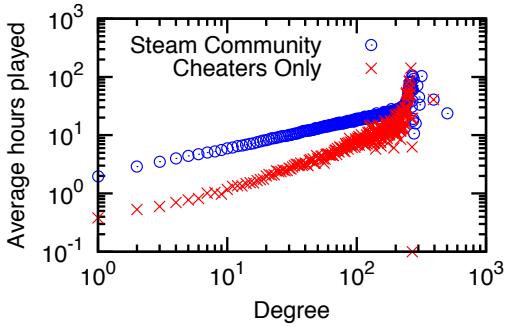

Figure 5.7: The number of hours played as a function of the number of friends.

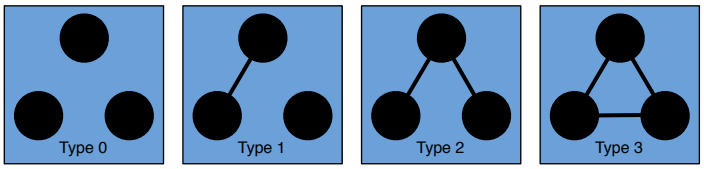

Figure 5.8: Possible triad configurations in an undirected social network.

Table 5.1: Undirected triad census of Steam Community.

Type 0	Type 1	Type 2	Type 3
3.23×10^{20}	1.10×10^{16}	4701213029	153466265

ena of gaming. This is an important result, as it introduces the possibility of VAC bans having more than just a utilitarian impact on cheaters: not only is their technical ability to game affected, but the ban might also impact their standing with their gameplay partners. In fact, in the next section we show that there are indeed negative social effects associated with being branded a cheater.

Next, we took a *triad census* of Steam Community. A triad census is a way of measuring the connectivity and clustering between people in a social network. Although dyad (2-person) level relationships are important, triadic (3-person) relationships are known to occur at significantly different rates than in random networks. Each triad type is differentiated based on the number of links present between the three nodes. Figure 5.8 illustrates the four possible triad configurations in an undirected social network like Steam Community, and Table 5.1 shows the results. Because there are a total of $\binom{n}{3}$ possible triads, we show approximate values for type 0 and 1 triads in Table 5.1.

We can get a rough measure of how often social clustering occurs by looking at the proportion of type 3 triads that could exist. I.e., we look at the proportion of closed triangles out of triangles with at least two edges. In a random network, we would expect this number to be very close to the density of the network. In our case, this is around 0.3, which is

much greater than the network density. We can thus say that users on Steam Community are indeed exhibiting social clustering.

5.3 The Social Response to the Cheating Flag

While aggregate-level information shows little differentiation between cheaters and non-cheaters, the effect of the VAC-ban mark can better be understood by analyzing the transition from non-cheater to cheater. We answer the following two questions: 1) Are cheaters shamed by the mark on their permanent record? and 2) Does the community shun cheaters once their transgressions are revealed?

Of the new cheaters discovered from our October re-crawl, 87% had no change in privacy state, and nearly 10% changed their privacy setting from public to a more restrictive setting. In comparison, in our control group of re-crawled non-cheaters, privacy settings remained unchanged for over 97% of users, and less than 3% changed to a more restrictive setting. Cheaters seem to value their privacy more once their sins are laid bare, perhaps in the naive hope that a more restrictive setting will provide a measure of protection from a potentially disapproving community.

But *is* the community disapproving? Figure 5.9 plots the CDF of net change in the number of friends for cheaters and non-cheaters during the six months between our two static snapshot crawls. Of the still public cheaters in our re-crawl, 44% had a net loss of friends, 13% had a net gain, and 43% had no change. Of the non-cheaters in our new crawl, 25% had a net loss of friends, 36% had a net gain, and 39% had no change. While both sets of users exhibited fluctuations in the size of their neighborhoods, more cheaters lost friends and fewer cheaters gained friends when compared to non-cheaters. Treated as a whole,

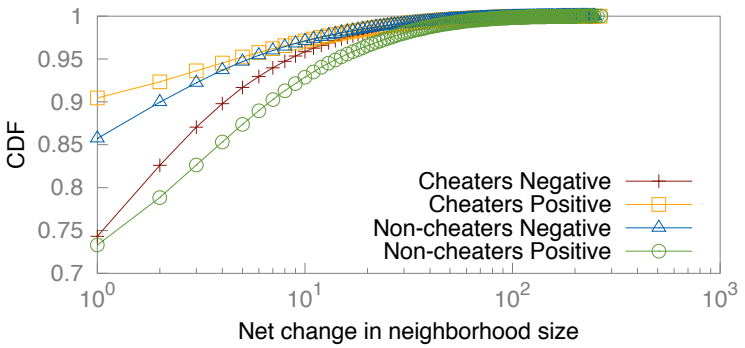

Figure 5.9: CDF of net change in cheaters' and non-cheaters' neighborhood size.

cheaters lost nearly twice as many friends as they gained, and non-cheaters gained twice as many friends as they lost. Overall, non-cheaters continue to gain friends, and cheaters, while not overtly ostracized, appear to have trouble making new friends and may lose a few of their previous ones.

Using the high resolution observations, we plot the total fraction of friends lost and gained, relative to the first day of observation, for the control users and new cheaters in Figures 5.10 and 5.11, respectively. Note the different scales.

We additionally ran an Analysis of Variance (ANOVA) to determine whether group and day of observation have a significant effect on degree. The findings show a significant interaction for between cheater/non-cheater status and day of observation ($p < 0.01$ for both). They thus confirm the findings from the low resolution static snapshots, and further demonstrate that newly banned cheaters, while not completely ostracized, tend to lose many more friends than they gain. Further, the reaction occurs very quickly: the greatest loss in friends happens within the first few days of the ban becoming publicly visible (Figure 5.11).

There are several possible explanations for the changes in neighborhood sizes we observe. First, evidence suggests that online gamers "clean up" their friend lists to make room for

new friends, removing people they no longer play with [XCS+11]. However, because so few users are near the 250 friend limit (as seen in Figure 5.2), we do not believe this is the primary contributing factor to neighborhood size fluctuations. Additionally, while the control group's pattern is what we might expect from a clean up of friends lists, the speed and degree of friendship loss for the new cheaters indicates a response to the application of the cheating flag.

This provides quantitative insight that human beings react to unethical behavior in a similar fashion as we do to epidemic disease (further explored in Chapter 7), and indicates future directions towards modeling and suppressing the spread of unethical behavior. For example, the publicly visible cheating flag can be seen as a form of information-driven vaccination [RTL12]: knowledge of the unethical behavior is not hidden, and thus spreads through the network causing non-cheaters to sever ties with new cheaters. Models with both rewiring [GDB06] and information-driven vaccination [RTL12] have been shown to have an effect on the spread of epidemic contagion in a network, and our analysis provides new empirical insight into these phenomena. In Chapter 7, we perform a more thorough analysis of cheating as a contagion process.

A second explanation is that the Steam client, by default, issues "pop up" notifications that are visible in game whenever a friend starts playing any game. If a gamer has many active friends, these pop ups might become distracting, possibly prompting a purge of friends they no long actively play with regardless of how close they are to the friend limit.

A final explanation, especially with respect to the net loss of friends cheaters suffer, is that cheaters are deliberately severing their ties once they are caught cheating. We observed one account in particular that went from 200 to 0 friends after the VAC ban was issued.

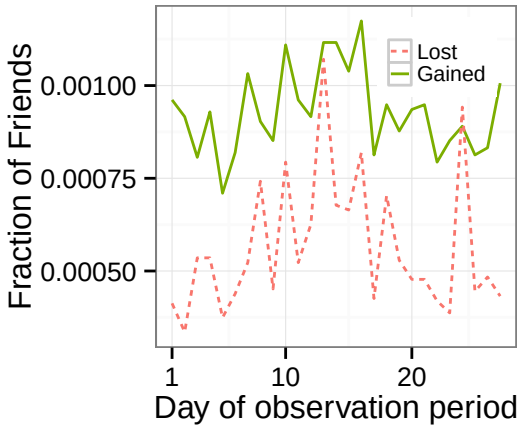

Figure 5.10: The total fraction of friends gained and lost for the control group of cheaters.

"Social suicide" might account for large decreases in degree, as it is far more probable that the cheater himself deletes friends, rather than each of his friends deleting the cheater.

5.4 Social Closeness of Cheaters

We use two metrics, one geographical, and another social, to quantify the strength of the relationship among Steam Community users in general and among cheaters in particular.

5.4.1 Geographical Proximity

Exploring the relationship between geographical and social-network proximity may give quantitative support to the theory proposed in [Dum11] according to which opinions on cheating are culturally derived. We thus first observe that user population on Steam Community does not follow real-world geographic population and, more importantly, that cheaters are not uniformly distributed with respect to geo-political boundaries.

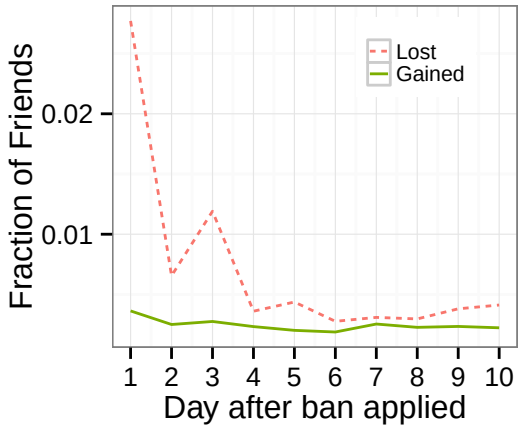

Figure 5.11: The total fraction of friends gained and lost for the cheaters.

Figure 5.12 shows the fraction of cheaters in the populations for the twelve countries comprising the union of the top ten user populations and the top ten cheater populations. The figure shows that cheaters are vastly overrepresented in some locations. For example, there are about 55,000 cheaters in the Nordic European countries (12.4% of the playing population of the region), while there are about 39,000 cheaters (3.9%) in the US. These regional differences become more striking when taking real-world population into account. In particular, we found enough Steam profiles to account for nearly 2.5% of Denmark's 5.5 million residents, of which cheaters account for nearly 0.5% of Denmark's population.

We now ask two additional questions: 1) Does the Steam Community exhibit properties of a location-based social network? and 2) Do cheaters tend to form geographically closer relationships with other cheaters than non-cheaters? To answer these, we measure node locality and geographic clustering coefficient introduced in [SMML10].

We begin by defining D_{uv} to be the geographic distance between two nodes, u and v. Similarly, the link length, l_{uv}, is the geographic distance between nodes u and v that share an

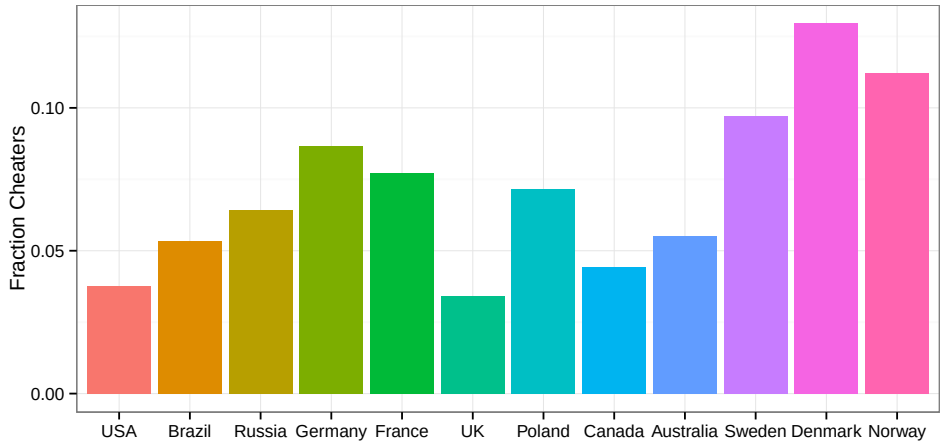

Figure 5.12: The fraction of cheaters for the union of the top 10 player and top 10 cheater population countries. The countries are sorted on the x-axis in decreasing order of their real-world populations.

edge in the network. *Node locality* NL_u of node u with degree k_u and neighborhood Γ_u is a measure of how geographically close u is to its direct neighbors, as:

$$NL_u = \frac{1}{k_u} \sum_{v \in \Gamma_u} e^{-l_{uv}/\beta}$$

Because various networks might have different geographic scale, β serves as a scaling function to allow comparisons of geo-social properties regardless of geographic scale. For this reason, we set β equal to the mean distance between nodes, $\langle D_{uv} \rangle$, for a given network.

Next, similar to the clustering coefficient which measures how densely connected a node's neighbors are by measuring how many triangles exist between its neighbors in proportion to the number of possible links between them, the *geographic clustering coefficient* measures the likelihood of triangles being geographically close. This is achieved by weighing the classical clustering coefficient of a node by the maximum length of links between nodes

in the triangle under consideration. The weight of a triangle consisting of nodes u, v, and w is defined as:

$$c_{uvw} = e^{-\frac{max(l_{uv}, l_{uw}, l_{vw})}{\beta}}$$

again scaling networks at different geographic scale with β. If there is no link between v and w, $c_{uvw} = 0$ by definition.

From here, the geographic clustering coefficient of a node u, with degree k_u, and neighborhood Γ_u is:

$$GC_u = \frac{1}{k_u(k_u - 1)} \sum_{v,w \in \Gamma_u} c_{uvw}$$

The node locality of a given node quantifies how close (geographically) it is to *all* of its neighbors in the social graph. Thus, a node locality of 1 indicates that a given node is at least as close to all of its neighbors as any other node in the graph is to their neighbors, and a value of 0 indicates that a given node is further away from all its neighbors than any other node in the graph.

We constructed the *location network* by including an edge from the social network if and only if both end points had a known location. Steam Community users can optionally choose to specify their location from a total of 33,333 possible locations at the country, state and city level. Because setting a locations is optional, this lead to a reduction in the size of the network, which, along with the geo-social properties of the resulting location network, can be seen in Table 5.2. We note that a subgraph composed solely of cheater-to-

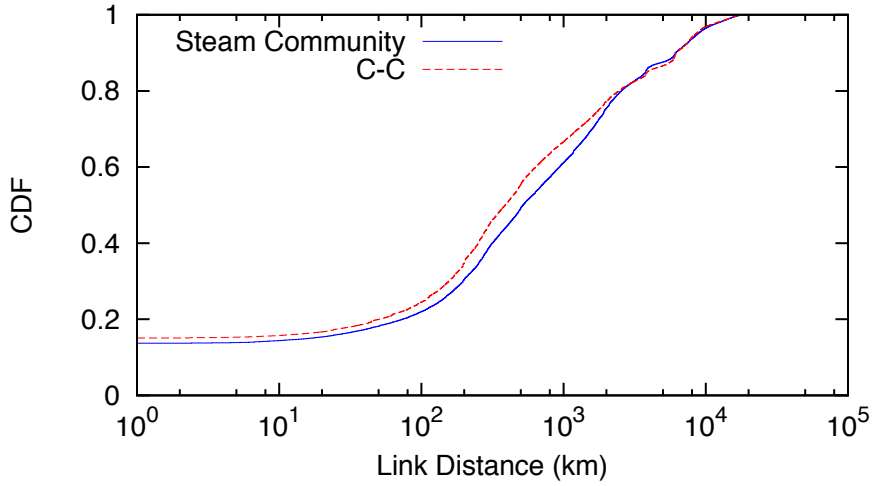

Figure 5.13: CDF of link length for Steam Community location network.

cheater relationships (C-C) has a lower mean inter-nodal distance and lower average link length than the location network as a whole.

Table 5.2: Location network properties: the number of nodes, edges, mean distance between users $\langle D_{uv} \rangle$, average link length $\langle l_{uv} \rangle$, average node locality $\langle NL \rangle$.

Network	# of nodes	# of edges	$\langle D_{uv} \rangle$ (km)	$\langle l_{uv} \rangle$ (km)	$\langle NL \rangle$
Steam Community	4,342,670	26,475,896	5,896	1,853	0.79
Steam Community C-C	190,041	353,331	4,607	1,761	0.79
BrightKite	54,190	213,668	5,683	2,041	0.82
FourSquare	58,424	351,216	4,312	1,296	0.85

The CDF of link length for the location network is plotted in Figure 5.13. About 50% of all link lengths in all networks are less than 500 km. In general, cheater-to-cheater relationships tend to be closer than the network as a whole.

Figure 5.14 plots the CDF of node locality for the location network (Steam Community), the cheater-to-cheater subgraph (C-C), which includes only edges between cheaters, as well as just the cheaters within the location network (Cheaters Only), which includes all edges with at least one cheater. We first note that about 40% of users in the location network have a node locality of above 0.9, a phenomena exhibited by other geographic online

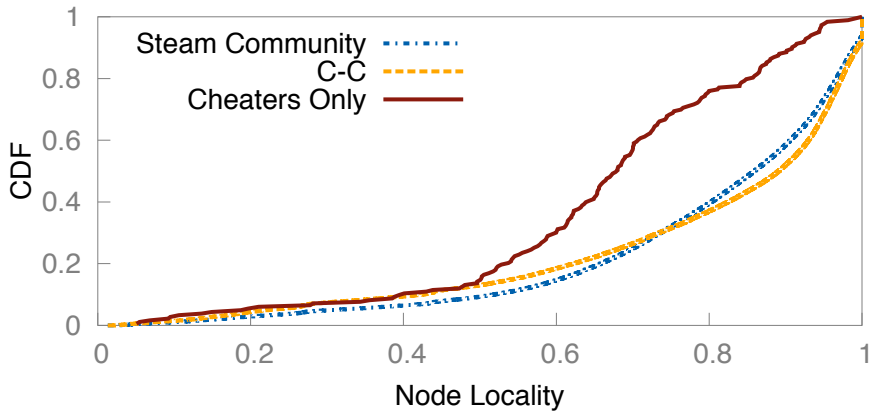

Figure 5.14: CDF of node locality.

social networks such as BrightKite and FourSquare [SMML10]. This is strong evidence that Steam Community relationships exhibit geo-social properties, a characteristic to be expected in the context of multiplayer gaming where high network latencies cannot be well masked by current game infrastructure. Next, we observe that the cheater-to-cheater network and the Steam Community at large have similar node locality distributions. Finally, when considering only the cheaters embedded within the location network, we see drastically lower node locality, with only about 10% of cheaters having a node locality greater than 0.9.

The CDF for geographic clustering coefficient is plotted in Figure 5.15. Cheaters embedded within the social network tend to have lower geographic clustering coefficients. As a whole, around 10% of the network has a geographic clustering coefficient larger than 0.5 with 4% having over 0.9. For embedded cheaters, we see only 5% with a geographic clustering coefficient of over 0.5 and 2% greater than 0.9. While a larger proportion of cheaters have a geographic clustering coefficient greater than 0.015 than non-cheaters, this trend quickly reverses, with about 40% of the whole network having a geographic clustering coefficient greater than 0.1 versus 30% of embedded cheaters. We also see that

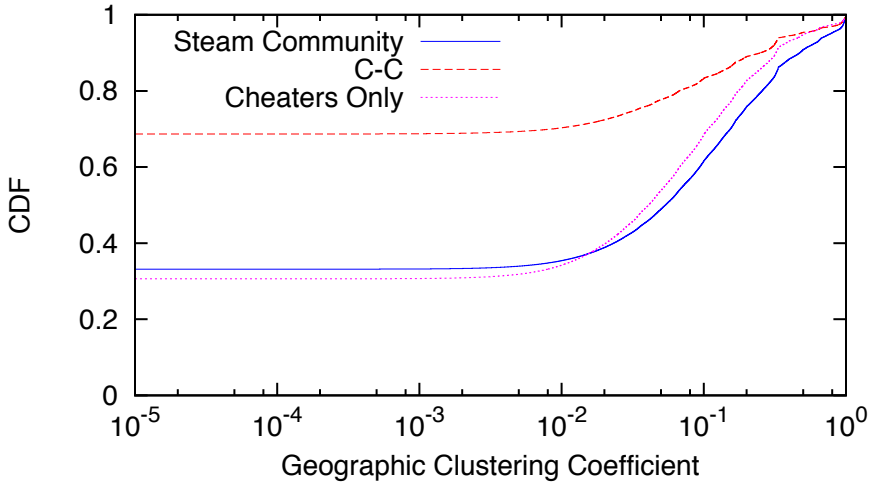

Figure 5.15: CDF of geographic clustering coefficient.

only about 30% of users in the cheater-to-cheater network, have a geographic clustering coefficient over 0.01. This is contrast to over about 65% for the entire location network and the cheaters embedded within it.

These results lead to three observations: 1) friendships tend to form between geographically close users, 2) cheaters tend to form relationships with other nearby cheaters and these links are geographically closer than those formed by non-cheaters, and 3) as evidenced by their lower node locality when considering the entire location network and not only the cheater-to-cheater subgraph, cheaters appear to befriend geographically remote fair players. This might indicate that cheaters form relationships with other cheaters via a different mechanism than they form relationships with non-cheaters. Cheater-to-cheater relationships appear geographically constrained, while their relationships with non-cheaters are over larger distances.

Finally, we note that there are some caveats with these results. First, locations in Steam Community are set by the user, and to the best of our knowledge, not enforced in any way. In other words, cheaters might be lying about their locations. Second, the locations are

selected from a predefined list, and users can specify only a country, or down to the city level. While these points should be taken into consideration, Steam Community clearly has properties of a location based social network, which is intuitive considering constraints on latency in multi-player games. In Chapter 6 we will provide evidence that Steam Community is an interaction backed social network. Since players choose servers based in large part on latency, this strengthens the premise that declared friends are geographically close. We thus believe that the effect of gamers potentially lying about their geographical location is inconsequential.

5.4.2 Social Proximity

We use *social proximity* as the second metric to characterize the strength of the relationships between Steam Community users and understand whether they materially differ for the cheater population. The social proximity metric is based on a previous study [OSH+07] that suggests that the overlap between the social neighborhood of two individuals is a good indicator of the strength of their relationship. We study the overlap of friends of users in the Steam Community networks to understand whether cheaters exhibit a stronger relationship with other cheaters than fair players do with fair players. We assess the strength of the relationship between two connected users by the overlap between their sets of friends, computed as follows:

$$Overlap_{uv} = m_{uv}/((k_u - 1) + (k_v - 1) - m_{uv})$$

where m_{uv} is the number of common neighbors between users u and v, k_u is the number of neighbors of user u and k_v is the number of neighbors of user v. This overlap is calculated

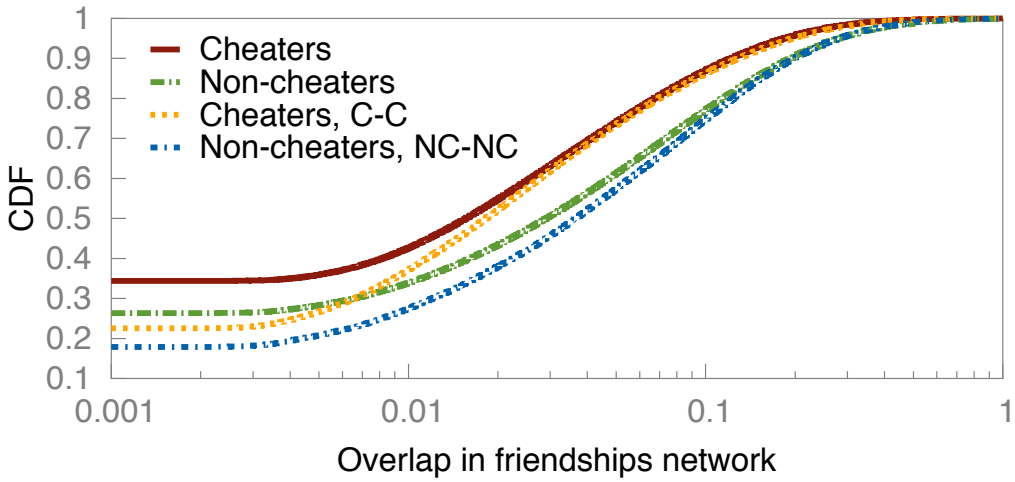

Figure 5.16: CDF of social proximity for cheater and non-cheater pairs when we consider all relationships, only cheater to cheater relationships (labeled C-C) and only non-cheater to non-cheater relationships (labeled NC-NC).

for two groups of user pairs: the 1.5 million pairs of cheaters (i.e., all cheater pairs in the full social network) and 1.5 million randomly selected pairs of non-cheaters (i.e., about 2% of the existing non-cheater pairs). Additionally, we also calculate the same metric on the cheater-only as well as on the non-cheater-only graphs.

Figure 5.16 shows a higher overlap for cheater pairs in the cheater-only graph and non-cheater pairs in the non-cheater-only graph compared to the respective overlaps in the overall social network. This suggests that social relationships are weaker between different types of players (cheaters to non-cheaters) than within a uniform group.

5.5 Summary

In this chapter, we performed a social network analysis of cheaters in the Steam Community Online Social Network. Via socio-gaming metrics, we discovered that cheaters are so-

cial gamers, greatly preferring multiplayer games over single player games. We also found that cheaters are well embedded on the social network, having about the same number of friends as non-cheaters. However, when we looked at the composition of neighborhoods, we discovered that cheaters tend to have more cheater friends, in terms of both absolute and percentage of neighborhood, than non-cheaters. Interestingly, we also saw that there is a reaction to the application of the cheating brand: newly branded cheaters tend to lose friends in comparison to non-cheaters. Finally, using geo-social metrics and neighborhood overlap, we observed that cheaters are socially close to each other.

Chapter 6: Analysis of Interactions in Team Fortress 2

Because SteamIDs of players are the same in the log files as in Steam Community, we are able to explore how in-game interactions relate to a declared social network. Thus, in this chapter we make use of the SH server logs (Chapter 4.2.1) to expand on an initial analysis of interactions that back the declared social network of Steam Community [BI13a].

This chapter is primarily concerned with answering the following research question:

- Research Question 1: What is the relationship between in-game interactions and declared relationships in the Steam Community?

Declared relationships in the Steam Community are a bit different than those in OSNs like Facebook. In Steam Community, interactions happen out of band from the social network. I.e., there is no requirement to be friends in order to play together. Thus, we can observe the events that lead up to an official declaration of friendship. We first find that interactions between friends dominate interactions between players where no friendship exists. We then examine the weeks prior to and after a relationship is created and observe an increasing number of interactions which peak right before the friendship forms and then begin to decrease almost immediately.

We start by describing several networks that we induce from the server logs and Steam Community data:

- The *server friends* network is composed of edges between players who appear in the server logs and are also declared friends on Steam Community.
- The *interaction* network is composed of edges between players with at least one in-game interaction.
- The *interacting friends* network is the intersection of the server friends and interaction networks. An edge between two nodes in this network indicates that the pair is both declared friends and had at least one in-game interaction on the TF2 server.
- The *co-presence* network is a dynamic network whereby an edge exists between two nodes at time k if they both played in game k.

Each of these networks represents a different aspect of the SH gaming community. The server friends network gives us a basic idea of how the structure of Steam Community maps on to a smaller, more localized community. One way to think about the SH server is that it is a local pub where people tend to gather to socialize and have fun. While the patrons of the pub have a relationship with each other because of their patronage, there is no guarantee this relationship persists elsewhere. To stretch the analogy a bit more, we can also say that there is no guarantee that patrons that might know each other from other situations (e.g., work) actually interact at the bar.

Conversely, the interaction network can be seen as patrons of a pub whose relationships are constrained within the walls of the pub. I.e., while they might have a drink and do some karaoke together, that is the extent of their relationship.

The interacting friends network demonstrates *interaction backed* declared relationships. In other words, these are individuals who not only had some form of social interaction, but felt strongly enough to create a declared relationship. This is important as it gives some

context to the declared relationships. Although we might not know all the details behind how the relationship was formed, what we do know is that it exists outside the ether of an OSN. Previous work [WSPZ12] has found flaws with using OSN relationships as ground truth for "real" relationships: they grow stale and users only interact with a small fraction of their declared friends. Thus, we posit that interacting friends have stronger relationships than players who either just interacted with each other or just have a declared relationship.

The construction of the server friends, interaction, and interaction friends networks is straight forward. There are 33,546 players on the server who are part of Steam Community, involved in over 1 million relationships. Of them, 22,099 have 50,522 friendships where both friends played on the server. Of these, 7,701 friends interacted on the server during our observations, forming 13,270 interactive pairs.

In the remainder of this chapter we explore the dynamics between patterns of play and our induced networks. Understanding these dynamics will help inform the simulation study we perform in Chapter 7 as well as interpret our results.

6.1 General Server Characterization

We begin with a general characterization of the SH community. Like any community, the social environment of a virtual community helps shape the interactions that occur.

Figure 6.1 shows that the community is quite active on any given day of the week. On all days, activity levels are the highest in the afternoon, and begin to fall off around midnight

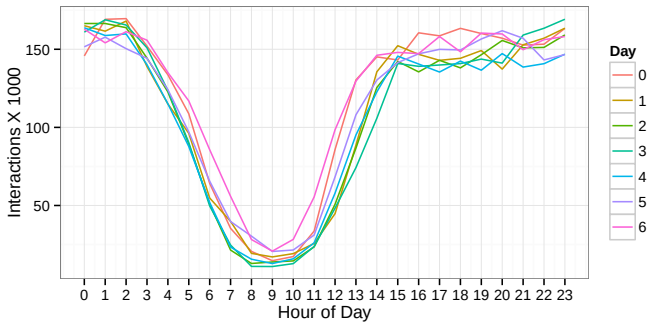

Figure 6.1: Number of interactions per hour, per-day of the week (from Sunday).

Pacific Time. We note that Saturday, a "pure" weekend day, has a relatively higher level of activity during normal working hours, with Sunday having the second highest level.

This sustained activity during non-working hours drastically differentiates this dataset from interactions in declarative OSNs such as Facebook. Gaming is a leisure activity that requires adequate, and often specialized, hardware, and more importantly, significant focus and concentration. Unlike other online social activities such as instant messaging or browsing Facebook profiles, gaming sessions are continuous and preclude multitasking. Hence, activity levels correlate to the times of day that gamers are not encumbered with the distractions of work or school.

6.2 Declared Relationships

Figure 6.2 plots the degree distributions of the players on the server. The Steam Community degree distribution is based off the entire friends list of players, while the Server Friends degree distribution is based off the subset of a player's friends that also played on the server. The interaction distribution portrays the number of interaction partners each player had.

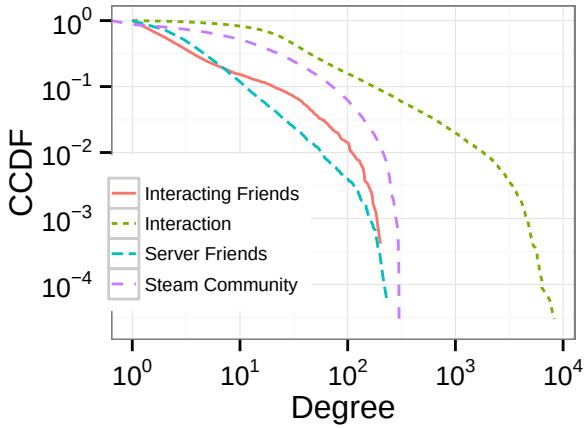

Figure 6.2: Static network degree distributions.

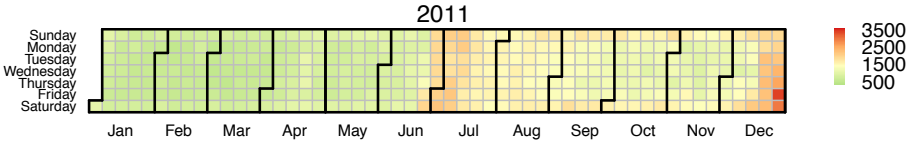

Figure 6.3: Heat map of friendships created between players on SH in 2011 per calendar day.

From Figure 6.2, we also see that players have many more interaction partners than they do declared friends, and tend to have fewer friends that play on the server than they do overall. Both of these results mirror real-life experiences that are not captured by interaction graphs or declarative OSNs alone: not everyone we interact with becomes a friend, and we interact with a subset of our friends depending on the venue.

What is not necessarily intuitive is the shape of the server friends curve. While we might expect the server friends curve to have the same shape (yet shifted) as the Steam friends curve, this is not the case. Instead, the shape mimics that of the interaction partners curve, giving us our first indication that interaction and declared friendships are related.

In Figure 6.3 we plot a heat map of the number of friendships created between SH players per calendar day for 2011. When viewing the plot, we see a clear demarcation between the first half of the year (ending in June) and the second half of the year (starting in July). For the first part of the year, friendship creation is steady, with a slight relative increase on weekends. In the second half of the year, however, we see a large uptick in friendship creations, followed by a gradual decline until December.

At face value, this seems like an odd result. However, it can be explained by a change in the pricing policy of TF2. In June of 2011 TF2 went Free-to-play. That is, while it had previously cost around $10 - $20 to purchase a copy of TF2, the game is now available to anyone for no cost. As might be expected, this lead to a sudden and dramatic increase in the number of global TF2 players.

The heat map thus shows a projection of the social effects of TF2 going free-to-play. The large influx of new players lead to a respectively large increase in the diversity of play partners. Simply put, there were suddenly a bunch more people to play with. This in turn leads to new found friendships. We see a corresponding reduction in the number of new friendships as the bump from the initial set of free-to-play adopters subsided.

It is important to note that there is a clear spike in friendships though. This can be explained by several things. First, TF2 going free-to-play was a very well publicized event. TF2 has a stellar reputation in the gaming world and thus many people likely rushed to try it out who had otherwise been unwilling to pay for it. Second, it is a big hint that there is a link between interactions and declared friendships, which we go into more detail about in the next section. Third, we can see that even though there is a drop off after the initial free-to-play announcement, that friendship creations are still occurring at a higher

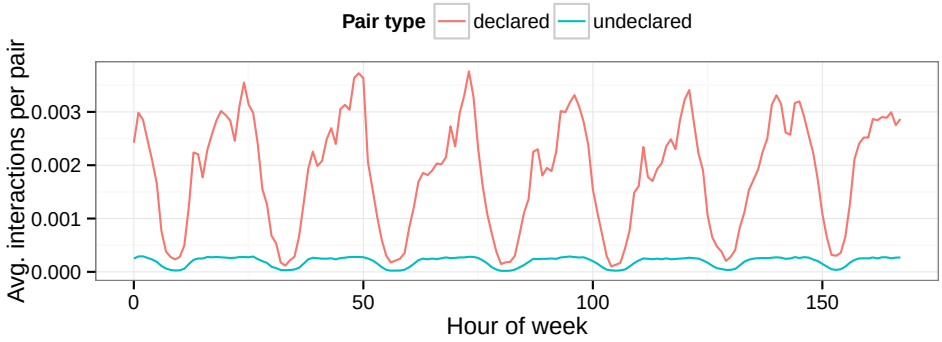

Figure 6.4: The average number of interactions per hour of the week for interacting pairs. Hour 0 is Sunday midnight (0:00)

than previous rate. This indicates that the free-to-play model continues to inject a diverse set of new players to the community.

6.3 Interactions and Declarations

Having established the activity level of the server community and the engagement of its players in the online social network of Steam, we next examine the relationship between the two. Ultimately, we find that declared pairs have *many* more interactions than undeclared pairs.

There is a striking difference between the interaction patterns of declared pairs and non-declared pairs of players, lending credence to the strength of declared relationships in an interaction-backed OSN like Steam Community. Figure 6.4 plots the average number of events per hour of the week for interacting pairs, differentiated by the existence of a declared relationship, across the entire span of our log files. Friends averaged several orders of magnitude more interactions than pairs without a declared friendship.

One unexpected finding is that the peaks in Figure 6.4 for declared pairs are during the week. This is in contrast to Figure 6.1 which shows higher levels of activity during the weekends. One possible explanation for this difference is that the server population skews towards "regulars" during the week and includes more "randoms" during the weekends. This hypothesis fits well with the concept of an interaction-backed OSN: regulars are more likely to interact more, and their interactions are more likely to spawn declared relationships in the OSN.

While it is clear that declared friends have significantly more interactions than non-declared friends, it is still unclear how direct an impact interactions have with friendship declaration. To begin answering this, we examine the number of interactions between pairs of players who became friends during the span of our log files. We thus look at the number of interactions between such pairs for each of the three weeks prior to and after the declared relationship forms.

In Figure 6.5 we compare the three weeks before the relationship forms, while Figure 6.6 compares the three weeks after the relationship forms. It is immediately apparent that friendship creation is preceded by a significant spike in interaction (Figure 6.5). More interestingly, we see in Figure 6.6 that once a relationship forms the volume of interaction begins to decline. Further, the rate of change in the volume of interactions is smaller for the three weeks after the relationship is formed than it is for the three weeks prior. This is an important finding for a few reasons. First, it provides solid evidence that relationships in Steam Community are interaction backed. I.e., changes in the volume of interaction between two gamers are a predictor for whether a friendship will form. Next, it highlights a similarity between interaction backed OSNs like Steam Community and more "traditional" OSNs like Facebook: in both cases, interactions begin to subside very soon after

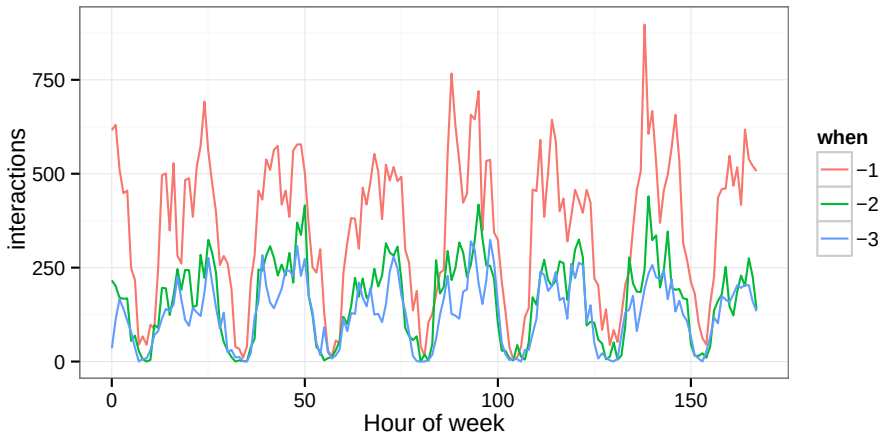

Figure 6.5: The number of interactions per pair for each of the three weeks prior to a declared relationship forming.

the relationship is declared. Third, it indicates that the intensity of a relationship is at its peak *prior* to the relationship being made official.

6.4 Summary

In this chapter we presented an analysis of the social behavior of gamers on a community owned and operated game server. We found diurnal patterns of play, essentially an inverse of the normal American work week. We used several networks induced from the in-game behavior of over 30 thousand players.and found differences in those players that were friends but did not interact, those players who interacted but were not friends, and players who both interacted and were friends. Next, we saw how an influx of new players (due to a change in the business model for TF2) resulted in a spike of new friendships. Finally, we showed interactions between declared friends dominated those between non-declared friends and that friendship creation was proceeded by a sharp increase in the volume of interactions which then slowly begins to decay.

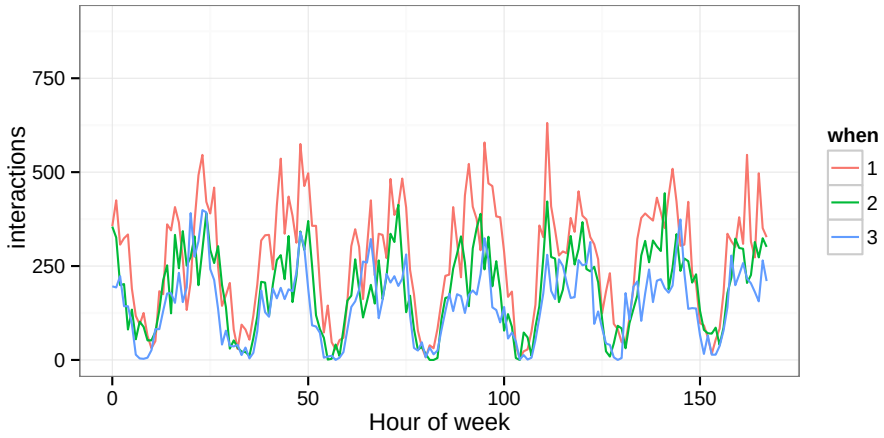

Figure 6.6: The number of interactions per pair for each of the three weeks after a declare relationship forms.

Chapter 7: Cheating in Online Games as a Contagion

[1]In this chapter we explore the mechanism by which cheating behavior spreads throughout the network of gamers. We identify several research questions, initially examined in [BKS+14, BSK+12]:

- Research Question 1: Is there evidence for a contagion process at play?

Literature from psychology indicates that cheating and unethical behavior spreads from individual to individual. Although positive results have been shown in controlled lab experiments, our large scale dataset can provide empirical evidence of such a mechanism. Using longitudinal data, we show that future cheating behavior is predicted by how many cheater friends one has, as well as visualizing the spread of behavior through the network over time.

- Research Question 2: How does positioning in a gaming network affect the processes described by accepted models of contagion?

Contagion has been studied quite extensively in a variety of fields. From this research, several models for contagious processes have been proposed. By performing a simulation

[1]Much of the work in this chapter was first published in Proceedings of the 21st International Conference on World Wide Web, 2012 [BSK+12] and ACM Transactions on Internet Technology, 2014 [BKS+14]Permission is included in Appendix B.

using data from our empirical networks, we are able to measure the effects that a cheater can have on epidemic spread. Our simulations show that the most engaged users are able to propagate a cheating epidemic much easier and faster than empirically observed cheaters or randomly selected seeds.

7.1 Evidence for Contagion

Having seen that there are clear differences in the social networks of cheaters and non-cheaters, and that the declared social relationships in Steam Community are backed by in-game interactions, we now analyze the hypothesis that this behavior is a *contagion*. That is, the reason we see such striking differences in the neighborhood compositions of cheaters and non-cheaters is that the cheating behavior spreads between individuals like a disease.

7.1.1 Potential for Influence

The position of a node in the social network affects the potential influence the node has on close or even remote parts of the network. For example, a high degree centrality node—one with many direct neighbors—can directly influence more nodes than a low degree centrality node. High betweenness centrality nodes, on the other hand, mediate the traffic along the shortest paths between many pairs of nodes, thus having influence on remote parts of the network. A high betweenness centrality cheater, for example, could facilitate the propagation of cheats and other deviant behavior to distant parts of the gamers network.

To understand the positional importance of the cheaters in the network, we study their potential for influence in the Steam Community by computing their degree centrality and node betweenness centrality. The degree centrality is simply the degree of the node in the network, and is thus a local metric. Betweenness centrality, however, is a global graph measure and consequently computationally expensive, requiring the calculation of the shortest paths between all pairs of nodes in the network. Due to the scale of our graph, we approximate betweenness centrality using κ*–path centrality*, a betweenness approximation method proposed in [KAS+12].

We observe a high correlation of 0.9731 between degree and betweenness centrality scores of the gamers. This high correlation remains consistent when we differentiate on the player type: 0.9817 for cheaters, and 0.9726 for non-cheaters. Consequently, if a player has many friends in the Steam Community network, (that is, high degree centrality), not only can she influence many players directly, but she can also mediate the information flow between remote players due to her likely high betweenness centrality. This result is consistent with theories of social influence [CF07].

We focus only on the most central players in the network and study how many of them are cheaters. Table 7.1 demonstrates that cheaters are under-represented among the most central players, despite the fact that they have about the same degree distribution as the fair players, as shown earlier in Figure 5.2. Over 7% of the entire player population in our dataset are cheaters, but they make up less than 7% of the top 1% most central players, and are not adequately represented until we consider the top 5% to top 10% most central players. Earlier results from Section 5.3 provide an explanation for this. There seem to be social mechanisms that retard the growth of cheaters' social neighborhoods which could be preventing them from entering the top 1% central players in the social network.

Table 7.1: Percentage of cheaters found in top-N% of high degree centrality (DC) and betweenness centrality (BC) users in the Steam Community.

Top-N%	0.1	0.5	1.0	5.0	10.0
DC	3.25	4.46	5.11	7.06	8.20
BC	5.16	5.95	6.35	7.86	8.58

Surprisingly, this mechanism also limits their opportunity to influence remote parts of the network, as they tend not to be on central betweenness positions.

While remote influence is unlikely, is there any direct social influence in the propagation of cheating in the network? Figures 5.4, and 5.5 seem to suggest so, and thus we hypothesize that the friends of known cheaters are at risk of becoming cheaters themselves.

7.1.2 How Does Cheating Propagate over Time?

Based on the observations from Figures 5.4, and 5.5, we hypothesize that the friends of known cheaters are at risk of becoming cheaters themselves. To test this hypothesis, we first explored whether cheaters discovered during a given time interval were more likely to be friends of previously discovered cheaters than to be friends with non-cheaters using the static Steam Community data set and ban dates. Again, we stress that banned dates must be treated as "on or before" as opposed to exact timestamps. To mitigate the effects of this uncertainty, we chose to examine cheaters discovered over 180-day long intervals.

We begin by assuming that all users in the Steam Community friendship network are non-cheaters. We then initialize the network by marking the 94,522 users found to have a VAC ban on or before December 29, 2009 (i.e., the earliest date retrieved from vacbanned.com). For the first 180-day interval (between December 30, 2009 and June 28, 2010), 34,681

players were found to have a VAC ban. For these users, we calculated and plotted the number and fraction of their cheater friends (i.e., from the 94,522 cheaters found previously). We repeat these steps for another 3 time intervals, with 19,294, 571,975, and 43,465 cheaters found in each. The third interval (starting at December 26, 2010) contains the bulk of cheaters, since their VAC ban was first observed by our initial crawl (and not from `vacbanned.com`). However, as shown next, the differentiation between cheaters and non-cheaters holds true for all intervals. In addition to a best effort approximation of the timestamp of the VAC bans, the data constructed this way has another caveat: the social network is from our March/April 2011 crawl, but we show in Section 5.3 that the network is quite dynamic. We verified that despite the change in number of friends over time, the trend is preserved: we recalculated the fraction of cheaters in the 1 hop neighborhoods of users based on the state of their relationships as determined by our October 2011 re-crawl and we found the non-cheaters CDF to dominate the cheaters CDF as in Figure 5.4.

Figures 7.1 and 7.2 plot the results of our experiments. Each subplot represents only the cheaters discovered during the corresponding time interval, and an equal number of randomly sampled non-cheaters (statistical significance confirmed by KS test at 5% significance level, $p < 0.01$ in all cases, and $D = 0.215$, 0.2172, 0.1963, and 0.2911 for the fraction of cheater friends distributions of each interval, respectively). From the plots we see evidence that users with both a higher absolute number of cheater friends, as well as those with proportionally more cheater friends are more likely to become cheaters themselves.

To understand if the number of cheater friends has any predictive power on the state of a player, we used the framework developed by Backstrom et al. [BHKL06]. We first created a dataset of fair players with at least one cheater friend at the time of our first crawl. We then labeled the players from this set who were marked as cheaters by the time of

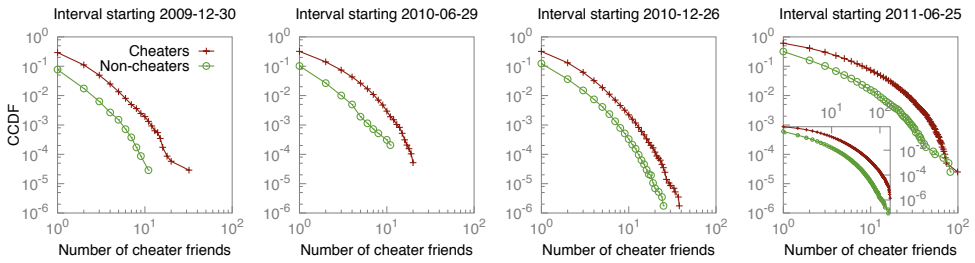

Figure 7.1: CCDF of the number of cheater friends of newly discovered cheaters and a random sample of non-cheaters over four 180-day time intervals.

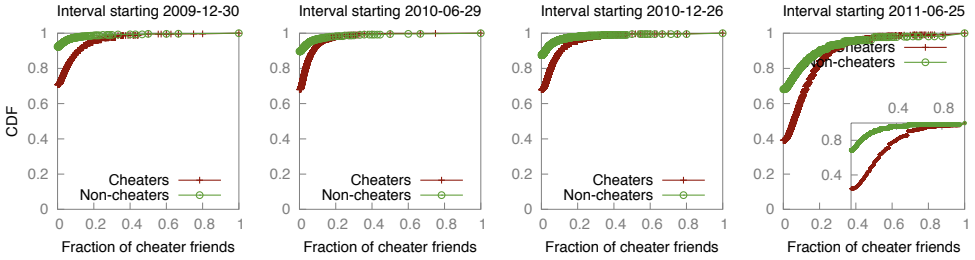

Figure 7.2: CDF of the fraction of cheater friends of newly discovered cheaters and a random sample of non-cheaters over four 180-day time intervals.

our second crawl. We consider the new cheaters as joining a group in the sense of the Backstrom study to estimate the probability of a fair player becoming a cheater. We then applied the decision tree technique, using the number of cheater friends as the single feature, and achieved an area under the curve (AUC) of .61. The AUC can be thought of the probability of the classifier ranking a cheater higher than a non-cheater for any random cheater/non-cheater pair. For our purposes, it shows that the number of cheater friends is a good predictor of becoming a cheater. With perfect knowledge, that is, if every single cheater was marked in our dataset by a perfect and timely cheating detection system, we would expect this value to be higher. (In comparison, Backstrom et al. show a AUC value of .6456 for the DBLP communities and .69244 for LiveJournal.)

While these results show that the number of cheater friends has predictive power for the transition of a player from fair to cheater, it is difficult to ascertain the specifics of how

the behavior propagates. Specifically, it is difficult to distinguish between homophily and contagion by simply observing the network properties and without strong assumptions about the social process [AMS09, ST11]. However, an individual's propensity towards unethical behavior, and cheating in particular, has been shown to be dependent on social norms, including the saliency of the behavior, and whether or not the behavior is observed from in- or out-group members [GAA09]. Our large-scale findings further corroborate this theory and suggest that cheating behavior is not preponderantly dependent on a personal strategy that takes into account player-local information only, but spreads through a contagion process.

A higher resolution data set, however, would provide stronger evidence as well as a deeper understanding of the speed of the cheating contagion. We thus begin by comparing the number of cheater friends for two groups of monitored users: non-cheaters and newly-labeled cheaters. Frequency distributions for the number of cheater friends for the monitored users (the number when banned for cheaters, and the number at the end of the observation period for control users) are plotted in Figures 7.3 and 7.4, respectively, with descriptive statistics listed in Table 7.2. Cheaters are *considerably* more likely to have multiple cheater friends at the time of the VAC ban application than our control group, and the samples are indeed drawn from different distributions ($D = 0.444$, $p_{ks} < 0.01$, $T = 36.80971$, $p_{perm} < 0.01$).

We built a logistic regression model using a single feature: the number of cheater friends at the time of the ban. Although there are many suitable classifiers, we next used a logistic regression as it is a well understood method suitable for binary predictions. The classifier built from this high-resolution dataset achieved an area under the receiver operating

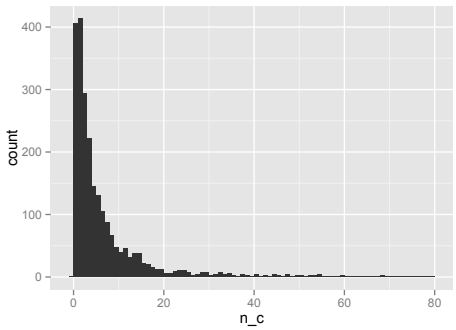

Figure 7.3: Frequency distribution of the number of cheater friends for newly discovered cheaters.

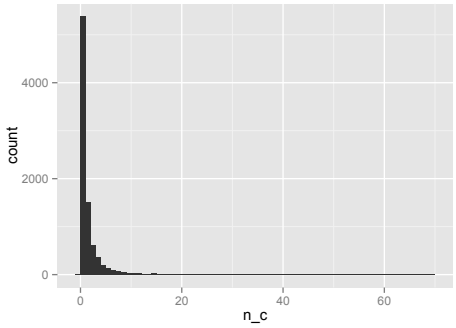

Figure 7.4: Frequency distribution of the number of cheater friends for a control group of non-cheaters.

characteristic curve (AUC) of 0.78, significantly higher than 0.61 which we achieved in using the static data set.

Table 7.2: Descriptive statistics for cheater friend distributions of the control group and new cheaters in the ban history dataset.

	n	mean	sd	median	min	max	se
Control	8,712	1.25	3.23	0.00	0.00	69.00	0.03
Cheaters	2,337	5.97	9.31	3.00	0.00	79.00	0.19

7.1.3 Hazard Analysis

We can view the adoption of behavior as a form of *survival*. Simply put, if the behavior is adopted by an individual, said individual can is said to have "perished", and if it is not adopted, the individual has "survived." Constructing the problem in this manner allows us to exploit the well studied area of survival analysis to model contagious effects of cheating. Specifically, we use the Cox proportional hazard model to assess the effects of predictors on the likelihood of becoming a cheater.

In a nutshell, a hazard function provides the instantaneous rate at which the event of interest occurs, given that it has not yet occurred:

$$h(t) = \lim_{t \to \infty} \frac{Pr[t < T \leq t + \Delta t) | T \geq t]}{\Delta t}$$

The proportional hazard model assumes there is a baseline risk that is altered by predictors that characterize a subject:

$$h(t|x) = h_i(t) = h_0(t) \exp(\beta_1 x_{i1} + \beta_2 x_{ik} + \cdots + \beta_k x_{ik}),$$

where $h(t)$ is the underlying hazard function and can be left *completely unspecified* when only the relative hazards of a subject with different x profiles are of interest. One technical detail is important. Parameter estimates are obtained by maximizing the partial likelihood rather than the maximum likelihood but this is valid only when there a not ties in the data set, that is, no two subjects have the same event time. If there are ties (as is the

case for our data set), the true partial likelihood involves permutations and can be time-consuming to compute but two approximations can be used.

To apply to cheating, we define an observation window for some start time to some end time. All non-cheaters at the start time are at risk of becoming cheaters. If they become cheaters the dependent variable is scored 1 otherwise it stays 0, the time since beginning of observation that they become cheaters is noted as well as the number of friends at the time who were also cheaters. Thus, we can view each case as a gamer characterized by:

1. An indicator variable (0 or 1) for whether or not player adopted cheating behavior (i.e., received a VAC ban) at some point during the duration of the observation period.

2. The time since start of observation that the adoption occurred. If the behavior was not adopted by the end of the observation period we *right-censor* the case by setting the time to the last time point of observation.

3. The number of signals received at the time of adoption, or received by the end of the observation period if adoption did not occur.

We performed the same set of experiments on both the low and high resolution data sets. We first built data files characterizing each subject as described above. We then compared the effects a number of signals (i.e., the number of cheater friends) had on the odds for a case becoming a cheater. To compare the effects of having just one friends to having no friends we selected the cases where the k_c variable was 0 or 1. To compare the effects of having an additional cheater friend to having just a single cheater friend, we selected the cases where k_c was 1 or 2. We continued to calculate the change in odds for additional

cheater friends as above until the analysis was unable to reject the null hypothesis at the 5% significance level.

The regression analysis calculates the hazard ratio: the amount by which the odds of user becoming a cheater increase given one additional cheater among his friends. The percentage increase is simply the computed hazard ratio -1.00. Thus, a hazard ratio of 1.00 means the additional cheater friend had no effect on the odds. The hazard ratio computation also produces a 95% confidence interval, and so we can reject the null hypothesis only if the confidence interval does not contain 1.00.

The estimated survival functions for are plotted in Figure 7.5. After 30 days, around 10% of the 2,840 cases remained unbanned, and the function appears linear. However, the effects of additional cheater friends begin to make themselves apparent. The survival odds continue to decrease as the number of cheater friends increases: less than 50% of players with 5 or 6 cheater friends remain non-cheaters at the end of the observation period. Clearly, the majority of users remain unbanned at the end of our observational period, but, this is somewhat expected considering that cheaters were observed to make up only 7% of the social network.

The results from each set of our experiments appear in Table 7.3, and we visualize these results in Figure 7.6 The results indicate that cheating behavior spreads via a contagion mechanism. Although the increase in odds is drastically greater when moving from 0 to 1 friends, we see a significant increase up through 4 cheater friends for the Spring 2011 dataset, and up through 5 cheater friends for the high resolution dataset. Having 1 cheater friend results in a 138.5% and 231.3% greater chance to become a cheater in the respective datasets, compared to an increase of 25.8% when moving from 3 to 4 cheater friends in the Spring 2011 dataset, and 2.52% when moving from 4 to 5 cheater friends for the August

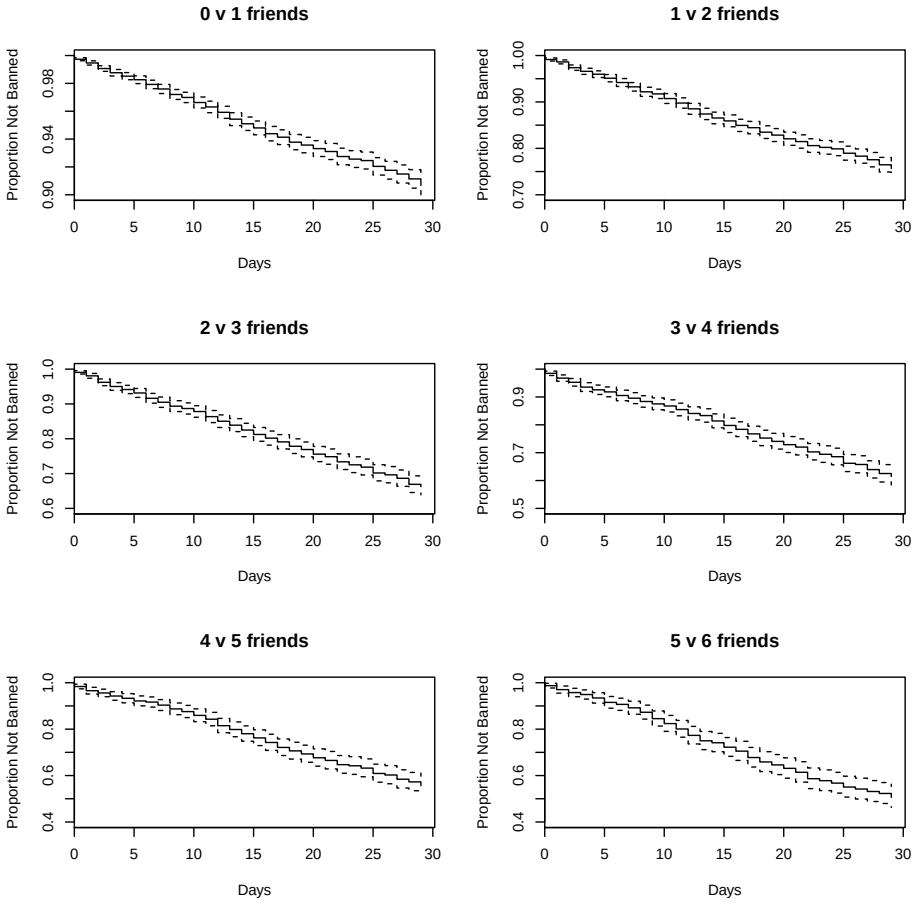

Figure 7.5: The estimated survival functions for the Cox regression from the August 2012 dataset. The dashed lines are the 95% confidence interval surrounding the estimated functions. The decreased survival rates as the number of cheater friends increases is quite apparent. With 0 or 1 cheater friends, the survival odds are about 90%, but with 5 or 6 cheater friends the survival odds fall to less than 50%.

Table 7.3: Cox proportional hazard model analysis for both datasets we studied.

Dataset	Signals	Cases	Events	Hazard	95% CI
March 2011	0 vs 1	76,760	4,060	2.385	(2.238, 2.541)
	1 vs 2	22,283	2,419	1.424	(1.312, 1.547)
	2 vs 3	10,305	1,432	1.067	(0.959, 1.188)
	3 vs 4	6,004	948	1.258	(1.106, 1.430)
	4 vs 5	3,864	684	0.965	(0.828, 1.224)
August 2012	0 vs 1	7,716	817	3.313	(2.889, 3.8)
	1 vs 2	2,840	702	1.584	(1.363, 1.841)
	2 vs 3	1,498	507	1.199	(1.005, 1.429)
	3 vs 4	927	358	1.143	(0.9248, 1.412)
	4 vs 5	607	270	1.252	(0.9861, 1.59)
	5 vs 6	471	233	1.077	(0.8317, 1.394)

2012 dataset. We note that both datasets see a statistically insignificant increase in odds followed by a resumption of statistically significant odds increases (2 to 3 friends from the March dataset, and 3 to 4 friends from the August dataset).

The hazard analysis provides clear, statistically significant evidence of contagion effects at play. Specifically, the odds of becoming a cheater greatly increase with the more cheater friends you have, up until about 3 friends.

7.1.4 The Diffusion of Cheating Behavior

If the theory that cheating behavior is a social phenomenon that can be tracked via a corresponding social network holds true, then we should see newly discovered cheaters connected to previously discovered cheaters, and thus likely to form components when compared to the randomly selected control group.

We began by looking at the components formed by all monitored users and their social neighborhoods from the high resolution data set. The relative component size distributions

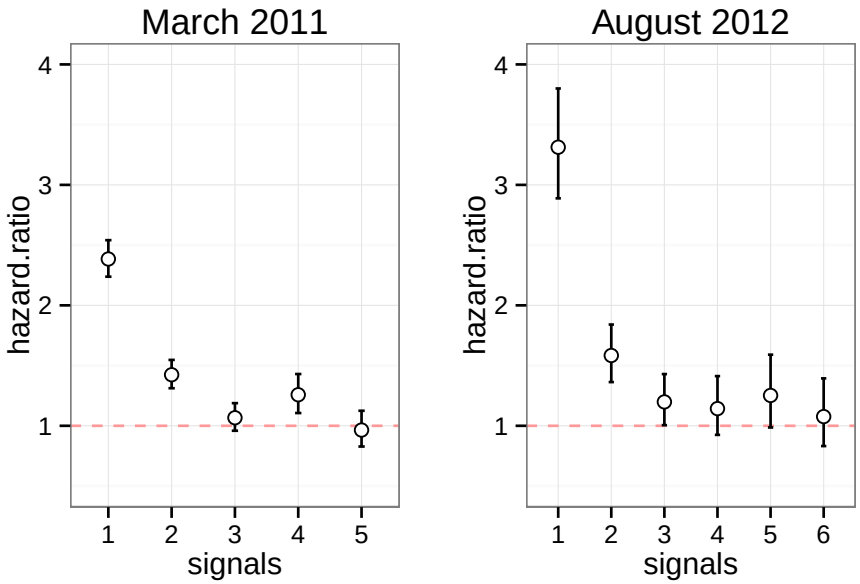

Figure 7.6: Hazard ratios for the 2011 and 2012 datasets. The hazard ratio represents the increase in odds to become a cheater when going from $x - 1$ to x cheater friends. Error bars represent the 95% confidence interval. As long as the confidence interval does not contain 1.00, the increase in odds is significant at the 5% level. For the March 2011 dataset, we see a significant increase up through 4 cheater friends. For the August 2012 dataset, the hazard ratio is significant up through the increase from 4 to 5 cheater friends.

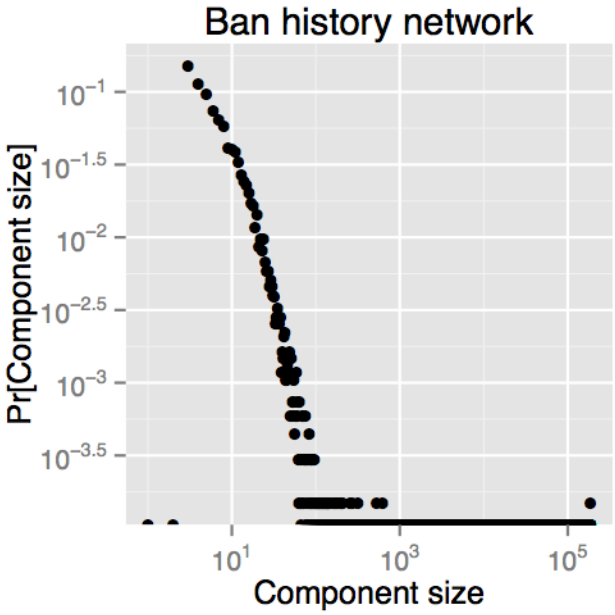

Figure 7.7: Relative component size distribution for the ban history network. There are 6,692 components, and the largest component has 186,360 members.

for this network is plotted in Figure 7.7. The network had 6,692 components, with the largest containing 186,360 nodes and 200,236 edges. The existence of a large connected component forming from the 1-hop neighborhoods of $\simeq$10,000 users indicates that our monitored users are not randomly dispersed throughout Steam Community, and additional evidence that cheaters are indeed clustering together.

Although the component distributions provide clear indication of cheaters clustering together, they do not necessarily address the issue of causality. To visualize the propagation of the cheating ban in the network (as a way of identifying causality), we plot in Figure 7.8 the progression of VAC bans in the 10 largest components composed entirely of monitored users between August 6th and September 5th 2012, divided into 6 intervals. We take network dynamics into account by coloring nodes and edges differently depending on the state of the network during the interval. Nodes are colored red if they were VAC

banned by the end of the interval, and green otherwise. Edges are colored black if they existed prior to the end of the interval. Gray edges have not been formed by the end of the interval, but will exist at some point in the future. Finally, if an edge has been deleted (i.e., the friendship was dissolved) prior to the beginning of an interval it is colored orange.

Figure 7.8 shows evidence of spread. Most newly flagged cheaters are soon followed by their neighbors, resulting in clusters of cheaters appearing as time progresses. This discovery is the first indication of how rapid the contagion spreads through the network: out of the 279 cheaters at the end of Interval 6, only 28 (10%) were cheaters a month before.

A clarification is necessary at this point: we do not believe that the VAC ban is what inspires friends of newly-marked cheaters to cheat and thus become cheaters themselves; about 90% of the ties between eventual cheaters existed well before either were cheaters. Of these, 75% were created more than 10 days prior to one of the friends being labeled a cheater, as seen in Figure 7.9. In fact, the delayed application of the VAC ban obscures the exact timing of the contagion somewhat, and also allows friends to see the gains from cheating without the public penalty of the cheating label. Moreover, Figures 5.9 and 5.11 show that non-cheaters sever their relationships with newly discovered cheaters, perhaps in an attempt to dissociate themselves from the dishonest behavior. What we believe happens is that the cheating behavior—implemented via sharing of cheat codes, for example—spreads along social ties prior to the cheating behavior being exhibited by the initial cheater in the relationship. When the VAC bans are applied some unknown time interval later, they reveal the time causality pattern of a social contagion.

There are two take-away lessons from this finding. First, it hints at possible improvements for the detection of cheaters. For example, the costs of detection efforts could be reduced by focusing on users who are in a relationship with a recently detected cheater. When

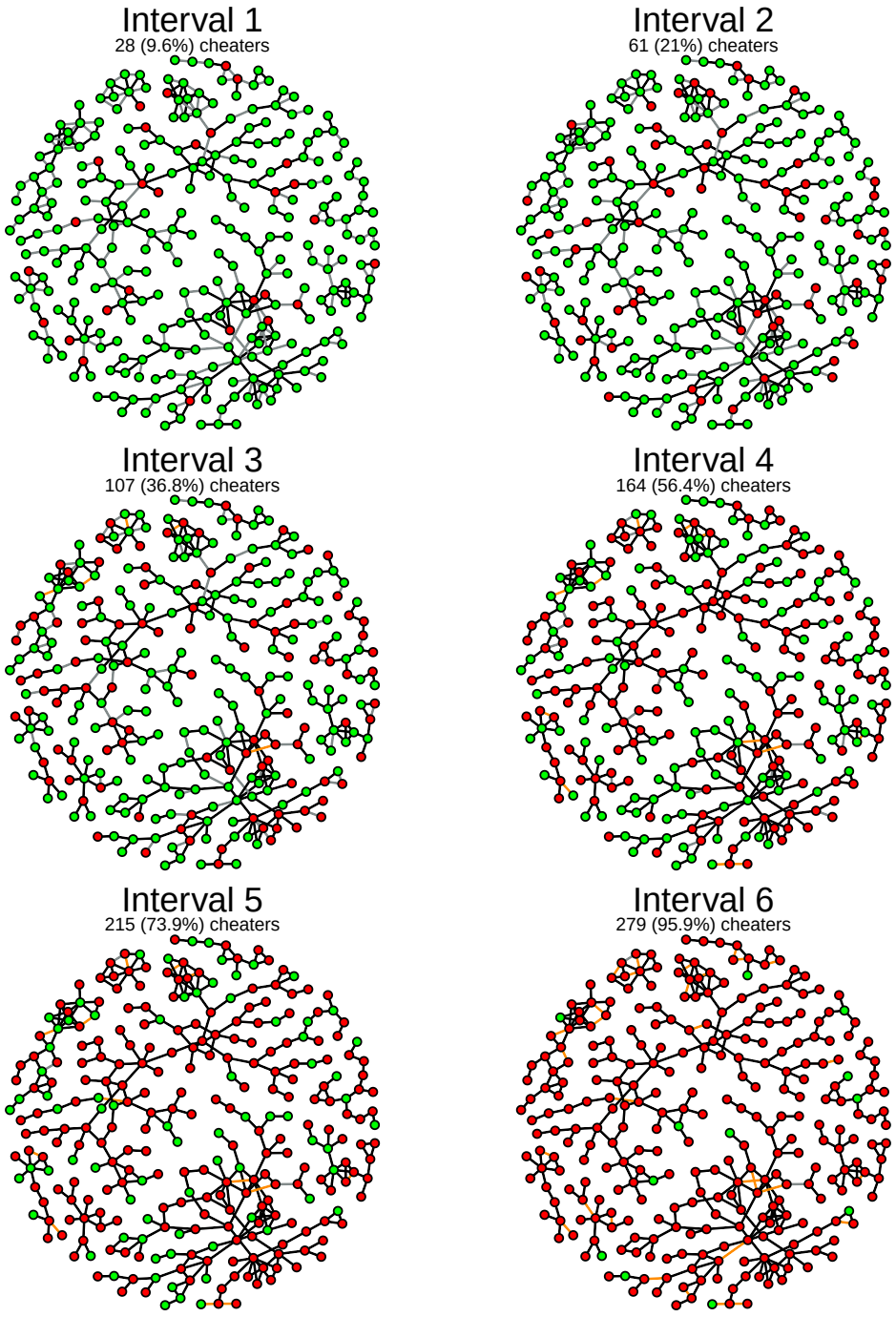

Figure 7.8: The spread of cheating behavior in the 10 largest connected components of monitored users over a 1-month period, split into six intervals. If a non-cheater transitioned to cheater prior to the end of an interval, the node is colored red, and green otherwise. Gray edges will form in a future interval, black edges have formed by the end of an interval, and orange edges have been deleted prior to the beginning of an interval.

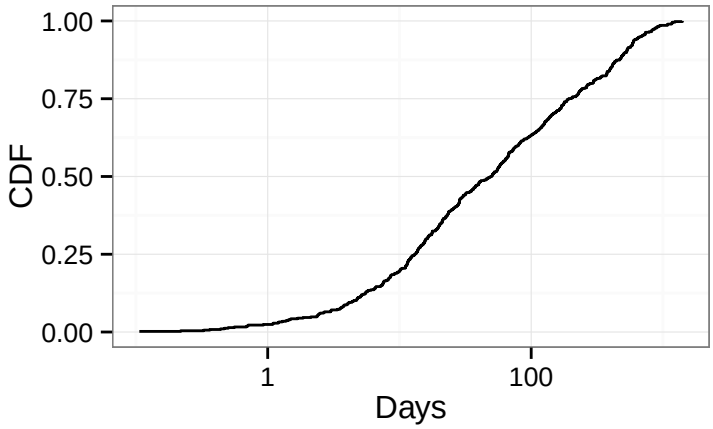

Figure 7.9: CDF of the age of a relationship between pairs of eventual cheaters prior to either of the friends becoming a cheater. I.e., how old the relationship was before either friend cheated. 75% of the ties existed more than 10 days prior to the application of the cheating flag, about 50% had existed for more 50 days, and over 25% existed for more than 6 months. This indicates that the cheating behavior itself is not the catalyst for friendship formation, but rather that existing friendships allow the cheating behavior to spread.

used in conjunction with crowd-sourcing techniques to discover "spontaneous" cheaters (e.g., allowing users to report suspicious behavior), leveraging the structure of the social network would drastically reduce the set of users deemed at risk. The reduction in at-risk users would allow more computational expensive detection techniques to be deployed. This filtering would also make further use of crowd-sourcing possible, for example, by allowing other gamers to make judgments based on recorded gameplay sessions of at-risk users.

Second, knowledge of an epidemic process can also help *prevent* the adoption of cheating behavior in the first place. One solution could be to quarantine recently discovered cheaters (especially before the VAC ban becomes publicly visible), to limit the time of exposure to the "infectious" gamer. While an intuitive solution might be to remove cheaters from the service completely, this does not fit the business model of many gaming platforms. Even more, cheaters are accepted by the communities of other games that they did

not cheat in. For this reason, we suggest a *temporary* quarantine of new cheaters that prevents direct, out-of-game communication via chat for a limited time. This may retard the cheating contagion at practically no cost to the service provider by, for example, allowing the delayed VAC ban to be applied before the newly discovered cheater is able to share his cheating tools via Steam Community's communication channels.

A more intriguing solution might be the development of interventions targeting at-risk users. In this case at-risk users could be actively targeted with a "vaccination" strategy, for example, personalized pop-up notifications reminding them of the consequences of cheating. Targeted interventions have been shown to effectively contain epidemic outbreaks, even with a lag between the identification of infected individuals and administration of the vaccine [LNX+05].

7.2 Data Driven Simulation of Contagious Behavior

Diffusion and contagion processes have been well explored in both the epidemiology as well as sociology literature [DW05, CEM07, ST11, AMS09]. Generally speaking, contagion models call for nodes to be in one of several states. *Infected* nodes have succumbed to the contagion, *susceptible* nodes are vulnerable to the contagion, and *recovered* where infected individuals can either gain permanent immunity or die. In our case, we restrict node states to either susceptible or infected, and nodes remain in the infected after switching from susceptible (using epidemiology terminology, the recovery rate is 0).

Models can broadly be classified as either *independent interaction* or *threshold*. In independent interaction models, the infection probability is uniform and independent between contacts. A single contact with an infected node is sufficient to spread the infection. Thus,

there is no consideration for how contact with *multiple* infected individuals alters the odds of becoming infected.

When contact with multiple infectious individuals alters the probability of becoming infected, the process is often called a *complex* contagion. Complex contagion is often used to model social diffusion processes where odds of infection increase as more of a person's friends become infected. The threshold model takes this into account by requiring contact with a certain number of infected individuals. Once the threshold is met, infection can occur.

Centola et al. used a *fractional* threshold model to study complex contagion cascade dynamics in [CEM07]. A *cascade*, the process of the contagion spreading through the network is deemed successful if at least 90% of the nodes are infected by the time the cascade stalls. They propose an estimation for a critical threshold, $T_c = \frac{1}{\langle k \rangle}$ where $\langle k \rangle$ is the average degree in the network. By randomizing the network, they show that while the original network can support $T > T_c$, the permuted network (which loses social clustering properties), cannot reach a successful cascade until $T = \frac{1}{\langle k \rangle}$. Their findings indicate that social clustering in small world networks allows for the propagation of *complex* contagion. In other words, only *simple* contagion can propagate on the randomized network.

Dodds and Watts [DW05] introduced a generalized model for contagion. This model is able to produce epidemics that fall into several universal classes including independent interactions and complex contagion. The system is dosage based, whereby an infected node is able to provide a dose to an uninfected contact. Each uninfected node u has a dosing threshold $g(u)$, which if exceeded causes it to become infected. An additional parameter of a memory window M keeps tracks of doses over time. When calculating whether or not a dosing threshold has been exceeded, only doses received in the last M timesteps are

Table 7.4: Summary of model parameters.

p	Exposure probability.
M	Memory window of contacts and dosings.
$g(u)$	Dosing threshold distribution.
$f(u)$	Dosage size distribution.
$\theta_t(u)$	u's valid interaction partners at time t.
$\xi_t(u)$	Max. doses receivable at time t.
$\zeta_t(u)$	Max. contacts that can be made at time t.

Table 7.5: Parameters to reproduce common contagion processes.

Process	p	M	$g(u)$	$f(u)$	$\theta_t(u)$	$\xi_t(u)$	$\zeta_t(u)$
Independent interaction	-	-	-	-	V	1	1
Complex contagion	-	1	2	1	$neighborhood(u)$	$degree(u)$	$degree(u)$

considered. A dose size distribution $f(u)$ controls how big of a dose u will receive. Finally, p is the probability of becoming infected when coming into contact with an infected individual. Only exposed individuals are dosed.

7.2.1 Augmented Model for Contagion

We augment the Dodds and Watts model [DW05] with a few additional parameters. First, we introduce θ_t as the set of *eligible contacts* of individual u at time t. For example, when contagion spreads through the ties of a social network, $\theta_t(u) = neighborhood(u)$. In an interaction network, $\theta_t(u)$ = the set of nodes u interacted with at t. We also introduce $\zeta_t(u)$ as the maximum number of doses that u can receive at timestep t and $\xi_t(u)$ as the maximum number of individuals u can come into contact with at time t.

These additional parameters allow us to capture all elements of the generalized model, as well as *complex* contagion models which require exposure to *multiple* individuals. Ta-

ble 7.4 summarizes the model parameters, and Table 7.5 shows how common contagion processes can be reproduced with our model.

These additional parameters also allow us to extend the theoretical models to better suit the online gaming domain and our empirical datasets. $\theta_t(u)$ allows an individual u to infect many others at the same timestep. This is a departure from the model in [DW05], which restricts u to coming into contact with only one individual per step. The reasoning behind this change is due to the application our datsets are derived from: players come into contact with all the other players in a given match or with their connections in the social or interaction networks.

First, although we have seen evidence that suggests complex contagion might occur (e.g., the odds of a player becoming a cheater increase with the number of cheater friends he has), previous work on OSNs has indicated that declared social ties might not be representative of real relationships. Specifically, Facebook relationships have proven problematic for systems that rely on trust derived from the network structure [WBS+09]. Because of this concern, we first want to determine whether complex contagion propagation is even possible on the Steam Community OSN. Additionally, because Steam Community is an interaction-backed OSN (Chapter 6, structural properties that are invisible at the OSN level might provide answers into how the spread of cheating projects onto the declared network's structure.

Next, although static network analysis provide insight into high-level propagation dynamics, it only provides high-level aggregate knowledge. By making use of temporal data on interactions, we can examine how behavioral patterns can influence the network's susceptibility to infection. For the domain of gaming, this temporal information is highly relevant: while declared ties provide knowledge of the existence, and possibly some attributes de-

scribing a relationship, the addition of time-aware analysis captures how relationships wax and wane over time.

By blending an independent interaction and complex contagion model we can use real workloads representative of gameplay patterns over time. By manipulating M, we account for domain specific social behavior.

The exposure probability p has the same meaning as in the original model, but a gaming-specific interpretation. Unlike biological contagions, human beings conciously decide whether to adopt an unethical behavior (and thus become "infected"). However, just like biological contagions, humans (generally speaking) do not have control over whether they are *exposed* to cheating behavior. For the domain of cheating in video games, p captures the effects of blatant versus more discrete cheating behavior. In the real world, some cheats are quite evident (thus p is close to 1) while others are not.

7.2.2 Experiments

We implemented the model presented in Section 7.2.1 in a simulator that takes as input one of the empirical networks presented in Chapter 4. Each step of the simulation represents one discrete timestep t. At $t = 0$ we seed n_{seed} nodes as infected. The seeding process differs with the experiment and will be noted ahead.

On each step t, every node u randomly selects $\zeta_t(u)$ contacts $v \in \theta_t(u)$. One outcome of this is that it allows us to scale the simulation timesteps differently than the real interaction frequencies. In this case we are "compressing" the per-second resolution of the log files into discrete, domain relevant matches.

For each v that u contacts, with exposure probability p, u receives a dose $f(u)$, unless u has already exceeded its dosage count $\xi_t(u)$. A newly infected node can not infect any other node until the *next* timestep, which could be seen as an incubation period of 1.

As stated previously, we have two primary research objectives: (1) Understanding how cheating behavior spreads in our empirical networks under accepted models of contagion; and (2) How networks representing different perspectives of the same reality affect contagion.

To do this, we measure two metrics:

- ϕ_t which measures the size of a cascade at a given time t, reported as either a fraction or absolute value of infected nodes.
- ψ_T the final size of the cascade for a given threshold T, reported as a fraction of infected nodes.

The remainder of this section describes our experimental setup. The results are presented in Section 7.2.3.

Each of these networks allows us to examine different scenarios under which cheating behavior spreads. The server friends network models contagion over purely social relationships. The interation network, however, represents interactions that are not necessarily associated with a social relationship. The intersection of these two, the interacting friends network, is built from *strong* ties, allowing us to explore the effects of relationship strength. Finally, the co-presence network places real world restrictions on through whom and when the contagion can spread.

Table 7.6: Network statistics for the largest connected components of the static networks.

Network	nodes	edges	density	mean degree	clustering coeff.	cheaters
Interacting Friends	2,406	9,720	0.0034	8.1	0.20	30
Server Friends	16,682	46,791	0.00034	5.6	0.20	274
Interaction	33,523	1,768,377	0.0031	106.0	0.16	527

The construction of the server friends, interaction, and interaction friends networks is straight forward. There are 33,546 players on the server who are part of Steam Community, involved in over 1 million relationships. Of them, 22,099 have 50,522 friendships where both friends played on the server. Of these, 7,701 friends interacted on the server during our observations, forming 13,270 interactive pairs. The metrics of the largest connected components of the three networks are presented in Table 7.6. Figure 6.2 plots the degree distributions for the three largest connected components of our static networks. The degree distribution of the interacting friends network retains some of the curve that shows up in the full Steam Community network.

We describe the likelihood of nodes with similar degrees being connected via k_{nn}, the degree correlation, which for our undirected networks is the average degree of the neighbors of nodes with a given degree k. Figure 7.10 plots the k_{nn} distributions for each of our static networks. The server friends, and, to a lesser degree the interacting friends network, retain the characteristically strong degree correlation of an OSN [WSPZ12]. The interaction network does not. Instead, there is a *negative* correlation. This is due to the behavioral patterns of players and the maximum capacity of the server. Because players can only interact with $\sim$30 other players in a match, and the number of games played is a heavy tailed distribution, those that play many matches will necessarily come into contact with those who played relatively few matches *in addition* to other highly active players.

Figures 7.11 and 7.12, plot edge and node betweenness of our networks, respectively . High betweenness nodes can serve as bottlenecks restricting information exchanges be-

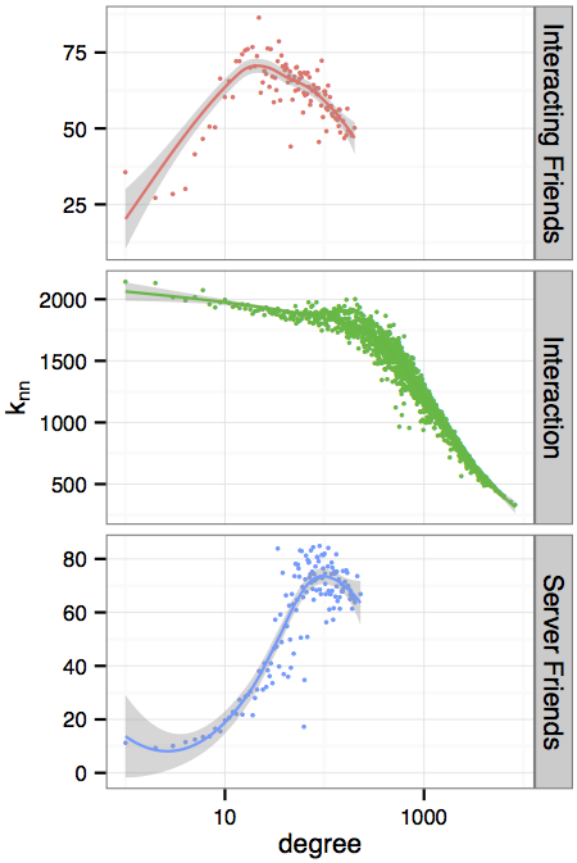

Figure 7.10: Average degree of neighbors (k_{nn}) as a function of degree for the three static networks.

tween distance nodes, but can also accelerate contagion by providing "shortcuts" along which the infection can reach new hosts. Figure 7.11 also includes the betweenness of the players appearing in the interacting friends network relative to their position in the interaction network. We see that the server friends and interacting friends networks have bi-modal distributions, with most nodes and edges having nearly no shortest paths flowing through them while a very small minority act as hubs. The implications of these network statistics on the contagion process will be discussed further in Section 7.2.3.

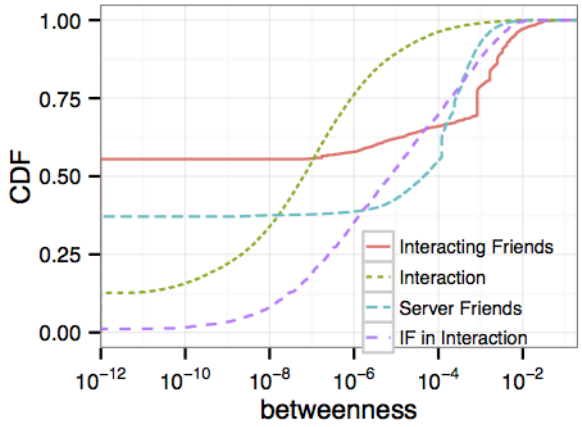

Figure 7.11: CDF of normalized node betweenness for the static networks and the interacting friends within the interaction network.

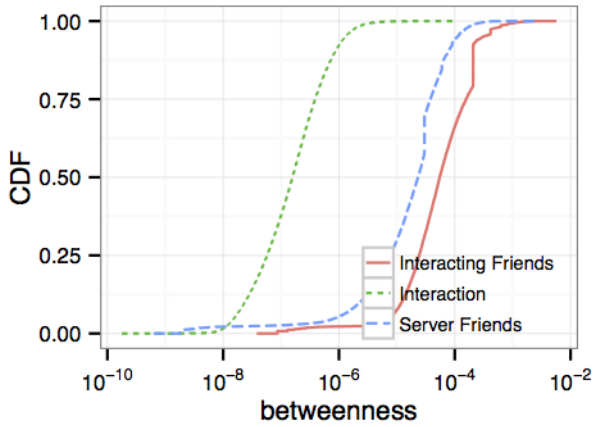

Figure 7.12: CDF of normalized edge betweenness for the static networks.

The presence network captures the co-presence of players over time. A user is deemed present during a match if the event game-join was recorded in the logs. Disconnects are measured using two event types: 1) player disconnected, and 2) end of log file. Player-disconnected events appear in the log whenever a player disconnects from the server. Since each match has its own logfile, log-file-end events delineate the matches.

There were 17,577 matches (i.e., log-file-end events). The number of players appearing in the log files is 40,663. However, we consider only those who joined matches (37,140 players) and discard those who never joined a match, i.e., had a server connect, but not a game join[2]

Formally, we define the *co-presence* network as an annotated graph $G(V, E, S)$ of vertices (V) and edges (E) where an edge $e(u, v) \in E$ iff players u and v were both present for at least one match together. Each edge is annotated with the series of matches the users had in common $s(u, v) \in S$.

To construct this network, consider an $m \times n$ matrix A, where m is the number of players and n is the number of matches, indexed according to an ascending temporal ordering. Let $A_{ik} = 1$ if player i was present for match k, and 0 otherwise. Now, we can see that

$$e(u, v) \in E \Leftrightarrow \sum_{k=1}^{n} A_{uk} A_{vk} > 0$$

Further, we see that

$$s(u, v) =< A_{uk_1} A_{vk_1}, ..., A_{uk_m} A_{vk_m} >$$

Note that $\sum s(u, v)$ is the number of matches that u and v were co-present for.

[2]Players first connect to the server, which then communicates with Steam to validate that the player is allowed to connect to that server.

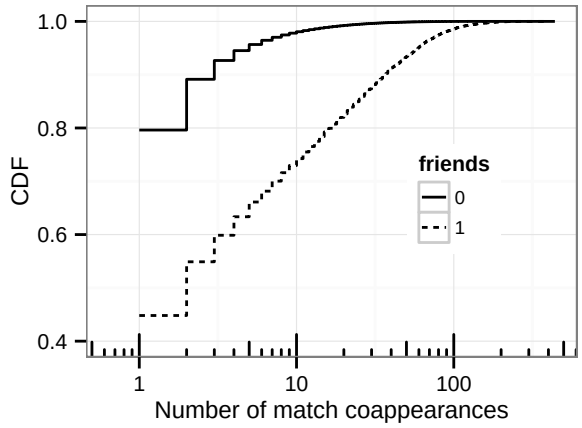

Figure 7.13: CDF of the number of matches pairs of players played together.

We use the link prediction model proposed by Mason and Clauset [MC13] (hereafter referred to as the FTW! model) to show that the co-presence network captures social relationships. This model is parameterized by the total number of mutual matches, the longest series of matches, as well as the number of series of matches of different lengths between pairs of players. As seen in Figures 7.13 and 7.14, declared friends play many more games with each other, and, they also play longer series of games together when compared to non-friends. Mason and Clauset achieved a remarkable area under the receiving operator characteristic curve (AUC) of 0.98 from survey derived ground truth. Using this model we are able to predict the presence of a declared relationship in our dataset with an AUC of 0.74 without any additional game-specific tuning.

7.2.2.1 Static Networks

This set of experiments uses three static networks: 1) the interaction network, 2) the server friends network, and 3) the interacting friends network. Although each of these networks

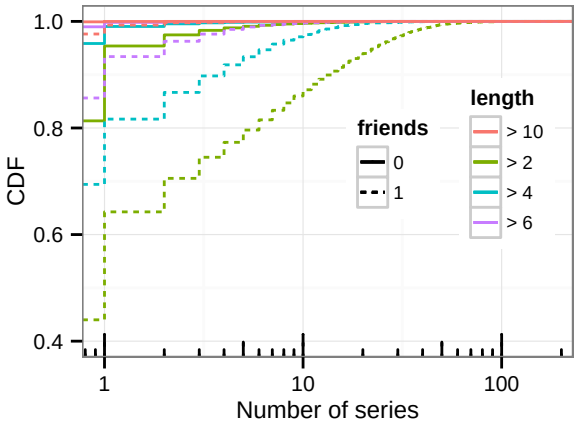

Figure 7.14: CDF of the number of series of varying length for both friends and non-friends.

has time information available, we look solely at empirical structural elements, and thus simulation timesteps are independent of real-world time.

We vary a uniform fractional threshold relative to a node's degree, $g(u) = T * deg(u)$, while keeping $p = 1.0$ and $f(u) = 1.0$. I.e., if $T = 0.24$, and node i has 8 neighbors, then $g(i) = 0.24 * 8 = 1.92$. We allow each node to contact all of its neighbors per timestep, and to receive a dose from each of them ($\xi(u) = \zeta(u) = degree(u)$). $\theta_t(u) = neighborhood(u)$ for all t. For each run of a simulation, a random node and its neighbors are selected as initial seeds. With $M = 1$, these experiments model a complex contagion [CEM07].

First, we want to examine whether the social clustering of the networks affects the propagation of contagion. To do this, we use a uniform distribution of fractional thresholds and compare ψ_T for each static network and its randomized version.

The randomized networks are constructed by applying the permutation algorithm to each network. In short, each edge has its end points swapped with another randomly selected edge. This maintains the degree distribution while changing the topology of the network.

By permuting the network, we decrease social clustering. In effect, we are reducing the overlap between neighborhoods, and connecting otherwise socially distant users to each other. In theory, a permuted network will not be susceptible to complex contagion, or at least have a reduced ability to support it [CEM07].

As mentioned previously, the critical threshold, $T_c = \frac{1}{\langle k \rangle}$ has been posited as an adequate estimate for the critical threshold of simple contagion, with $T_c > \frac{1}{\langle k \rangle}$ for complex contagion.

As mentioned in Section 7.2.1, by varying exposure probability we can simulate the inherently clandestine nature of cheating. We choose a uniform exposure $p \in \{1.0, 0.75, 0.5\}$ to provide baseline measurements for the effects of how fair players observe an act of cheating in the real world.

7.2.2.2 Dynamic Network

This set of experiments uses the co-presence network. For these experiments, only a subset of nodes are eligible for contact at each step t: the players that played during game t, independent of the existence of declared Steam relationships or recorded in-game interactions. For example, a player might observe another player cheating, but not actually have a direct (as measurable via game logs) interaction. Observing cheating behavior, even from a stranger, has been shown to affect the odds of cheating in controlled experiments [GAA09, Ari12].

We use a uniform linear threshold, $g(u) = T$, in these experiments as opposed to the fractional threshold in the previous section, while the dosage size distribution remains

$f(u) = 1$. We let θ_t be the set of players that played in match t, and $\xi_t(u) = \zeta_t(u) = |\theta_t(u)|$. Finally,

$$\theta_t(u) = \begin{cases} \theta_t & \text{if } A_{ut} = 1 \\ \emptyset & \text{if } A_{ut} = 0 \end{cases}$$

.

We are interested in examining the relationship between the following parameters: 1) the threshold, 2) the memory window, 3) the exposure probability, and 4) seed selection.

As noted in the FTW! model, the number of series of matches of certain lengths is a good predictor for friendship. If we set $M > 1$, then, as seen in Figure 7.13, it will provide friends more opportunity to spread contagion to each other then non-friends do. If we set $p = 1.0$, then being co-present with an infected once will guarantee a dose. If $M = 1$, then a player must play with T infected players during a single match.

But, friends are more likely to play *multiple* matches in a row with each other. If we set $M > 1$, it will give friends the ability to reinforce cheating behavior. Specifically, we test $M \in 1, 3, 5, 7$. By using θ_t in combination with the appropriate selections of M, we are thus projecting the effects of the social structure onto the behavioral network.

To examine the relationship between thresholds, we use a set of discrete, integer values for the threshold, where a node becomes infected if the number of doses received exceeds the given value. This differs from the fractional thresholds used in the static experiments in that it is not relative to the number of contacts a player has. However, there is a clear correspondence between the discrete threshold and a fractional threshold, because the vast majority of games have an average number of players around 34.

Some cheats are quite evident. For example, a "spinbot" is a type of cheat that automatically aims for the player by continuously rotating his character at super human speed. When the cheater sees another player he can "fire"; the bot stops spinning and automatically aims at the other player. Once the cheater stops firing, the bot resumes spinning. From the perspective of the cheater the world is constantly spinning, and from the perspective of other players, the cheater's character is constantly spinning, except when he is shooting with perfect aim. In real gameplay, spinbots are obvious.

Other cheats are less obvious. "Wallhacks" allow a cheater to see through walls and often display otherwise hidden information such as outlining enemy's characters and showing their remaining health or ammunition. When used by a savvy cheater, a wallhack is nearly undetectable to other players. The cheater simply appears to have good situational awareness. With consistent use over time, however, suspicions about whether or not a player downloaded his skills might arise.

With $p = 1.0$, we are recreating a scenario where every cheater uses a spinbot type cheat (or that all cheaters are bad at concealing their behavior). In any case, when $p = 1.0$, every player is aware of all cheaters while $p < 1.0$ means that some cheating goes unnoticed.

With $p < 1.0$, we are again projecting aspects of the social network onto the co-presence network. As $p \to 0$, repeated contact with infected individuals within a memory window becomes increasingly important for the spread of infection. Thus friend relationships (as indicated by the FTW! model predictors) will inherently become the prime vectors for infection. A susceptible is much more likely to have repeated exposures to infected friends.

Of the 37,140 players in the co-presence network, 607, or $\sim$1.6% had a VAC ban on record as of April 22, 2012. Although we note that these players did not have a ban for TF2 at the time they appeared in our logs, they serve as a baseline for the behavior of cheaters

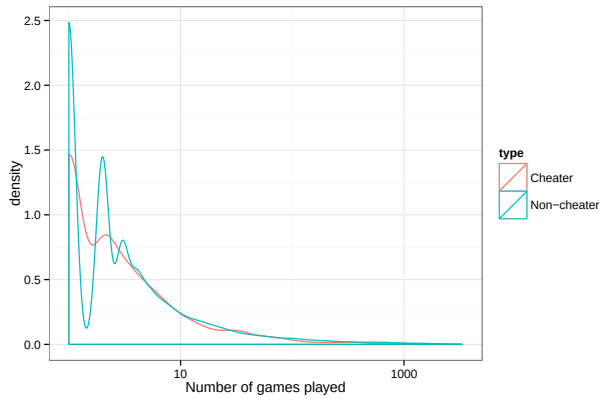

Figure 7.15: Density of the number of matches played for both cheaters and non-cheaters.

in general. Further, as can be seen in Figure 7.15, the cheaters have (mostly) the same distribution of games played, and thus our randomly selected seed nodes should have approximately the same number of chances to contact uninfected individuals as the empirical cheaters did. Thus we seed a randomly selected 1.6% of nodes as infected at the start of the simulation (i.e., $\phi_0 = 607$).

We draw the seeds from one of 3 player sets. The first is the full set of players, i.e., any player that was present for at least 1 game. The second set is the empirical cheaters. The third set is composed of players appearing in the largest component of the interacting friends network, thus having the most active patterns of interactions (and stronger ties).

7.2.3 Experimental Results

In this Section we present our experimental results on the static networks followed by the results on the dynamic co-presence network.

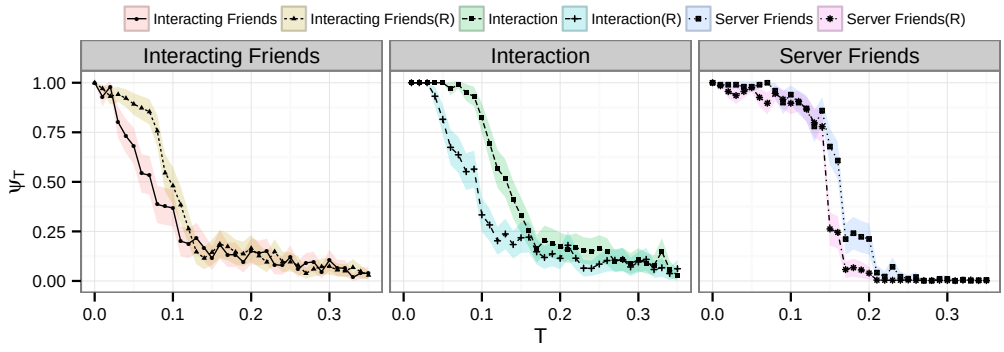

Figure 7.16: Relative cascade sizes for static networks with $M = 1$, $g(u) = T \times degree(u)$, $p = 1$, $f(u) = 1$.

7.2.3.1 Static Networks

Figure 7.16 plots the percentage of contaminated nodes as a function of the contagion threshold for each empirical network and for their permuted versions. (As a reminder, the higher the threshold, the harder for an individual to become infected). Each data point is the result of 100 simulations for each network; we generate 100 permutations of each empirical network.

We make several observations from Figure 7.16. First, while all networks supports contagion, the critical threshold T_c (the threshold at which 90% of the nodes become infected) is different for each of them. Thus, the server friends network is the most susceptible for cascading behavior, and the interacting friends is the least susceptible under the conditions simulated.

Second, the networks inferred from the declared OSN relationships do not support complex contagion for the critical threshold estimation used in previous studies [CEM07], where a contagion is deemed complex if $T_c > \frac{1}{\langle k \rangle}$. One reason for the failure to support

complex contagion under this definition is that while our networks are not scale free, as seen in Figure 6.2, most of the literature assumes scale free networks.

Third, and most interesting, while the permuted versions of the server friends and interaction networks are *less* susceptible to contagion, the permuted interacting friends network is more susceptible. Specifically, for a contagion to penetrate 75% of the interacting friends network, the largest threshold is about 0.04. In comparison, for the randomized versions of the interacting friends network, a threshold of about twice that much (0.08) is sufficient. This is unexpected and likely due to the bi-modal betweenness distributions seen in Figures 7.11 and 7.12. Only about 20% of nodes in the interacting friends network have a normalized betweenness greater than 0.001, and less than 3% have more than 0.01; the overwhelming majority of nodes have almost no shortest paths flowing through them. When we permute the network, we are also redistributing much of the concentrated betweenness, making the distribution more uniform. Because the high betweenness centrality nodes are also high degree nodes, making them more difficult to infect, the permutation removes the need for the contagion to pass through them to still have a successful cascade. In other words, permuting the network does not necessarily make infecting hub nodes any easier (they still need many cheater friends to become cheaters), but, it allows the infection to spread between nodes with few neighbors who were not previously directly connected.

Moreover, the middle row in Figure 7.17 ($T = 0.1$) shows that the server friends network experiences a quick acceleration of infection, while the spread is more consistent over time for the interacting friends and interaction networks. In the interaction network, at time $t = 10$, about 65% of the network is infected, but only 30% is infected in the permuted version. The differences in the server friends and interacting friends networks are less pro-

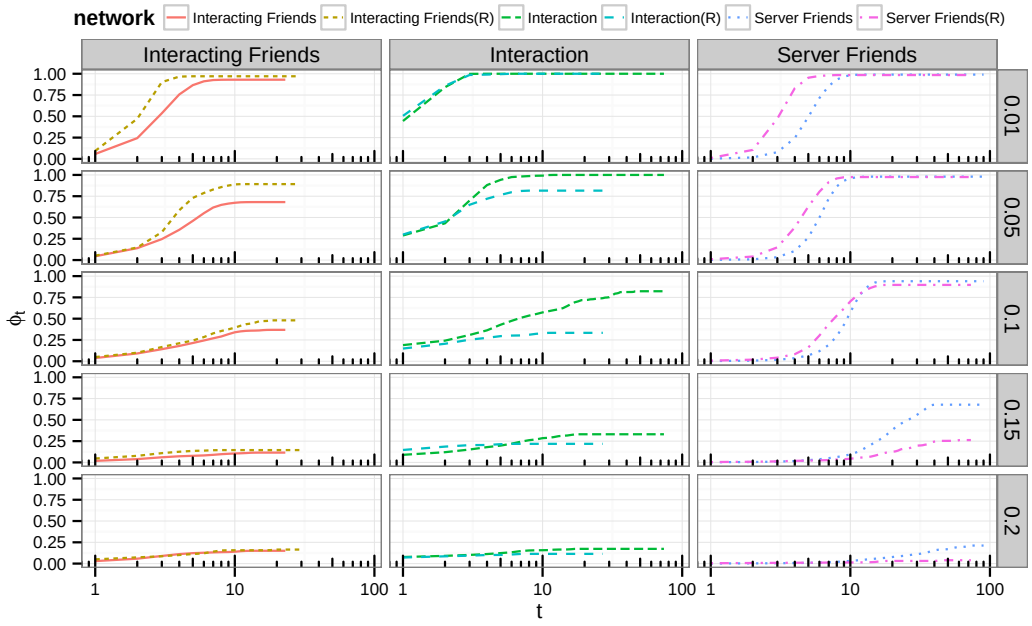

Figure 7.17: The fraction of infected nodes in the static networks as a function of time. Rows correspond to different fractional thresholds T. Columns group each empirical network with its permuted version "(R)".

nounced, with only about a 10% difference between the server friends and its permutations when $T = 0.1$.

To summarize, our simulations showed three things. First, because the permutation algorithm maintains the degree distribution, we know that the degree distribution is not a sufficient metric to predict the spread of the contagion. Next, neither the interacting friends nor server friends network support complex contagion when using the Centola estimation of critical threshold. Finally, the interacting friends network defies expectations and becomes *more* susceptible to contagion when randomized.

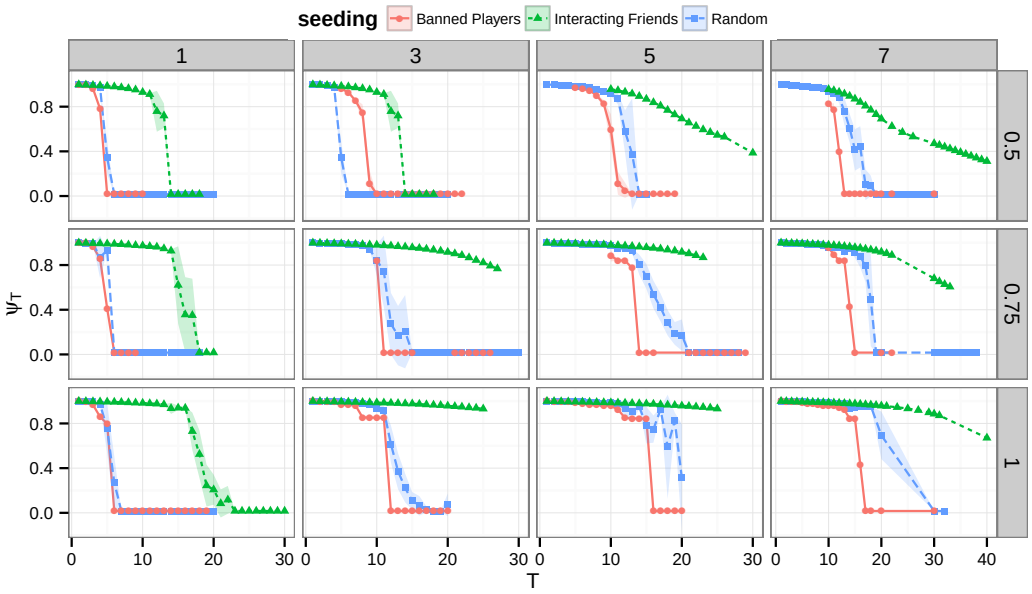

Figure 7.18: ψ_T for various values of M (columns), p (rows), and various seeding strategies.

7.2.3.2 Dynamic Network

Figure 7.18 plots ψ_T for the co-presence network with different exposure probabilities and shows a very surprising result with respect to cheaters' positions in the network and how it affects the contagion process. When seeded with the empirically known cheaters (the red line with circles), the network is at its least susceptible. In fact, the critical threshold when seeding with the banned players is the lowest of our three seeding strategies. In short, the empirical position of cheaters in our co-presence data prevents them from initiating an epidemic.

The interacting friends are clearly the most dangerous vector for the transmission of cheating behavior. For relatively high infection thresholds, cascades occur. In fact, when seeded from the interacting friends, the critical threshold is nearly triple what it is for the other

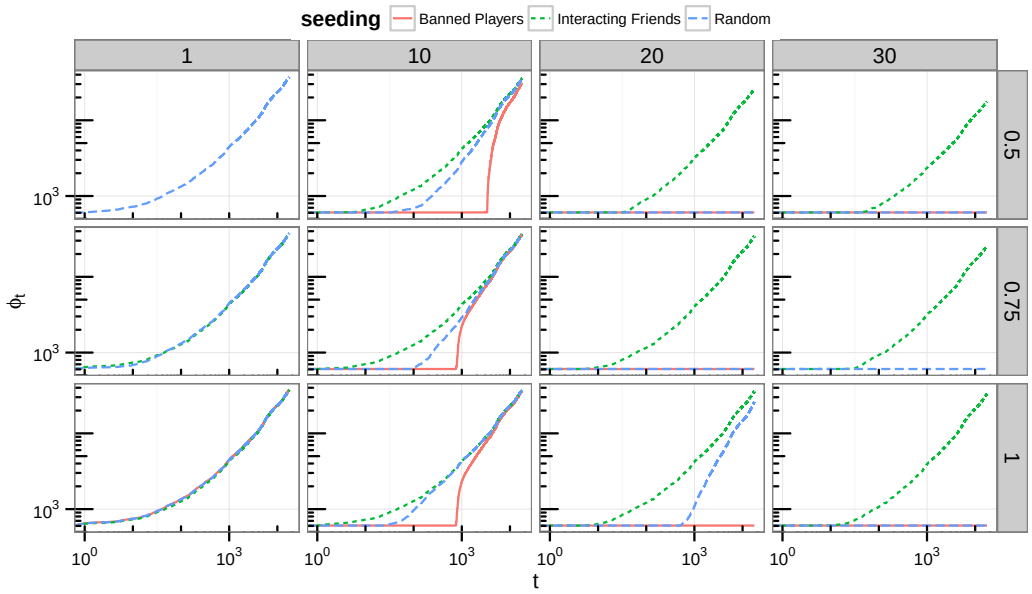

Figure 7.19: The total number of nodes in the co-presence network infected over time for different p (rows), thresholds (columns), and seeding strategies.

seeding strategies. This finding has direct implications for the gaming industry. Depending on the position of cheaters in the network, i.e., if they are both declared friends and actively interact with each other, the risks of contagion are more serious. Consequently, as Figure 7.19 shows, there is less time to intervene and disrupt potential breakouts.

Another point of interest is how exposure probability affects contagion over time. Seeding with the interacting friends reduces the relative impact of decreased exposure probabilities. As indicated in Figures 7.13 and 7.14 friends tend to play many more games, especially series of matches, with each other than non-friends. Because the reduction in exposure probability reduces the odds of receiving a dose from any single contact with an infected individual, the repeated co-play behavior leads to more exposures (and thus a higher cumulative probability of receiving a dose) from infected friends.

Interacting friends are a prime vector to contaminate the network because of their co-play habits. For example, consider a pair of players that plays 7 matches in a row together. If one player is infected, that gives 7 opportunities for his friend to receive a dose. Because friends play *series* of matches together, this increases the odds that doses will "stack" up together within the limits of the memory window. Because the interacting friends are some of the most central in terms of betweenness (Figure 7.11), the newly infected interacting friends are able to efficiently spread the infection further and faster than other seed selections, especially to the relatively poorly connected nodes on the edge of the network. As can be inferred from the negative degree correlation in the static interaction network (Figure 7.10), the network has a core-periphery structure where the interacting friends act as hubs, exposing the maximum number of susceptible players to the cheating contagion.

The results from the co-presence experiments can be summarized into two findings. First, the actual position of the cheaters in our dataset is the most inoffensive in terms of cascading. Second, seeding with interacting friends *greatly* increases the network's susceptibility to infection.

Chapter 8: Predicting Crowdsourced Decisions on Toxic Behavior

[1]In this chapter, we use the League of Legends Tribunal dataset (Section 4.3) and raise two research questions initially explored in [BK14].

- Research Question 1: Can we predict the crowdsourced decisions?

Since the perception of toxic behavior is subjective and different across individuals, the Tribunal deals with toxic behavior by a majority rule based on crowdsourcing. It has worked quite well, but requires a long time to obtain enough votes. Thus, our research question is to validate whether a certain part of the Tribunal can be assisted by machine learning. A few considerations are carefully addressed here.

First, we must define machine learning tasks. We can define various machine learning tasks on crowdsourced decisions in the Tribunal. Classifying 6 different combinations of decision and level of agreements, dividing cases into punished or pardoned, extracting high agreement cases, and recognizing less agreement cases are all possible tasks that machine learning could help. It is not only about the accuracy of the classifier but also about the application.

[1]Much of the work in this chapter was first published in Proceedings of the 23rd International Conference on World Wide Web, 2014 [BK14]. Permission is included in Appendix B.

Next, we must refine the training set. A Tribunal decision is either punish or pardon with a level of agreement: majority, strong majority, and overwhelming majority. Previous literature demonstrate that crowdsourced responses with high agreement are better for training a classifier than less agreement [BGC10]. Unfortunately, it is unknown how Riot Games divides these levels of agreement. We thus create different training sets and compare the accuracy of trained classifiers.

Third, we examine the use of non-linguistic features only. LoL has a global userbase in a few tens of countries across the world. This implies that chat logs in the Tribunal are not always written in English. For example, most messages in the Korean Tribunal are written in Korean. Various languages can potentially limit the portability of the classifier if it largely depends on the textual information left in chats. In other words, more non-linguistic features increase the generality of the classifier and bring higher practical impacts.

Finally, we detect sentiments in chats. This is the inverse of the above, maximizing the benefit of linguistic features. Many methods have been developed for detecting sentiments conveyed in texts. Our intuition is that negative sentiments might be captured from toxic behavior or other players' reaction when toxic playing occurs.

The work in this chapter indicates that yes, the Tribunal can be successfully assisted by a machine learning model.

- Research Question 2: What do the important features imply?

The next research goal is understanding decision-making in the Tribunal from the important features observed through supervised learning. Which features are important for predicting crowdsourced decisions? What do they mean? Answering these questions leads

to several interesting challenges. First of all, features we find could be a huge advance in online violence research. Toxic behavior has been typically considered hard to define. If we obtain a good quality supervised-learned classifier, it indicates the important building blocks in defining and understanding toxic behavior. Then, we draw a subsequent question. Can we apply our classifier or the important features to other games, or Internet communities? We find that our model is reasonably portable, even without training it on region specific data.

8.1 Features

The Tribunal can be seen as a 2-stage crowd-sourced solution. Stage 1 is the per-match reporting done by players that actually experienced the alleged toxic behavior. Stage 2 is the Tribunal case-by-case judgments. Toxicity is not having a bad game (possibly perceived as feeding, i.e., repeatedly dying on purpose, or assisting the enemy) or having a bad day and lashing out at a teammate (harassment). According to Riot, a certain threshold of reports from stage 1 must be met before moving on to stage 2, which reveals *repeated* toxic behavior. The reason that Stage 2 is necessary has to do with the vagueness of toxic behavior. Consider a player who is not very skilled. This player might exhibit tendencies that could be interpreted as toxic, for example intentionally feeding. This is compounded by attribution theory where a negative outcome (losing a match) triggers a search for an external cause. I.e., when a team loses, as is the the case in the majority of reported matches [KH], reporters might attribute the loss to a poorly performing player. Although there is an unskilled player report type, players are aware that no punishment is handed out for this category and thus might choose one of the punishable offenses instead. Thus, the second stage removes the subjectivity associated with direct interaction

with the possibly toxic player, as well as providing a more complete view of the accused's behavior over multiple matches. Players that have invested significant time, and are thus familiar with LoL, are able to pick out patterns of toxic behavior when reported matches are combined into a case.

The challenge lies in representing the parsimonious data presented to Tribunal reviewers in a form digestible by machine learning algorithms. We thus extract summarized statistics from each Tribunal case. We make use of two primary sources of information: 1) domain specific values extracted from the results of reported matches, and 2) the information provided by the stage 1 Tribunal participants.

There are, unfortunately, several points of variation when it comes to extracting the in-game values. First, each case has a varying amount of matches with no guarantee on the sequence in which the matches took place. Second, because of the variety of game play in LoL, there is no guarantee that matches are directly comparable, especially across different players. For example, a player with below average skill is likely to have a lower KDA (an in-game performance metric explained in the next section) than a player with higher skill. Now assume that the low skill player is not toxic, while the high skill player *is* toxic, yet both are reported for intentional feeding. There might not be a way of discriminating between the two using just KDA.

Although we average the per-match statistics across all matches with a given report type for each case, we could also use the worst/best matches. This is somewhat problematic as it requires a definition of worst and best. We include the standard deviation of each averaged statistic as a feature, which provides a sense of inter-match performance differences.

We then augment each instance with information provided by the Stage 1 Tribunal participants. Namely, we make use of the number of allied and enemy reports in a given match,

the number of reports where the reporter included optional human readable text about the offense, and the most common type of behavior reported for a given match. For each possible report type, we compute the relevant statistics across all matches in the case with said most common report type.

8.1.1 In-game Performance

In-game performance is the category of features that most requires the input of *experts*. LoL is a complicated game and the meaning of the various match-related statistics is unlikely to be divined by a reviewer, especially with respect to toxic behavior, without having invested a significant number of hours in gameplay themselves. Nevertheless, they are the easiest features to represent to a computer due to their purely numerical nature, so we extract a set of relevant statistics from the matches in each Tribunal case.

We first begin with the most basic statistic, one that is common to nearly all competitive games: kills, deaths, and assists. Kills and deaths are relatively self explanatory: simple counts of the number of enemies a player killed and the number of times said player died. Likewise, an assist is awarded to a player that participated in eliminating an enemy, but did not land the killing blow. The details of what qualifies an assist varies per game, but LoL awards an assist to any player that did damage or contributed passively (e.g., healed a teammate that landed the killing blow) within 10 seconds prior to the death of an enemy. Kills, deaths, and assist are raw scores, but are often normalized to a KDA metric, defined as:

$$KDA = \frac{kills + assists}{deaths + 1}$$

.

Unfortunately, due to the reliance on teamwork in games like LoL, a single toxic player can severely impact his teammates abilities to perform at a high level. For example, an intentional feeder is supplying the enemy team with gold and experience points, allowing them to acquire powerful items and abilities much faster than the feeder's allies. In turn, this can result in a low KDA not only for the toxic player, but makes it difficult for his allies to maintain a good KDA as well. For this reason, it might be difficult to distinguish toxic players based solely on KDA and our initial analysis [KH] indicated reviewers were not basing decisions only on KDA. However, two other statistics, damage dealt and received, might shed additional light on toxic players.

In LoL, attacks do a certain amount of base damage to other players, removing a portion of their hit points ("life"). When a player's hit points reach 0, he dies, a kill (with associated gold and experience) is awarded to his killers, and he must wait a certain period of time to "respawn" and return to the fight. The base damage is modified depending on both the offensive and defensive items a player has purchased. Anecdotally, toxic players in the feeding and assisting enemy categories will not buy items that aid in offense or defense. Thus, we might expect a feeder to have very low damage dealt and very high damage received relative to his teammates who have made purchases of useful items; even though they might not have the power to actually kill enemies (due to a gold and experience advantage given to the other team by the feeder), fair players' efforts are likely to show in terms of damage. Seeing which items a player bought could give more details, but it is overly specific and loses generality.

Next, we include the total gold and gold per minute earned by the offender. In LoL players earn gold in several ways: 1) passively at a pre-determined rate, 2) destroying towers,

3) kills or assists, and 4) killing creeps, which are computer controlled monsters. Gold is known to be a primary determinant of a match's outcome.

The final performance related feature is time played, which is useful for detecting leavers or players going AFK. In total, there are 364 features generated from in-game performance values only. This number is large because we group per-match values based on the most common report type.

8.1.2 User Reports

The first stage of the Tribunal, reports submitted players who directly observed the behavior, provides us with several pieces of information. First, we know the most common category of behavior reported per match. Next, we know how many allies and enemies made a report. Finally, reports can include a short (500 character limit) comment from the reporter. Intuitively, the extra effort required to add a comment to a report might indicate the intensity of the toxic behavior. Again, we group the per match values in the case based on the common report type for that match, resulting in a total of 28 features.

8.1.3 Chats

As seen in Figure 8.1, around 60% of cases have more than about 25% of the matches in them reported for offensive language or verbal abuse. This indicates that the observed toxic behavior was expressed (at least partially) via the chat system. For this reason, we intuit that the chat logs have predictive power. Linguistic analysis is an area of intense research [FMDD13, GM11, TSSW10], and the corpus of chat logs in our data set would

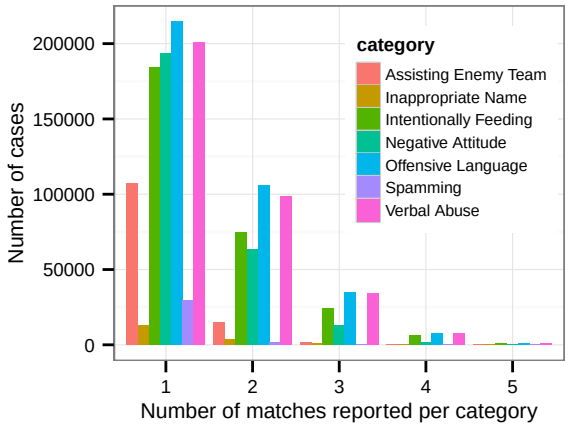

Figure 8.1: The number of matches reported in a case for each category of toxic behavior per Tribunal case.

provide an interesting case for cutting edge techniques. We use an intuitive and straightforward method, "happiness" index.

Happiness, and its relation to language, is a widely studied area of psychology [BGB86]. For example, in the Affective Norms for English Words (ANEW) study [BL99], participants graded a set of 1034 words on a *valence* scale of 1 to 9 (in 0.5 increments). Valence is the psychological term for the attractiveness (positive) or aversion (negative) to something; in this case a word. In other words, valence quantifies the "goodness" or "badness" of a word. Valence scores in the ANEW dataset are well distributed, as can be seen in Figure 8.2 (a), which plots the distribution of scores for all words in the ANEW dataset.

The ANEW study polled both female and male respondents. Riot reports that 90% of LoL players are male[2], and we thus use male respondent scores only. Although gender swapping often occurs in social games [LPC+13], according to Flurry's report[3], action and

[2]http://tinyurl.com/stfunub6
[3]http://tinyurl.com/stfunub7

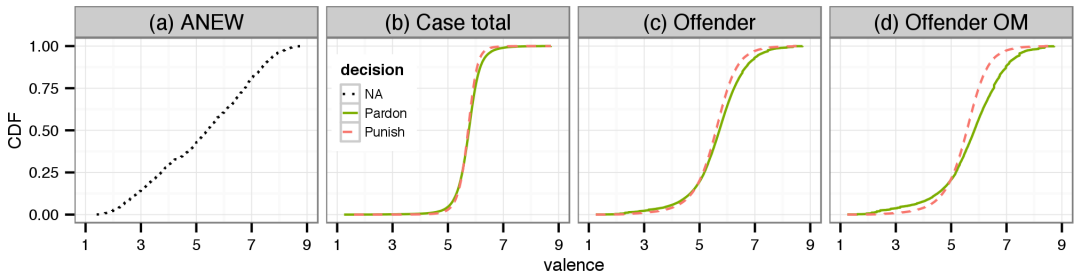

Figure 8.2: CDF of valence scores.

strategy are the top two genres that females usually avoid. Also, positive effects of gender swapping, enjoying a kind of "second" life, do not apply to LoL.

As demonstrated by Dodds and Danforth [DD10], valence scores for individual words can be used to estimate the valence for a larger corpus of text. The valence of a piece of text is defined as:

$$v_{text} = \frac{\sum_{i=1}^{n} v_i f_i}{\sum_{i=1}^{n} f_i}$$

where v_i is the valence score of the ith word from the ANEW study, and f_i is the number of times said word appears in a given piece of text.

While we acknowledge that chat logs are likely to contain typos and abbreviations, v_{text} has been shown to be robust across genres of text, including tweets [DD10, MFH+13], another medium where we might expect "Internet-style" speech patterns. For cases where no ANEW words were present, we define $v_{text} = 0$.

Figure 8.2 (b) plots the distribution of valence scores of all messages sent in a case for both pardoned and punished cases where $v_{text} \geq 1$. A two-sample Kolmogorov-Smirnov test confirms the distributions are different. When compared to Figure 8.2 (a), we see that "verbal" communication in LoL is generally neutral: most valence scores fall between 5 and 6. Further, cases that resulted in a punishment tend to have a lower valence score

when compared to pardoned cases. This indicates that the chat logs are likely to be valuable in detecting toxic behavior.

When looking at the scores of messages sent only by potential toxic players, offenders, in Figure 8.2 (c), it becomes clear that toxicity is present in a quantifiable way within the chat logs. The distributions for both punished and pardoned offenders is lower than the valence of the entire chat logs. The mean for punished and pardoned users are 5.725 and 5.779, respectively. Pardoned users are indeed more likely to have higher valence scores. Interestingly, the difference is mainly present in terms of "above average" (≥ 5) valence scores for pardoned users as opposed to a tendency towards below average scores for punished players. We also discover that the difference between punished and pardoned offender becomes bigger if more reviewers are agreed. Figure 8.2 (d) shows the valence score of toxic players when overwhelming majority is agreed. The mean for only those who are punished or pardoned in this case are 5.699 and 5.751, respectively.

In addition to the scores described above, we also include the valence scores for bystanders and victims for each report category. Here, we treat verbal abuse, offensive language, and negative attitude differently from the other categories. For these cases we have previously observed that reports by enemies are much more common. This can be attributed to bystander theory, which says that bystanders, i.e., those not directly harmed by bad behavior, are much less likely to take action against it. In the case of, e.g., intentional feeding, not only are enemy teams not directly harmed by the behavior, they actually receive a benefit. While the quality of the competition may decrease, the odds of a win are much more in their favor.

When it comes to chat based offenses, however, a toxic player can lash out at everyone in the match. He can insult the enemy team when they are performing well, and trash

talk when they are performing poorly. For example, a common insult in LoL is to call someone a "noob," slang for "newbie," implying a lack of ability. It is a catch-all negative term used as a response to criticism, to call out poor play, as a form of trash talk, and just plain meanness. The word "noob" appears 3,551,328 times in the chat logs for the NA region; an average of 6 times per case and 1.7 times per match, with players under review saying "noob" at more than double the rate per message sent as non-offenders. Because these communication based categories can affect both enemies of the offender, allies, or neither, we consider their victims to be the offender's allies if only allies reported him, enemies if only enemies reported, and all players if both enemies and allies reported him.

With the above in mind, we extract 60 features per case from the chat logs.

8.2 Models

As we introduced above, we extract features from different categories. We then build separate models for each category, and a full model to contain all the features: an in-game performance model, a user report model, a chat model, and a full model. The intuition behind these basic models is comparing different sources.

First, in the in-game performance model, we can divide features by offender performance and other players performance. Offender performance can reflect intentional feeding. Players with noticeably bad performance might die over and over intentionally. We note that non-toxic *unintentional* feeding does occur, but only *intentional* feeding is toxic behavior. Other players' performance relates to the team competition aspect of LoL. Attribution theory says that individuals will look for external causes of failure [Wei80]. The most obvious cause would be poor performance by an ally, and is likely to manifest as verbal abuse

(harassment). In other words, a toxic player might lash out at the worst performing ally due to the perception that a loss was the fault of said ally. We hypothesize that the intensity of the verbal abuse, and thus the likelihood of punishment in the Tribunal, increases as the offender's performance diverges from the worst performing player on his team.

A less intuitive reasoning in favor of this model is that a poor performance by a player *does* have an impact on the rest of his team. Previous analysis indicates that KDA alone is insufficient in predicting the judgment of the tribunal: there was no correlation with KDA and corresponding Tribunal decision [KH]. I.e., players with the best KDA were about as likely to be punished as those with the worst.

The user report model depends on how players in a match perceive toxic playing. Although user perception on toxic behavior is different, more reports in a single match means more people recognize it, and implies more severe toxic behavior. In our initial analysis [KH], we find that the number of reports in a single match is highly correlated with the likelihood of being punished. The chat model relates to verbal abuse and offensive language. In the Tribunal, reviewers can see the final in-game performance but not how the match played out over time. Chats are the only source to give reviewers *context* about what happened and what other players thought about it.

8.3 Results

In this Section we present the results of our experiments. We discuss the effects of confidence in crowdsourced decisions, the model features that are most helpful in predicting toxic behavior, and the portability of our classifier with respect to different regions.

8.3.1 The Effects of Confidence in Crowdsourced Decisions

The first research question we answer is whether or not reviewer agreement is relevant when training a classifier. To do this, we grow random forest classifiers from only cases of a given agreement. We then evaluate each classifier with a separate test set for each agreement. The intuition is that the cases with the highest level of agreement from the reviewers display the most egregious toxic behavior, providing a baseline for the remainder of cases.

Figures 8.3, 8.4, and 8.5 plot the Receiver Operating Charactering (ROC) curves for testing sets of each agreement type with classifiers trained from varying agreement cases. We observe that training the classifier with overwhelming majority decisions results in the highest Area Under the ROC Curve (AUC) across all cases. Our ability to distinguish between guilty and not guilty increases with the level of agreement that we train the classifier with. This is consistent with previous research [BGC10]. While the classifier trained with overwhelming majority is the most discriminating across the board, training with strong majority cases has similar performance, while performance drops off considerably when training with the majority decision cases.

This experiment has several implications. First, it might be beneficial to look at the extremes of behavior, clear cut pardons and punishes, to better predict borderline cases. Second, it might be fruitful to predict decisions based on confidence. I.e., finding the most obviously guilty or innocent individuals, leaving the borderline cases to human reviewers. Third, it reveals the difficulty in discriminating between all but the most egregious offenses.

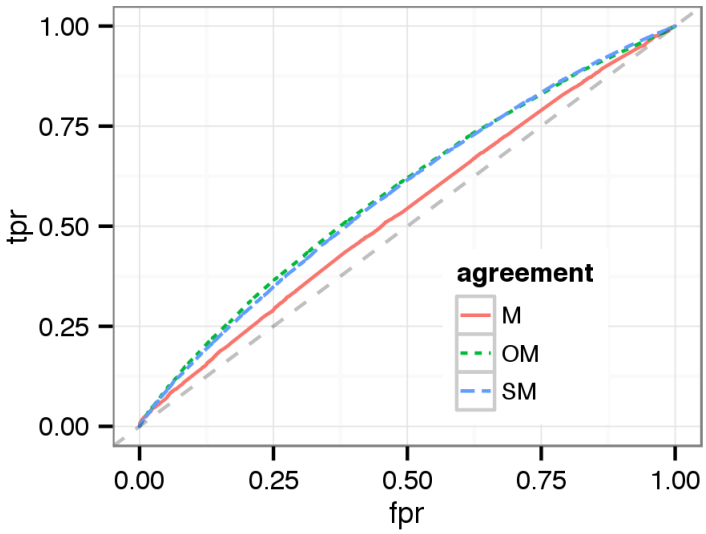

Figure 8.3: ROC curves for cases with a *majority* decision using a classifier trained from majority cases ("M"), strong majority cases ("SM"), and overwhelming majority cases ("OM").

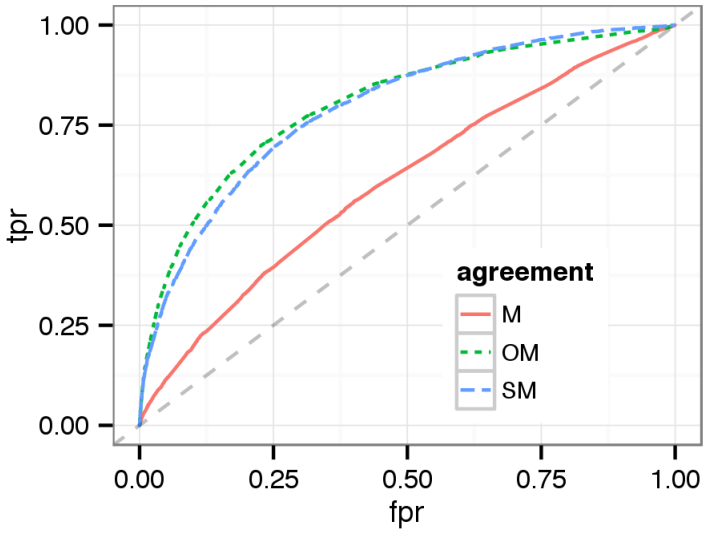

Figure 8.4: ROC curves for cases with a *strong majority* decision using a classifier trained from majority cases ("M"), strong majority cases ("SM"), and overwhelming majority cases ("OM").

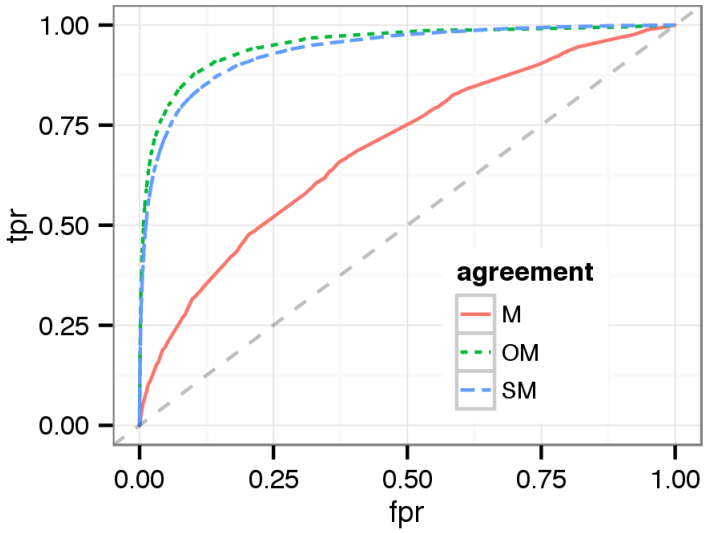

Figure 8.5: ROC curves for cases with a *overwhelming majority* decision using a classifier trained from majority cases ("M"), strong majority cases ("SM"), and overwhelming majority cases ("OM").

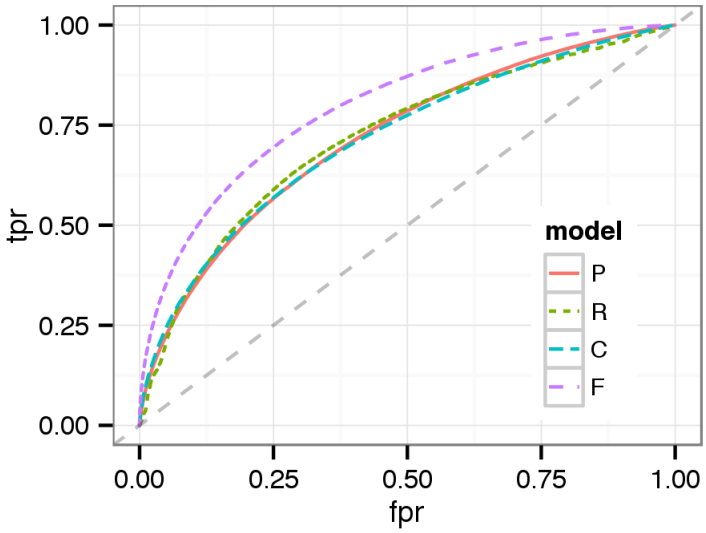

Figure 8.6: ROC curve for predicting Tribunal decisions with models using in-game performance (P), user report (R), chats (C), and all available features (F).

8.3.2 What Do Tribunal Reviewers Base Their Decisions On?

We now address what information Tribunal reviewers might be basing their decision on. We present a few learning results to show the performance of our Random Forest classifier.

We begin with the performance of predicting decisions, pardon or punish without considering the agreement level. Figure 8.6 presents ROC curve for predicting decisions, punish or pardon, with the performance (P), report (R), chat (C), and full (F) models. AUCs are 0.7187, 0.7195, 0.7157, and 0.7991 for the performance, report, chat, and full models, respectively. We observe that each model shows comparable performance.

Table 8.1 shows the 5 most important variables for predicting decisions. We omit the important variables for each category of toxic behavior, but it is similar across the categories.

In the performance model, we find that enemy performance is a good predictor for decisions because offender or ally performance is relative in team competition games. Also, offender performance itself is important for decision making in the Tribunal. Interestingly, the number of deaths is more important than KDA. This might relate to partner blaming. Toxic players blame teammates, e.g., saying "noob", when the toxic player dies. The implication is the toxic player died because allies did not help him. In this situation, the number of deaths could be a good measure to reveal which is true: allies did not help or toxic players performed poorly.

In the chat model, we find that the most important variable is the valence score of offender no matter what the reported toxic category is. This agrees with our intuition that reviewers can see the context from chats and infer what happened. They gain insights from chat not only for verbal abuse or offensive language, but also other categories of toxic

Table 8.1: The top 5 ranked features from an information gain evaluator for Tribunal decisions.

Rank	Feature
Performance only	
1	verbal.abuse.enemies.kda
2	verbal.abuse.enemies.gpm
3	verbal.abuse.offender.deaths
4	verbal.abuse.enemies.kda.avg.per.player
5	verbal.abuse.offender.kda
Chat only	
1	case.offender.valence
2	verbal.abuse.offender.chat.msgs
3	offensive.language.offender.chat.msgs
4	verbal.abuse.offender.valence
5	verbal.abuse.total.chat.msgs
Report only	
1	verbal.abuse.allied.report.count
2	verbal.abuse.allied.report.comment.count
3	intentionally.feeding.allied.report.count
4	intentionally.feeding.allied.report.comment.count
5	offensive.language.allied.report.count
Full	
1	case.offender.valence
2	verbal.abuse.allied.report.count
3	verbal.abuse.offender.chat.msgs
4	offensive.language.offender.chat.msgs
5	verbal.abuse.allied.report.comment.count

behavior. Also, this demonstrates that the ANEW dataset works well even with chats in online games. The second and third most important variable are the number of messages sent by the offender when the reported reason is verbal abuse and offensive language. This implies that the likelihood of toxic behavior goes up if the offender talks more. This can be easily recognized in real-time and thus pragmatically used for warning toxic players through visual cues. The fourth important variable is the valence score when the behavior is reported as verbal abuse. This is straightforward to understand. We find that number of total messages sent when the report reason is verbal abuse is the fifth important variable. This is the only feature in the top five that is not only related to the offender, but also others. If more players are involved in a quarrel, it is a strong sign of verbal abuse having occurred.

In the user report model, top variables are related to how many reports are submitted. The more reports, the more likely the Tribunal will decide to punish. This strongly agrees with previous analysis [KH]. Additionally, we find that reviewers care about short comments in user reports. This implies that a user interface encouraging comments might be helpful for crowdsourced decision-making.

The top 5 features in the full model are entirely from the chat and report models. The total valence of the case is the number one feature, which highlights how much toxic behavior is visible/expressed via in-game communication. The second most important feature in the full model comes from the report only model, highlighting how our approach dovetails with the first crowdsourced stage of the Tribunal. The hints provided by those that directly experience toxic behavior are useful not only to human reviewers, but, for an algorithmic solution as well. Next, we note that the 6th and 7th most important features

in the full model are from the performance model. Thus, while in-game performance numbers *are* a predictor of toxic behavior, context is key.

We also look into the top 5 important variables in predicting overwhelming majority pardon and punish, respectively. We omit the details but we highlight two findings by comparing them. One is that there are great discrepancies of important variables between the model for predicting overwhelming majority pardon and punish. It implies that reviewers might make a decision for punish and pardon according to different mechanisms. The other is that, similar to predicting decisions, there are some commonalities in important variables across the category of toxic behavior for predicting overwhelming majority pardon and punish. For example, in predicting overwhelming majority pardon, the most important variable in the report only model is the number of reports by allies across the category. Similarly, in predicting overwhelming majority punish, the most important variable in the chat only model is the number of messages sent by the offender across the categories. Of course, there are some specifics for each category. For predicting overwhelming majority punish, in the report only model, the number of reports by enemies is more important than the number by allies in intentional feeding, but in verbal abuse, allies' reports are more important than enemies'. For future work, we intend to combine this result with qualitative user interviews and plan to reveal details of the mechanism of reviewers' decisions.

Figures 8.7 and 8.8 show ROC curves for predicting overwhelming pardon and overwhelming punish, respectively. Their AUC are 0.8049, 0.8055, 0.8269, and 0.8811 for overwhelming pardon decisions and 0.6509, 0.6886, 0.619, and 0.7461 for overwhelming punish decisions. There are some interesting differences between the curves for the agreement levels. First, detecting overwhelming pardon is easier to find than overwhelming majority punish

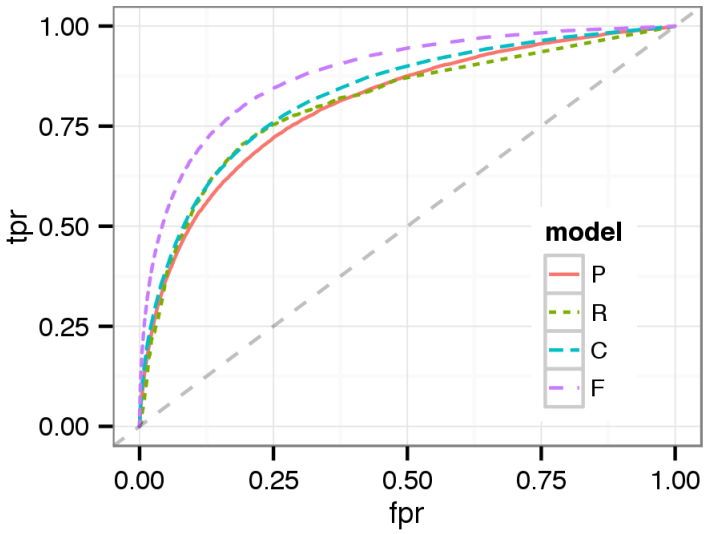

Figure 8.7: ROC curve for predicting overwhelming pardons with models using in-game performance (P), user report (R), chats (C), and all available features (F).

and shows quite good performance. It is mainly because overwhelming majority punish is very close to strong majority punish, as we mentioned in Figure 8.3.

This proves the feasibility of automatically assigning tasks to crowds and machines according to their difficulties. Quinn *et al.* demonstrate that crowd-assisted machine learning can achieve high overall accuracy when assigning easy tasks to machine and fuzzy tasks to human [QBYL10]. Although they divide cases into two classes by human experts, our result demonstrates that we can do it automatically.

In context of LoL, properly dealing with overwhelming pardon case is more important than overwhelming punish. Wrongly punished players would leave LoL, while wrongly pardoned players sometimes would be back to the Tribunal. If they do not come to the Tribunal again, it means that they are reformed and fine for the overall LoL ecosystem.

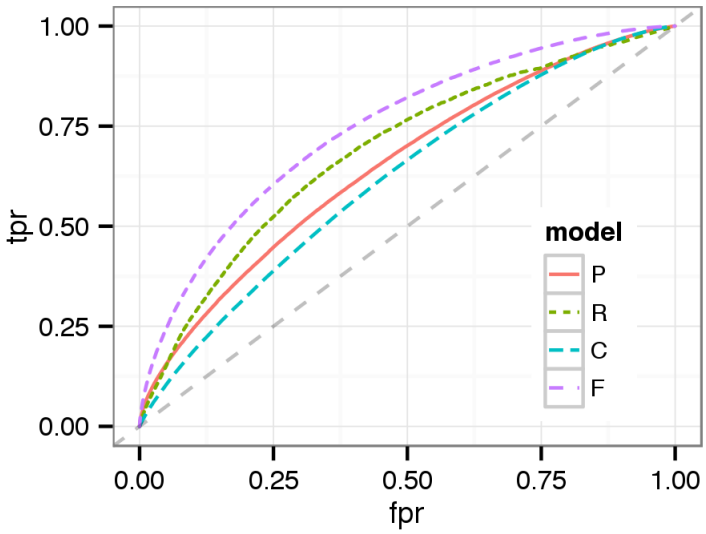

Figure 8.8: ROC curve for predicting overwhelming punish decisions with models using in-game performance (P), user report (R), chats (C), and using all available features (F).

In addition, we achieve high accuracy to predict decisions of punish or pardon on clear-cut cases, overwhelming majority punish and pardon cases, as in Figure 8.3. Thus, it is feasible that our classifier can automatically extract clear-cut cases and make accurate decisions on them. It is great way to assist crowdsourcing platform by machine learning.

Second, the order of models by performance is different in two cases. In detecting overwhelming majority pardon, we observe that a chat model shows the best performance, while a user report model is quite comparable for the most part. By contrast, in detecting overwhelming majority punish, a user report model shows quite good performance. This is an interesting finding. Reviewers need to understand context from chats to prove not guilty, but they also see why and how many times a toxic player is charged. This is consistent with our initial analysis, revealing the number of user reports is highly correlated with the likelihood of being punished [KH].

8.3.3 Classifier Portability

Finally, we explore whether or not our classifier is *portable*. Based on previous analysis [KH], we saw that there were statistically significant differences in Tribunal cases across the various regions that LoL operates. One underlying reason behind this is likely due to cultural differences realizing themselves both in the tendencies of toxic players as well as the reviewers. Because of these differences, we expect models trained on the North American dataset to not perform as well on the other regions. However, we *do* expect the models to remain superior to a coin flip in terms of discriminatory power. In other words, while we believe the models we specify are meaningful regardless of the region, the thresholds learned are probably sub-optimal.

Before presenting results, we stress an additional caveat related to linguistic features. The ANEW dataset is based on *English* words and was built from *American* respondents. This makes the linguistic features (highly relevant in the NA dataset) useless when applied to the KR dataset since English language words are almost non-existent. The EUW dataset includes players with a variety of native tongues, and anecdotally French, German, and Spanish are all spoken in-game. However, there is no language requirement to become a reviewer; you only have to have an account within the region you are reviewing. Also, English is a common tongue for gamers world wide. In fact, we see that less than 1% of EUW cases have an undefined v_{text}.

Figure 8.9 and 8.10 show ROC curves of predicting EUW decisions and detecting EUW overwhelming majority pardon cases by using classifier trained on NA. The performance of predicting EUW decision does not reach that of NA decision, but detecting EUW over-

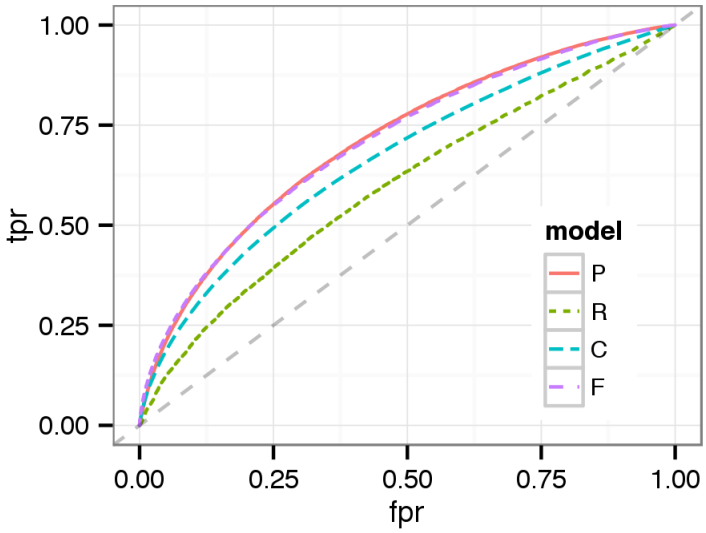

Figure 8.9: ROC curve for EUW decisions with classifier trained on NA.

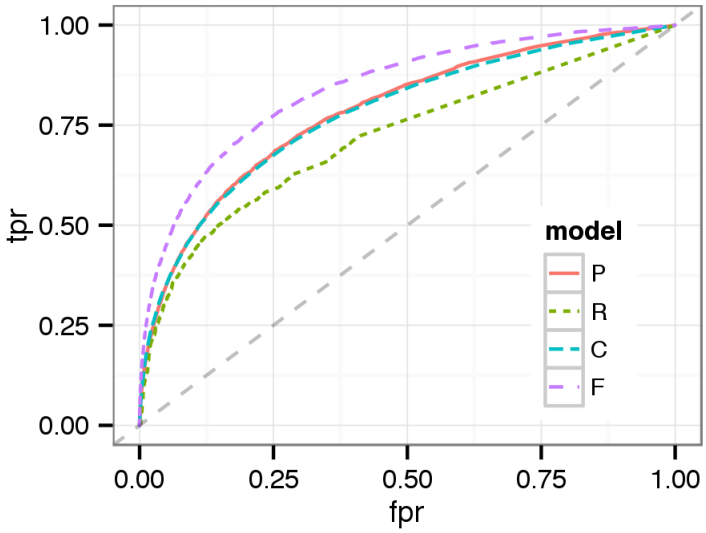

Figure 8.10: ROC curve for EUW Overwhelming Majority Pardons with classifier trained on NA.

whelming majority pardon is as good as NA. As with our hypothesis, this shows the potential of classifier portability, but at the same time, the existence of regional differences [KH].

8.4 Estimating the Real-world Impacts

We present the potential gain of time, cost, and accuracy if our classifier assists the Tribunal. One challenge is estimating the actual cost, time, and accuracy of reviewing cases in the Tribunal because Riot does not release detail statistics thereof, except a few infographics. We collect and complement partially available information to estimate required sources for the Tribunal.

First, we estimate the actual cost for crowdsourcing decisions. Initially, Riot gave 5 Influence Points (IP) as rewards to each vote only when a majority vote is reached, but have since removed this payment system. In LoL, IP is used for buying champions or skins. To measure how big 5 IP is, we need to convert it to real money. Some champions whose price are 450 IP can also be bought for 260 Riot Points (RP), that can be purchased by real money. Players pay $10 for 1380 RP. Through a simple calculation, we reach $0.02 for each correct vote. This is comparable fare with other crowdsourcing platforms [QBYL10].

Second, we estimate the time required for each case. According to the talk by Jefferey "Lyte" Lin at Game Developers Conference (GDC) 2013[4], reviewers have cast 105 million votes and reformed 280,000 toxic players. Other announcements by Steve "Pendragon" Mescon[5] reveal 50% of players warned by Tribunal are reformed. We thus assume that 105 million votes make verdicts for 560,000 toxic players and half of them are reformed. We

[4]http://tinyurl.com/stfunub8
[5]http://tinyurl.com/stfunub9

conservatively assume this is the lower bound of players who came to the Tribunal. This means that 187.5 votes are required for majority votes on a single case. From the same source, Riot reveals more than 47 million votes were cast in the first year of the Tribunal, implying that 1.49 votes per second are casted. From both, we can infer 125.85 seconds are required to reach a verdict for one case in the Tribunal. We reasonably assume that this is the acceptable speed where Riot's in-house experts manually review some intriguing cases.

Finally, we estimate the accuracy of the Tribunal. Lin said, "approximately 80% agreement between the [Tribunal] community and Riot's in-house team", in the same GDC talk. He added that the in-house team is less lenient than the Tribunal. Surprisingly, the overall accuracy of the Tribunal is comparable with our classifier with respect to Riot's in-house decisions. That is, in contrast to CrowdFlow [QBYL10], our supervised learner has the potential to replace the crowdsourcing part of the Tribunal with no real sacrifice in accuracy.

We now estimate the gain of Riot from the view of cost and victim players.

We already compute that the cost of each correct vote is $0.02. Conservatively, we estimate 50% of all votes fall into majority for each case. Since the Tribunal got 47 million votes the first year, its cost is 47M (votes) x 50 (%) x 0.02 ($) = 470,000 ($). As of March 2013, the number of votes reached 105 millions. Its potential cost surpasses 1 million dollars. With the success of new regions and a growing userbase, this cost will become huge.

As of October 2012, Riot announced that 12 million players play LoL everyday[6] and they play more than 1 billion hours every month. Thus, we estimate that a player enjoys LoL 83 minutes everyday which equates to 2.21 matches where one match usually spans 30 to

[6]http://tinyurl.com/stfunub10

45 minutes. In the first year, the Tribunal detected 47M / 187.5 = 250,667 toxic players. On average, 686.75 toxic players are warned by the Tribunal everyday. From this number we can compute number of potential victims who are protected by the Tribunal everyday. The number of matches toxic players participate everyday is 1517.74, and 13659.61 innocent players are exposed to toxic players. If our classifier and the Tribunal works together in a 50-50 manner, we can protect 13,659 more players everyday and more than 400 thousand per month.

8.5 Limitations and Consequences of Our Approach

Although we have shown our approach is relatively robust, even across different regions of the world, there are some limitations. First, although outside the scope of this paper, LoL features an ever changing "meta-game" which dictates commonly used strategies and tactics. Although the features we use in this paper are not directly tied to the meta-game, for example which particular items or champions are selected, they are indirectly related. E.g., the amount of damage dealt and received or gold earned might be influenced by the current meta-game. Although the dataset in this paper spans multiple meta-games and we still see good results, accuracy might be improved by training classifiers on data only from the meta-game of cases under examination.

Toxic players could, in theory, adapt to avoid detection. For example, toxic players might say "Wow, you are a great player!" sarcastically instead of calling someone a noob. Or, perhaps an intentional feeder would go out of his way to damage the enemy team, but not enough to actually kill them. This raises some interesting points. First, toxic behavior only has an impact if it actually affects people in a negative way. It would take a ma-

jor shift in the community's understanding of language to infer that a seemingly positive statement was meant in a negative way. Second, in the case of toxic players adapting their play style, we argue that is a hidden benefit of our approach. Detecting toxic behavior has significant value, but preventing it wholesale or reducing its impact is a much better solution. Although a feeder could attempt to hide his intentions by damaging the enemy team, he is consequently reducing the negative impact of his feeding by still providing some utility to his team.

8.6 Summary

This chapter presented an analysis of toxic behavior in the League of Legends. From millions of crowdsourced decisions, we modeled toxic behavior from features derived from in-game performance, the perspective of players who directly experienced the behavior, and chat logs. We discovered that all three models were successful in labeling toxic behavior, but, combined they were much more powerful. Additionally, we demonstrated that our model was robust across regions of the world, and an automated system could save significant time and money.

Chapter 9: Conclusion

This dissertation presented an analysis of bad behavior in online video games. We provided an architecture and reference implementation for the collection and study of large scale longitudinal social network data. Using this system, we collected data on over 12 million gamers and 700 thousand cheaters from the largest gaming social network on PC. We performed a quantitative study of about 10 months of interactions among over 30 thousand gamers on a community owned and operated game server. We then used these findings to perform an analysis of contagion with an interaction based model. Finally, we analyzed and extracted a model for detecting toxic behavior of gamers.

Our primary findings can be summarized as follows.

1. Cheaters are as well embedded in the social network as fair players.

   Although they exhibit anti-social behavior, cheaters had about the same number of friends as non-cheaters and are well represented in terms of degree and betweenness centrality. When we looked a bit deeper, we discovered that cheaters and non-cheaters had different neighborhood compositions. In short, cheaters tended have a higher fraction of cheaters in their neighborhoods.

2. Gamers form relationships such that interactions between friends dominates interactions between non-friends.

We analyzed detailed logs from a community owned and operated game server to explore the impetus behind the creation of declared relationships. We found that friends interact with each other much more often than non-friends interact. When exploring events prior to and after a declared relationship was formed, we found that interactions steadily increase, peaking when the relationship is declared, and then begin to decay.

3. There is a statistically significant effect such that having more cheater friends is a predictor for becoming a cheater. I.e., there is evidence that a contagion process is at play.

   Using low resolution timing data, we built a classifier that indicated a relationship between the number of cheater friends and the likelihood of becoming a cheater in the future. We used a proportional hazard model to show this effect was increasing in a statistically significant fashion up until about four cheater friends. We then visualized the spread of this behavior using high resolution timing data, showing how clusters of cheaters appear within the network.

4. There is a social penalty involved when knowledge of cheating behavior is made public (via the application of a VAC ban); cheaters tend to lose friends.

   When we examined the effects of the publicly visible cheating flag, we discovered that cheaters tended to lose friends after a VAC ban was applied. Conversely, non-cheaters tended to gain friends over the same period of time. When using high-resolution data, we discovered that this loss of friends happened very quickly after the ban become publicly known, indicating the existence of a social penalty.

5. Toxic behavior can be modeled and detected to a high degree of accuracy using crowd-sourced decisions.

We derived a model for predicting toxic behavior using millions of crowdsourced decisions from the most popular online game in the world. The model incorporates domain specific features as well as a semantic analysis component. We were able to discriminate between guilt and innocence upwards of 80% of the time, and in some circumstances about 90% of the time. Moreover, the model proved portable across geo-political regions.

The remainder of this chapter provides future work and discussion related to this dissertation.

9.1 Future Work

There are several avenues of future work related to this dissertation. First, we wish to explore the impact of match making services on bad behavior. Unlike our SH dataset, players of games with match making systems do not choose who they play with, but instead are placed into games according to their skill level. As noted by Herbrich et al. [HMG07], when the TrueSkill system was implemented, there was an increase in cheating behavior, and even new forms of cheating springing up. Because match making is such an integral part to modern multiplayer games, this relationship is due for a deeper exploration. What is it about match making systems that encourage cheating? How can match making systems be altered or augmented to take this into account?

We also wish to validate the model presented in Chapter 7. The main challenge for this avenue of study is acquiring sufficient data. Unfortunately, we only have data from one of thousands of servers from TF2, and the cheaters appearing in our dataset did not cheat in that game when they appeared in our dataset; they would have been VAC banned and

unable to play. However, we are currently devising a data collection strategy to monitor the virtual whereabouts of gamers from a wider sample of servers. Although it is unlikely we will be able to obtain as detailed a dataset as our TF2 logs, our model indicates that co-presence data should be sufficient.

We additionally wish to explore whether or not toxic behavior has contagious effects. Based on our experiences in the real world, we hypothesize that as players are exposed to toxic behavior over time, they themselves are more likely to become toxic. Although both cheating and toxic behavior are "bad," they are fundamentally different, and we suspect these differences to be exposed in terms of modeling the underlying contagion process.

We wish to continue building on our Pregel implementation, Elaine 3.4.3. Elaine continues to be developed ahas been made available to the public at http://github.com/worst/elaine. It is particularly lacking a robust fault tolerance model. From our instrumentation, we found that while Elaine scales both in number of compute nodes and the size of the graph, there are performance bottlenecks in inter-node communication. The performance hits appear to be taking place in DCell, the distributed object library Elaine is built on top of, but, we are uncertain whether this is a fundamental issue or something that can be mitigated via Elaine's implementation. We are also exploring different graph partitioning methods to increase performance.

Finally, we are currently exploring performance analytics for games. A growing trend in real world sports is the use of statistical methods to inform, in a quantitative manner, strategies, tactics, and other organizational decisions. While professional eSports differ from real world sports in many ways, the techniques originating from, e.g., baseball, have been successfully used, albeit with domain specific modifications, to other sports. These analyses have been exceeding successful in increasing performance of professional sports

teams, and we see no reason to believe they would not be successful for eSports. As eSports continue to expand, a foundation for analytics could have a large impact on the industry.

9.2 Discussion

Our findings have impact not only in video games but also in the real world. As has been shown by other researchers, the behavior of people online maps to their behavior in the real world [BTP14, CWC+09]. This mapping principle allows us to transfer our findings, with some caveats, to more general behavior.

Unlike many studies on bad behavior, we were able to show hard empirical evidence at planetary scale that cheating is contagious. For gaming in particular this suggests possible mechanisms to detect and, more importantly, prevent cheating. For example, the social penalty that cheaters face could be exploited. As it stands now, a change in cheater status, while public, requires actually looking at a player profile. If it was instead, say, broadcast via a message to all the friends of the cheater, then it might serve as an information driven vaccine [RTL12] in addition to being a deterrent.

As we saw in Section 5.4.1, there appears to be substantial differences in cheating behavior in different socio-geographic regions. On the other hand, in Section 8.3.3, we found that a toxic behavior classifier trained on North American data was reasonably successful at detecting toxic behavior on European data. Any solution for dealing with bad behavior online should take socio-geographic concerns into account. Even though underlying mechanisms might be similar across regions, the details, or in the case of Section 8.3.3 model parameters, likely differ.

For online video games, designers should keep in mind that although they might codify the rules of the virtual world with programming, players are influenced by their real world experiences. As we noted in Section 5.2, we discovered a cheater who had exceeded the maximum number of friends that is supposed to be allowed on Steam Community, seemingly in contradiction to the social penalty we saw. As it turns out, this cheater was from one of the Nordic countries, where we also found a very high population of cheaters. Although more study is necessary, this indicates that a solution that might work in one part of the world, e.g., an information driven vaccine, might not work elsewhere.

In Chapter 8 we developed a predictive model for toxic behavior. While certainly relevant to online games, the effectiveness of this model implies something more. Recall that toxic behavior has a somewhat ambiguous definition. This ambiguity is what makes derivation of a mathematical model difficult. How can we precisely represent toxic behavior to a computer if we cannot explicitly define it for humans? The solution that we took amounts to using the crowdsourced decisions as an *implicit* definition. The implication is that by dissecting the derived model, we might increase our knowledge and ability to describe certain types of behavior.

As with any observational study, there are some caveats to keep in mind. While research across disciplines indicates that our findings are portable, we caution that our findings might capture specifics of unethical behavior that are *not* cross applicable. For example, while a student cheating on a test can make his own cheat sheet, most of the cheaters in our study are dependent on 3rd party cheat creators. Although cheating behavior definitely diffuses in both cases, in video games the knowledge of how the cheat is executed does not.

There are also significant concerns about privacy and validation. Any mechanism that exploits the structure of a social network for detection of unethical behavior inherently faces concerns of "guilt by association." Indeed, our findings indicate that not *all* of a cheaters friends will become cheaters themselves, but rather that there is a statistically significant increased *chance* they will cheat. Taking action against a *potential* cheater based solely on the actions of his associates will undoubtedly result in the condemnation of innocents. Similarly, deploying the model for detecting toxic behavior (Chapter 8) will produce both false positives and false negatives: some innocents will be found guilty while some guilty will go free.

While the consequences of deviations from the model in video games are important, when applied to the real world they could be disastrous. A false positive for cheating in academics or business could ruin someone's life. That said, we implore researchers delving deeper into this problem to tread carefully. While our findings can, and should, be used to inform future studies and solutions, these caveats should be carefully considered.

List of References

[AKS+11] M.A Ahmad, B Keegan, S Sullivan, D Williams, J Srivastava, and N Contractor. Illicit Bits: Detecting and Analyzing Contraband Networks in Massively Multiplayer Online Games. In *IEEE 3rd International Conference on Privacy, Security, Risk and Trust*, PASSAT '11, pages 127–134, October 2011.

[AMS09] Sinan Aral, Lev Muchnik, and Arun Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, 106(51):21544–21549, 2009.

[APB11] APB Reloaded Dev Blog. http://goo.gl/m9klr, 2011.

[Ari12] Dan Ariely. *The (Honest) Truth About Dishonesty: How We Lie to Everyone—Especially Ourselves*. Harper, 2012.

[BASS+10] Shiraz Badurdeen, Omar Abdul-Samad, Giles Story, Clare Wilson, Sue Down, and Adrian Harris. Nintendo Wii video-gaming ability predicts laparoscopic skill. *Surgical Endoscopy*, 24(8):1824–1828, January 2010.

[BCF10] Jane Barnett, Mark Coulson, and Nigel Foreman. Examining player anger in world of warcraft. In *Online Worlds: Convergence of the Real and the Virtual*, pages 147–160. 2010.

[BCR11] Darrell Bethea, Robert A Cochran, and Michael K Reiter. Server-side verification of client behavior in online games. *Transactions on Information and System Security*, 14(4), December 2011.

[bE07] danah m boyd and Nicole B Ellison. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[BGB86] Francis S Bellezza, Anthony G Greenwald, and Mahzarin R Banaji. Words High and Low in Pleasantness as Rated by Male and Female College Students. *Behavior Research Methods, Instruments, & Computers*, 18(3):299–303, 1986.

[BGC10] Anthony Brew, Derek Greene, and Pádraig Cunningham. Using Crowdsourcing and Active Learning to Track Sentiment in Online Media. In *ECAI*, pages 145–150, 2010.

[BHKL06] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 44–54, New York, NY, USA, 2006. ACM.

[BI13a] Jeremy Blackburn and Adriana Iamnitchi. Relationships under the Microscope with Interaction-Backed Social Networks. In *The 1st International Conference on Internet Science*, 2013.

[BI13b] Jeremy Blackburn and Adriana I Iamnitchi. An Architecture for Collecting Longitudinal Social Data. In *IEEE International Conference on Communications Workshop on Beyond Social Networks: Collective Awareness*, pages 184–188, Budapest, Hungary, 2013.

[BK14] Jeremy Blackburn and Haewoon Kwak. STFU NOOB! Predicting Crowdsourced Decisions Toxic Behavior in Online Games. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, 2014.

[BKS+14] Jeremy Blackburn, Nicolas Kourtellis, John Skvoretz, Matei Ripeanu, and Adriana Iamnitchi. Cheating in online games: A social network perspective. *ACM Trans. Internet Technol.*, 13(3):9:1–9:25, May 2014.

[BL99] Margaret M Bradley and Peter J Lang. Affective Norms for English Words (ANEW): Instruction Manual and Affective Ratings. Technical report, Technical Report C-1, The Center for Research in Psychophysiology, University of Florida, 1999.

[BSK+12] Jeremy Blackburn, Ramanuja Simha, Nicolas Kourtellis, Xiang Zuo, Matei Ripeanu, John Skvoretz, and Adriana Iamnitchi. Branded with a Scarlet "C": Cheaters in a Gaming Social Network. In *Proceedings of the 21st International Conference on World Wide Web*, WWW'12, pages 81–90, New York, NY, USA, 2012. ACM.

[BSL+11] Jeremy Blackburn, Ramanuja Simha, Clayton Long, Xiang Zuo, Nicolas Kourtellis, John Skvoretz, and Adriana Iamnitchi. Cheaters in a gaming social network. *SIGMETRICS Performance Evaluation Review*, 39(3), December 2011.

[BTP14] Erin E Buckels, Paul D Trapnell, and Delroy L Paulhus. Trolls just want to have fun. *Personality and Individual Differences*, February 2014.

[Cas06] Edward Castronova. On the Research Value of Large Games: Natural Experiments in Norrath and Camelot. *Games and Culture*, 1(2):163–186, April 2006.

[CCLM09] Thomas Chesney, Iain Coyne, Brian Logan, and Neil Madden. Griefing in Virtual Worlds: Causes, Casualties and Coping Strategies. *Information Systems Journal*, 19(6):525–548, 2009.

[CDN09] Vivian Hsueh-Hua Chen, Henry Been-Lirn Duh, and Chiew Woon Ng. Players Who Play to Make Others Cry: The Influence of Anonymity and Immersion. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ACE '09, 2009.

[CEM07] Damon Centola, Víctor M Eguíluz, and Michael W Macy. Cascade dynamics of complex propagation. *Physica A: Statistical Mechanics and its Applications*, 374(1):449–456, 2007.

[CF07] Nicholas A Christakis and James H Fowler. The spread of obesity in a large social network over 32 years. *New England journal of medicine*, 357(4):370–379, 2007.

[CFS+10] Chris Chambers, Wu-Chang Feng, Sambit Sahu, Debanjan Saha, and David Brandt. Characterizing online games. *IEEE/ACM Transactions on Networking*, 18(3):899–910, 2010.

[CLW03] Mark Claypool, David LaPoint, and Josh Winslow. Network analysis of Counter-strike and Starcraft. In *Proceedings of the IEEE International Conference on Performance, Computing, and Communications*, pages 261–268, April 2003.

[CMW08] Scott E Carrell, Frederick V Malmstrom, and James E West. Peer Effects in Academic Cheating. *Journal of human resources*, 43(1):173–207, December 2008.

[Con07] Mia Consalvo. *Cheating: Gaining Advantage in Videogames*. MIT Press, Cambridge, Massachusetts, 1st, edition, 2007.

[CPWF07] Duen Horng Chau, Shashank Pandit, Samuel Wang, and Christos Faloutsos. Parallel crawling for online social networks. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 1283–1284, 2007.

[Cro11] Tim Cross. All the world's a game. http://www.economist.com/node/21541164, 2011.

[CWC+09] Edward Castronova, Dmitri Williams, Cuihua Shen, Rabindra Ratan, Li Xiong, Yun Huang, and Brian Keegan. As real as real? Macroeconomic behavior in a large-scale virtual world. *New Media & Society*, 11(5):685–707, August 2009.

[DC09] Henry Been-Lirn Duh and Vivian Hsueh Hua Chen. Cheating Behaviors in Online Gaming. *Online Communities and Social Computing*, 5621:1–7, May 2009.

[DD10] Peter S Dodds and Christopher M Danforth. Measuring the Happiness of Large-Scale Written Expression: Songs, Blogs, and Presidents. *Journal of Happiness Studies*, 11(4):441–456, 2010.

[DSN+11] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification. using game-design elements in non-gaming contexts. In *Conference on Human Factors in Computing Systems Extended Abstracts*, CHI EA '11, pages 2425–2428, New York, NY, USA, 2011. ACM.

[Dum11] Delia D Dumitrica. An exploration of cheating in a virtual gaming world. *Journal of Gaming & Virtual Worlds*, 3(1):21–36, March 2011.

[DW05] Peter S Dodds and Duncan J Watts. A generalized model of social and biological contagion. *Journal of Theoretical Biology*, 232(4):587–604, February 2005.

[FCFW02] Wu-Chang Feng, Francis Chang, Wu-chi Feng, and Jonathan Walpole. Provisioning on-line games: a traffic analysis of a busy counter-strike server. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, pages 151–156, New York, NY, USA, 2002. ACM.

[FCFW05] W. Feng, F. Chang, W. Feng, and J. Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Transactions on Networking*, 13:488–500, 2005.

[FF03] Wu-Chang Feng and Wu-chi Feng. On the Geographic Distribution of Online Game Servers and Players. In *Proceedings of the 2nd Workshop on Network and System Support for Games*, Netgames '03, pages 173–179, New York, New York, USA, 2003. ACM Press.

[FK04] Chek Yang Foo and Elina M. I. Koivisto. Defining Grief Play in MMORPGs: Player and Developer Perceptions. In *ACE '04*, ACE '04, 2004.

[FMDD13] Morgan R Frank, Lewis Mitchell, Peter S Dodds, and Christopher M Danforth. Happiness and the Patterns of Life: A Study of Geolocated Tweets. *Scientific Reports*, 3(2625), 2013.

[GAA09] Francesca Gino, Shahar Ayal, and Dan Ariely. Contagion and Differentiation in Unethical Behavior: The Effect of One Bad Apple on the Barrel. *Psychological Science*, 20(3):393–398, March 2009.

[Gar11] Gartner. Gartner says by 2015, more than 50 percent of organizations that manage innovation processes will gamify those processes. http://www.gartner.com/newsroom/id/1629214, 2011.

[GDB06] Thilo Gross, Carlos J. Dommar D'Lima, and Bernd Blasius. Epidemic Dynamics on an Adaptive Network. *Phys. Rev. Lett.*, 96:208701, May 2006.

[Gil10] Jim Giles. Inside the race to hack the kinect. *New Scientist*, 208(2789):22 – 23, 2010.

[Gla11] René Glas. Breaking reality: exploring pervasive cheating in Foursquare. *Proceedings of DiGRA 2011*, pages 1–15, 2011.

[GM11] Scott A Golder and Michael W Macy. Diurnal and Seasonal Mood Vary with Work, Sleep, and Daylength across Diverse Cultures. *Science*, 333(6051):1878–1881, 2011.

[Gol08] Alan Goldman. Company on the Couch: Unveiling Toxic Behavior in Dysfunctional Organizations. *Journal of Management Inquiry*, 17(3):226–238, September 2008.

[Hal12] Halotags. Halotags halo 3 & halo 4 mlg level 50 accounts for sale. http://www.halotags.com, 2012.

[HHLD11] J Han, E Haihong, G Le, and J Du. IEEE Xplore Full-Text PDF:. *Pervasive computing and ...*, 2011.

[HHvdWZ08] Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel, and Achim Zeileis. Implementing a class of permutation tests: The coin package. *Journal of Statistical Software*, 28(8):1–23, 2008.

[HL10] J. Hamari and V. Lehdonvirta. Game design as marketing: How game mechanics create demand for virtual goods. *International Journal of Business Science & Applied Management*, 5(1):14–29, 2010.

[HLR11] W. He, X. Liu, and M. Ren. Location cheating: A security challenge to location-based social network services. In *Proceedings of 31st International Conference on Distributed Computing Systems*, ICDCS '11, pages 740–749. IEEE, 2011.

[HM08] Shirley S Ho and Douglas M McLeod. Social-Psychological Influences on Opinion Expression in Face-to-Face and Computer-Mediated Communication. *Communication Research*, 35(2):190–207, 2008.

[HMG07] R. Herbrich, T. Minka, and T. Graepel. Trueskill: A bayesian skill rating system. *Advances in Neural Information Processing Systems*, 19:569, 2007.

[Hui50] Johan Huizinga. *Home Ludens: A Study of the Play-Element in Culture*. Beacon Press, Boston, Massachusetts, reprint edition, 1950.

[Inc12a] Fitocracy Inc. Fitocracy. http://www.fitocracy.com, 2012.

[Inc12b] Foursquare Labs Inc. Foursquare. http://www.foursquare.com, 2012.

[KAA+13] Zuhair Khayyat, Karim Awara, Amani Alonazi, Hani Jamjoom, Dan Williams, and Panos Kalnis. Mizan: a system for dynamic load balancing in large-scale graph processing. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13. ACM, April 2013.

[KAS+12] Nicolas Kourtellis, Tharaka Alahakoon, Ramanuja Simha, Adriana Iamnitchi, and Rahul Tripathi. Identifying high betweenness centrality nodes in large social networks. *Social Network Analysis and Mining*, pages 1–16, 2012.

[KAW+10] B Keegan, M.A Ahmed, D Williams, J Srivastava, and N Contractor. Dark Gold: Statistical Properties of Clandestine Networks in Massively Multiplayer Online Games. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 201–208, August 2010.

[KCWC07] Y. Ku, Y. Chen, K. Wu, and C. Chiu. An empirical analysis of online gaming crime characteristics from 2002 to 2004. In *PAISI 07*, pages 34–45, 2007.

[KH] Haewoon Kwak and Seungyeop Han. "So Many Bad Guys, So Little Time": Understanding Toxic Behavior and Reaction in Team Competition Games. Submitted.

[KKR10] Simi Kedia, Kevin Koh, and Shivaram Rajgopal. Evidence on Contagion in Corporate Misconduct. *SSRN Electronic Journal*, 2010.

[KLPM10] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 591–600, New York, New York, USA, 2010. ACM Press.

[KOS08] Brian W Kulik, Michael J O'Fallon, and Manjula S Salimath. Do Competitive Environments Lead to the Rise and Spread of Unethical Behavior? Parallels from Enron. *Journal of Business Ethics*, 83(4):703–723, January 2008.

[KTCB05] Patric Kabus, Wesley W Terpstra, Mariano Cilia, and Alejandro P Buchmann. Addressing cheating in distributed MMOGs. In *Proceedings of the 4th ACM SIGCOMM Workshop on Network and System Support for Games*, Netgames '05, pages 1–6, New York, New York, USA, 2005. ACM Press.

[Kus10] David Kushner. Steamed: Valve software battles video-game cheaters. http://goo.gl/wWgq6n, 2010.

[LC11] Richard N. Landers and Rachel C. Callan. Casual social games as serious games: The psychology of gamification in undergraduate education and employee training. *Serious Games and Edutainment Applications*, pages 399–423, 2011.

[LNX+05] Ira M Longini, Azhar Nizam, Shufu Xu, Kumnuan Ungchusak, Wanna Hanshaoworakul, Derek A T Cummings, and M Elizabeth Halloran. Containing Pandemic Influenza at the Source. *Science*, 309(5737):1083–1087, 2005.

[LPC+13] Jing-Kai Lou, Kunwoo Park, Meeyoung Cha, Juyong Park, Chin-Laung Lei, and Kuan-Ta Chen. Gender Swapping and User Behaviors in Online Social Games. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW'13, pages 827–836, 2013.

[LS05] Holin Lin and Chuen-Tsai Sun. The "White-Eyed" Player Culture: Grief Play and Construction of Deviance in MMORPGs. In *DiGRA '05*, 2005.

[LZ12] RJ Lin and X. Zhu. Leveraging social media for preventive care-a gamification system and insights. *Studies in health technology and informatics*, 180:838, 2012.

[MAB+10] Grzegorz Malewicz, Matthew H Austern, Aart J C Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146, New York, NY, USA, 2010. ACM.

[MB00] Katelyn Y. A. McKenna and John A. Bargh. Plan 9 From Cyberspace: The Implications of the Internet for Personality and Social Psychology. *Personality and Social Psychology Review*, 4(1):57–75, 2000.

[MC13] Winter Mason and Aaron Clauset. Friends FTW! Friendship and Competition in Halo: Reach. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 375–386, New York, NY, USA, 2013. ACM.

[MDF11] Sam Moffatt, Akshay Dua, and Wu-Chang Feng. SpotCheck: an efficient defense against information exposure cheats. In *Proceedings of the 10th Annual Workshop on Network and Systems Support for Games*, pages 8:1–8:6, Piscataway, NJ, USA, 2011. IEEE Press.

[MFE07] Jason Mitchell, Moby Francke, and Dhabih Eng. Illustrative rendering in Team Fortress 2. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 71–76, New York, NY, USA, 2007. ACM.

[MFH+13] Lewis Mitchell, Morgan R Frank, Kameron Decker Harris, Peter Sheridan Dodds, and Christopher M Danforth. The Geography of Happiness: Connecting Twitter Sentiment and Expression, Demographics, and Objective Characteristics of Place. *PloS one*, 8(5):e64417, 2013.

[MPK03] Jessica Mulligan, Bridgette Patrovsky, and Raph Koster. *Developing Online Games: An Insider's Guide*. Pearson Education, 2003.

[NRCS10] Atif Nazir, Saqib Raza, Chen-Nee Chuah, and Burkhard Schipper. Ghost-busting facebook: detecting and characterizing phantom profiles in online social gaming applications. *Proceedings of the 3rd Conference on Online Social Networks*, June 2010.

[Olw96] Dan Olweus. Bullying at School: Long Term Outcomes for the Victims and an Effective School-based Intervention Program. *Aggressive Behavior: Current Perspectives*, pages 97–130, 1996.

[OSH+07] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A.-L. Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, 2007.

[Par84] D Parfit. *Reasons and Persons*. Oxford University Press, 1984.

[Pel10] Kathie L Pelletier. Leader toxicity: An empirical investigation of toxic behavior and rhetoric. *Leadership*, 6(4):373–389, November 2010.

[PKZ12] K Pelechrinis, P Krishnamurthy, and K Zhang. Gaming the Game: Honeypot Venues Against Cheaters in Location-based Social Networks. *arXiv*, 2012.

[QBYL10] Alexander J Quinn, Benjamin B Bederson, Tom Yeh, and Jimmy Lin. CrowdFlow: Integrating Machine Learning with Mechanical Turk for Speed-Cost-Quality Flexibility. *Better Performance Over Iterations*, 2010.

[RBR11] Nicolas Ruflin, Helmar Burkhart, and Sven Rizzotti. Social-data storage-systems. In *Databases and Social Networks*, pages 7–12, New York, New York, USA, 2011. ACM Press.

[Re-11] Re-logic. Terraria. http://store.steampowered.com/app/105600/, 2011.

[RK09] David Rettinger and Yair Kramer. Situational and Personal Causes of Student Cheating. *Research in Higher Education*, 50:293–313, 2009.

[RTL12] Zhongyuan Ruan, Ming Tang, and Zonghua Liu. Epidemic spreading with information-driven vaccination. *Physical Review E*, 86:036117, September 2012.

[Sek11] Jasjeet S. Sekhon. Multivariate and propensity score matching software with automated balance optimization: The matching package for r. *Journal of Statistical Software*, 42(7):1–52, 6 2011.

[Sen11] Tom Senior. Steam controls up to 70% of the pc download market and is "tremendously profitable". http://goo.gl/MjgA3, 2011.

[SI11] Siqi Shen and A Iosup. The XFire online meta-gaming network: observation and high-level analysis. In *Haptic Audio Visual Environments and Games (HAVE), 2011 IEEE International Workshop on*, 2011.

[SKP12] D. Souravlias, G. Koloniari, and E. Pitoura. Intersocialdb: An infrastructure for managing social data. In *Intersocial Workshop on Online Social Networks: Challenges and Perspectives*, 2012.

[SMC+08] Peter K Smith, Jess Mahdavi, Manuel Carvalho, Sonja Fisher, Shanette Russell, and Neil Tippett. Cyberbullying: Its Nature and Impact in Secondary School Pupils. *Journal of Child Psychology and Psychiatry*, 49(4):376–385, 2008.

[SMML10] Salvatore Scellato, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. Distance matters: geo-social metrics for online social networks. *WOSN'10: Proceedings of the 3rd conference on Online social networks*, June 2010.

[SR03] Mani R Subramani and Balaji Rajagopalan. Knowledge-sharing and influence in online social networks via viral marketing. *Communications of the ACM*, 46(12), December 2003.

[ST10] M. Szell and S. Thurner. Measuring social dynamics in a massive multiplayer online game. *Social Networks*, 32:313–329, 2010.

[ST11] C R Shalizi and A C Thomas. Homophily and Contagion Are Generically Confounded in Observational Social Network Studies. *Sociological Methods & Research*, 40(2):211–239, May 2011.

[TM69] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):pp. 425–443, 1969.

[TMD12] Jennifer Thom, David Millen, and Joan DiMicco. Removing gamification from an enterprise sns. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, CSCW '12, pages 1067–1070, New York, NY, USA, 2012. ACM.

[TMW+12] Anthony Tang, Jonathan Massey, Nelson Wong, Derek Reilly, and W. Keith Edwards. Verbal coordination in first person shooter games. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 579–582, New York, NY, USA, 2012. ACM.

[Tre11] Trendy Entertainment. Dungeon Defenders. http://store.steampowered.com/app/65800/, 2011.

[TSSW10] Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. In *ICWSM '10*, pages 178–185, 2010.

[Val90] Leslie G Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, August 1990.

[Val12a] Valve. Steam community update now live. http://store.steampowered.com/news/8788/, September 2012.

[Val12b] Valve. Steam: Game and player statistics. http://store.steampowered.com/stats, 2012.

[Val12c] Valve. Team Fortress 2: History. http://www.teamfortress.com/history.php, 2012.

[VC13] Irene Serrano Vázquez and Mia Consalvo. Cheating in social network games. *New Media & Society*, December 2013.

[WBS+09] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P. N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. In *Proc. of ACM EuroSys*, Nuremberg, Germany, April 2009.

[WC13] Yuehua Wu and Vivian Hsueh Hua Chen. A social-cognitive approach to online game cheating. *Computers in Human Behavior*, 29(6):2557–2567, November 2013.

[Wei80] Bernard Weiner. A Cognitive (Attribution)-Emotion-Action Model of Motivated Behavior: An Analysis of Judgments of Help-Giving. *Journal of Personality and Social Psychology*, 39(2):186, 1980.

[Wel88] B. Wellman. *Structural Analysis: From Method and Metaphor to Theory and Substance*. Cambridge University Press, 1988.

[WH12] K. Werbach and D. Hunter. *For the Win: How Game Thinking Can Revolutionize Your Business*. Wharton Digital Press, 2012.

[WR05] D. E Warner and M. Ratier. Social Context in Massively-Multiplayer Online Games (MMOGs): Ethical Questions in Shared Space. *International Review of Information Ethics*, 4(7), 2005.

[WRT+10] Walter Willinger, Reza Rejaie, Mojtaba Torkjazi, Masoud Valafar, and Mauro Maggioni. Research on online social networks. *ACM SIGMETRICS Performance Evaluation Review*, 37(3):49, January 2010.

[WSPZ12] Christo Wilson, Alessandra Sala, Krishna P N Puttaswamy, and Ben Y Zhao. Beyond Social Graphs: User Interactions in Online Social Networks and their Implications. *Transactions on the Web (TWEB*, 6(4), November 2012.

[XCS+11] Yan Xu, Xiang Cao, Abigail Sellen, Ralf Herbrich, and Thore Graepel. Sociable killers: understanding social relationships in an online first-person shooter game. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, CSCW '11, pages 197–206, New York, NY, USA, 2011. ACM.

Appendices

Appendix A: Statistical Tests

We used two tests of statistical significance when comparing distributions in this paper: the two sample Kolmogorov-Smirnov (KS) test and the permutation test.

The KS test is nonparametric and uses the supremum (least upper bound) of the set of distances between each point of two empirical distribution functions for its test statistic (D). Unfortunately, while the KS test is distribution agnostic (e.g., the data does not have to be normally distributed) it operates only on *continuous* distributions with no ties. Fortunately, the KS test can be bootstrapped to avoid these issues [Sek11]. For all reporting in this paper we used 1,000 bootstrap samples.

A permutation test, on the other hand, uses random reshuffles of the data to get the distribution of a test statistic under a null hypothesis. In our case the null hypothesis is that of no difference between two groups, cheaters and noncheaters, and the random reassignment of these labels to the elements of the data vector followed by the calculation of the statistic of interest yields a distribution of this statistic, when the reassignment is done many times, against which its observed value can be compared.

The permutation method, also known as a randomization test, is a straight forward and intuitive approach. Consider a 2 column vector representing two distributions where the first column is a label identifying the distribution (e.g,. "cheaters" or "noncheaters") to which the value in the second column belongs. We then calculate a test statistic T[1].

Operating under the null hypothesis that the labels are meaningless (and thus the distributions are the same), we randomly permute the labels and compute a new test statistic

[1]In theory, *any* appropriate test statistic can be used, but we use a standardized linear statistic as described in [HHvdWZ08]

Appendix A: (continued)

T_i. Due to the random labeling, if we perform the permutation repeatedly, our T_is should be uniformly distributed, and thus the null hypothesis would have the original T statistic appearing anywhere in the ordered distribution of T_is with equal probability.

If we perform the permutation M times, we can then calculate a p value as the fraction of permutations where $T_i \geq T$. In other words,

$$p_{permute} = \frac{1}{M} \sum_{i=0}^{M} I(T_i \geq T)$$

(where I is the indicator function returning 1 if its argument is true, and 0 otherwise). For all reporting in this paper we use $M = 100{,}000$.

Appendix B: Copyright Permissions

The following are copyright permissions for the use of materials in Chapters 4, 5, 7, and 8.

2.5 Permanent Rights held by original Owners/Authors

The original Owner/Author permanently holds these rights:

- All other proprietary rights not granted to ACM, including patent or trademark rights.
- Reuse of any portion of the Work, without fee, in any future works written or edited *by the Author***, including books, lectures and presentations in any and all media.
- Create a "Major Revision" which is *not* subject to any rights in the original that have been granted to ACM
- Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, or (3) any repository legally mandated by an agency funding the research on which the Work is based.
- Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;
- Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version") to non-peer reviewed servers;
- Make distributions of the final published Version of Record internally to the Owner's employees, if applicable;
- Bundle the Work in any of Owner's software distributions; and
- Use any Auxiliary Material independent from the Work.