8-2011

Click fraud : how to spot it, how to stop it?

Chamila Kumara Walgampaya
University of Louisville

CLICK FRAUD: HOW TO SPOT IT, HOW TO STOP IT?

By

Chamila Kumara Walgampaya
BSc(Eng), University of Peradeniya, 2001
MS, University of Louisville, 2006

A Dissertation Proposal Submitted to the
Faculty of the Graduate School of the
University of Louisville

Computer Engineering and Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, Kentucky

August, 2011

CLICK FRAUD: HOW TO SPOT IT, HOW TO STOP IT?

By

Chamila Kumara Walgampaya
BSc(Eng), University of Peradeniya, 2001
MS, University of Louisville, 2006

A Dissertation Approved on
May 27th, 2011
by the following Dissertation Committee:

Dr. Mehmed Kantardzic (Dissertation Director)

Dr. Adel Elmaghraby

Dr. Dar-jen Chang

Dr. Roman Yampolskiy

Dr. Jeffrey Hieb

DEDICATION

To Thulani

ACKNOWLEDGMENTS

I have the pleasure of expressing my appreciation and gratitude for those who have helped me in preparing this dissertation.

First of all, I would like to thank my supervisor Dr. Mehmed Kantardzic, for his guidance and continuous encouragement. He inspired many of the ideas presented in this dissertation. He is a superb teacher and a great friend.

Secondly, I want to thank my other committee members Dr. Adel Elmaghraby, Dr. Roman Yampolskiy, Dr. Dar-jen Chang, and Dr. Jeffrey Hieb for their comments and feedback.

I thank all the faculty members and the staff of the Computer Science and Computer Engineering Department and the University of Louisville. I also thank all my friends, especially colleagues in the data mining lab.

I am thankful to Mr. Sean Higgins at Nomad LLC and Mr. Darren King at Hosting.com for their generous support.

Finally and most importantly, I would like to thank my parents for their efforts to provide me with the best possible education and my brother, sister, wife, and my wonderful daughter for continuous encouragements at difficult moments.

ABSTRACT

CLICK FRAUD: HOW TO SPOT IT, HOW TO STOP IT?

Chamila Walgampaya

May 27th, 2011

Online search advertising is currently the greatest source of revenue for many Internet giants such as Google^{TM}, Yahoo!^{TM}, and Bing^{TM}. The increased number of specialized websites and modern profiling techniques have all contributed to an explosion of the income of ad brokers from online advertising. The single biggest threat to this growth is however *click fraud.* Trained botnets and even individuals are hired by click-fraud specialists in order to maximize the revenue of certain users from the ads they publish on their websites, or to launch an attack between competing businesses.

Most academics and consultants who study online advertising estimate that 15% to 35% of ads in pay per click(PPC) online advertising systems are not authentic. In the first two quarters of 2010, US marketers alone spent \$5.7 billion on PPC ads, where PPC ads are between 45 and 50 percent of all online ad spending. On average about \$1.5 billion is wasted due to click-fraud. These fraudulent clicks are believed to be initiated by users in poor countries, or botnets, who are trained to click on specific ads. For example, according to a 2010 study from Information Warfare Monitor, the operators of Koobface, a program that installed malicious software to participate in click fraud, made over \$2 million in just over a year. The process of making such illegitimate clicks to generate revenue is called click-fraud.

Search engines claim they filter out most questionable clicks and either not charge for them or reimburse advertisers that have been wrongly billed. However this is a hard task, despite the claims that brokers' efforts are satisfactory. In the simplest scenario, a publisher continuously clicks on the ads displayed on his own website in order to make revenue. In a more complicated scenario, a travel agent may hire a large, globally distributed, botnet to click on its competitor's ads, hence depleting their daily budget. We analyzed those different types of click fraud methods and proposed new methodologies to detect and prevent them real time. While traditional commercial approaches detect only some specific types of click fraud, Collaborative Click Fraud Detection and Prevention (CCFDP) system, an architecture that we have implemented based on the proposed methodologies, can detect and prevents all major types of click fraud.

The proposed solution analyzes the detailed user activities on both, the server side and client side collaboratively to better describe the intention of the click. Data fusion techniques are developed to combine evidences from several data mining models and to obtain a better estimation of the quality of the click traffic. Our ideas are experimented through the development of the Collaborative Click Fraud Detection and Prevention (CCFDP) system. Experimental results show that the CCFDP system is better than the existing commercial click fraud solution in three major aspects: 1) detecting more click fraud especially clicks generated by software; 2) providing prevention ability; 3) proposing the concept of click quality score for click quality estimation.

In the CCFDP initial version, we analyzed the performances of the click fraud detection and prediction model by using a rule base algorithm, which is similar to most of the existing systems. We have assigned a quality score for each click instead of classifying the click as fraud or genuine, because it is hard to get solid evidence of click fraud just based on the data collected, and it is difficult to determine the real intention of users who make the clicks.

Results from initial version revealed that the diversity of CF attack types makes it hard for a single counter measure to prevent click fraud. Therefore, it is important to be able to combine multiple measures capable of effective protection from click fraud. Therefore, in the CCFDP improved version, we provide the traffic quality score as a combination of evidence from several data mining algorithms.

We have tested the system with a data from an actual ad campaign in 2007 and 2008. We have compared the results with Google Adwords reports for the same campaign. Results show that a higher percentage of click fraud present even with the most popular search engine. The multiple model based CCFDP always estimated less valid traffic compare to Google. Sometimes the difference is as high as 53%.

Detection of duplicates, fast and efficient, is one of the most important requirement in any click fraud solution. Usually duplicate detection algorithms run in real time. In order to provide real time results, solution providers should utilize data structures that can be updated in real time. In addition, space requirement to hold data should be minimum. In this dissertation, we also addressed the problem of detecting duplicate clicks in pay-per-click streams. We proposed a simple data structure, Temporal Stateful Bloom Filter (TSBF), an extension to the regular Bloom Filter and Counting Bloom Filter. The bit vector in the Bloom Filter was replaced with a status vector. Duplicate detection results of TSBF method is compared with Buffering, FPBuffering, and CBF methods. False positive rate of TSBF is less than 1% and it does not have false negatives. Space requirement of TSBF is minimal among other solutions. Even though Buffering does not have either false positives or false negatives its space requirement increases exponentially with the size of the stream data size. When the false positive rate of the FPBuffering is set to 1% its false negative rate jumps to around 5%, which will not be tolerated by most of the streaming data applications. We also compared the TSBF results with CBF. TSBF uses only half the space or less than standard CBF with the same false positive probability.

One of the biggest successes with CCFDP is the discovery of new mercantile click bot, the Smart ClickBot. We presented a Bayesian approach for detecting the Smart ClickBot type clicks. The system combines evidence extracted from web server sessions to determine the final class of each click. Some of these evidences can be used alone, while some can be used in combination with other features for the click bot detection. During training and testing we also addressed the class imbalance problem. Our best classifier shows recall of 94%, and precision of 89%, with F_1 measure calculated as 92%. The high accuracy of our system proves the effectiveness of the proposed methodology. Since the Smart ClickBot is a sophisticated click bot that manipulate every possible parameters to go undetected, the techniques that we discussed here can lead to detection of other types of software bots too.

Despite the enormous capabilities of modern machine learning and data mining techniques in modeling complicated problems, most of the available click fraud detection systems are rule-based. Click fraud solution providers keep the rules as a secret weapon and bargain with others to prove their superiority. We proposed validation framework to acquire another model of the clicks data that is not rule dependent, a model that learns the inherent statistical regularities of the data. Then the output of both models is compared.

Due to the uniqueness of the CCFDP system architecture, it is better than current commercial solution and search engine/ISP solution. The system protects Pay-Per-Click advertisers from click fraud and improves their Return on Investment (ROI). The system can also provide an arbitration system for advertiser and PPC publisher whenever the click fraud argument arises. Advertisers can gain their confidence on PPC advertisement by having a channel to argue the traffic quality with big search engine publishers. The results of this system will booster the internet economy by eliminating the shortcoming of PPC business model. General consumer will gain their confidence on internet business model by reducing fraudulent activities which are numerous in current virtual internet world.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

ONLINE ADVERTISING

1.1 Introduction

The Internet is probably the most important technological creation of our times. It is a universally acknowledged truth that the world wide web has revolutionized our planet. Today the web allows us to communicate with people nearly anywhere in the world and indeed, with billions of people at once. It has changed the way we think, as well as the way we do business. It provides many immensely useful services to the masses for free, including such essentials as web portals, web email and web search [Immorlica et al., 2005].

The web is not only a means of communication, it also contains a wealth of knowledge. Since the rise of the search engines, huge amounts of information have become readily accessible. Web search engines provide information access to millions of users per day. For many people, Web search engines are now the primary method for finding information, news, and products. According to a recent study 99% of Internet users utilize search, while the known web alone spans several billion pages [iProspect, 2004]. In addition to addressing information requests, modern web search engines are navigational tools that take people to specific websites or are an aid in browsing. People also employ search engines in new and increasingly diverse ways, and search engines are constantly trying to improve the retrieval aspects of their services.

Most of the information and services on the web come at no cost to the user. The publisher or webmaster, on the other hand, has to pay to provide them. In order to profit from their complimentary services, webmasters rent out space on their websites to advertisers. This is currently the main means of covering the costs of providing online services [Soubusta, 2008]. Advertising plays an important role for every company. Most businesses operate with an advertising budget of 2 to 5 percent of their previous year's gross sales. Since its inception, Internet advertising has grown rapidly and negatively affected traditional mass media advertising like newspapers and television. Table 1.1 shows the variation of advertisement budget in for each media [TNS, 2010]. Spending on Internet ads is growing faster than any other sector of the advertising industry and is expected to surge from $12.5 billion in 2009 to $29 billion in 2010 in the US alone, according to researcher eMarketer Inc.

Table 1.1: Advertising Expenditure of all Media

Media Type	2008 Vs. 2007	2009 Vs. 2008
Internet	4.6%	7%
TV	0.1%	-12.1%
Magazine	-7.5%	-19.7%
Paper	-11.8%	-22.8%
Radio	-10.3%	-22.8%
Outdoor	-1.5%	-16.2%

As the Internet continues to grow, the Internet advertising industry flourishes as a means of reaching the appropriate market segments. Internet advertising is different from conventional TV/Radio advertising, which tends to be expensive, broadcast-based, and diluted. On the other hand, Internet advertisers are currently able to inexpensively direct their campaigns to the appropriate audience. Because information flow through cyberspace is electronic, online marketers can selectively advertise to consumers at high volume with costs far below that of older media such as television and radio. Current data management and Web programming technologies allow the classification of surfers and thus allow the targeting of advertisements to the appropriate customer body on the fly [Metwally et al., 2006].

Internet advertising comes in various forms. For example as a result of a search query, banners, multimedia etc. Figure 1.1 shows the variation of revenue shared by advertising format from year 2004 to 2009. It is obvious that "search query ads" are responsible for the largest market share in Internet advertising and it is continuously growing [IAB, 2010].

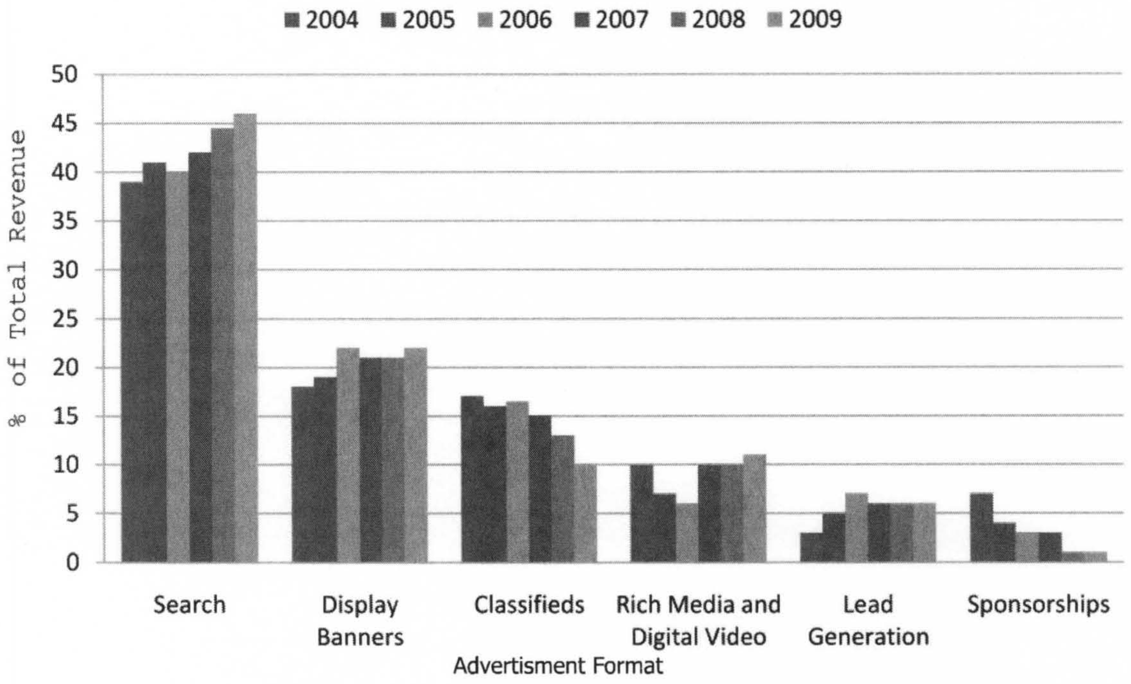


Figure 1.1: Variation of Revenue Shared by Advertising Format

Typical search engine queries are short and reveal a great deal of information about user preferences. This gives search engine companies a unique opportunity to display highly targeted ads to the user [Mehta et al., 2007]. When a user types a query or search keywords, major search engines offer at least two types of results on a search engine results page. One set is composed of *organic* links that were determined using the search engines matching algorithm. The other set is composed of *paid* links that appear because a company purchased the keyword(s) used in the search query.

The first question comes into mind is, if we are able achieve the top organic positions of the favorite (or most valued) keywords, we can eliminate or at least try to minimize money paying for sponsored search. It sounds very tempting, and getting top organic positions does have a huge incremental value. This is particularly true if the page that ranks well contains a clear offer leading to a good conversion rate and site stickiness.

However, the ability to achieve top organic placement above any universal search element is increasingly in doubt, except perhaps for a brand search. Also consider that search engine results pages are no longer a "one size fits all" scenario. Therefore, sponsored results play a vital role in advertising the business to the user [Lee, 2009].

1.2 Online Advertising Models, CPM, CPA, and PPC

In the past, online advertising used the *Cost-Per-Impression*(CPI or CPM) model to charge for advertisements. An online advertisement impression is a single appearance of an advertisement on a web page. Each time an advertisement loads onto a user's screen, the ad server may count that loading as one impression. The cost per impression is often measured in *Cost-Per-Mille* (CPM), that is, the cost of one thousand impressions of the ad. This advertising model was based on traditional TV and print advertising, where the advertiser is charged on the basis of the number of times that the ad is viewed. Cost-Per-Impression is the preferred model of publishers (webmasters), because they are paid regardless of the effectiveness of an ad. Obviously, advertisers would prefer to pay only for ads that lead to a *conversion*.

Successful conversions are interpreted differently by individual marketers, advertisers, and content creators. To online retailers, for example, a successful conversion may constitute the sale of a product to a consumer whose interest in the item was initially sparked by clicking a banner advertisement. To content creators, however, a successful conversion may refer to a membership registration, newsletter subscription, software download, or other activity that occurs due to a subtle or direct request from the content creator for the visitor to take the action.

In internet marketing, conversion rate is the ratio of visitors who convert casual content views or website visits into desired actions based on subtle or direct requests from marketers, advertisers, and content creators. The Conversion rate is defined as follows:

$$Conversion\ rate = \frac{Number\ of\ Goals\ achieved}{Total\ Visits} \tag{1.1}$$

Casale Media^{TM}, Burst Media^{TM}, Value Click^{TM}, and Tribal Fusion^{TM} are the popular advertising companies that provide CPM based advertising.

Advertisers have an advantageous position in the *Cost Per Action* model or CPA (sometimes known as Pay Per Action or PPA). It is an online advertising pricing model, where the advertiser pays for each specified action (a purchase, a form submission, and so on) linked to the advertisement.

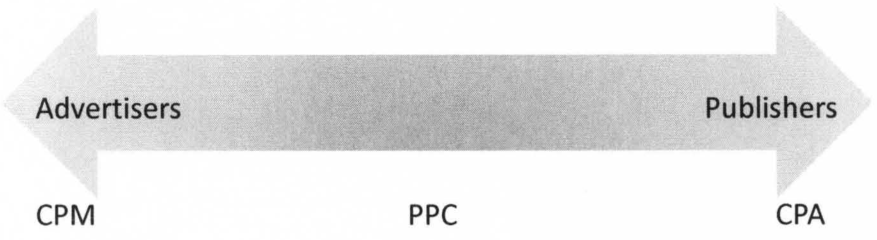


Figure 1.2: CPM, CPA vs. PPC

1.3 Pay-Per-Click Model

Pay-Per-Click(PPC) model was developed by Google in 2004 as a balance between CPM CPA methods. It has continued to attract more and more businesses and Figure 1.3 shows the growth of the PPC in the recent years.

In this type of arrangement, advertisers pay a certain amount of money to the publisher for every click on their ad (which leads to the advertiser's website). That is to say costs are performance dependent. Consequently, the model is still favored by advertisers, who get an interested visitor to their website for the money they pay. Sponsored search owes much of its success to the pay-per-click performance advertising model. In traditional advertising, product advertisers target an audience and pay for each impression. In sponsored search, advertisers target advertisements at search keywords, but only pay if a user actually engages by clicking on the offered link. The close coupling between payment and an easily measurable performance metric creates an unrivaled performance marketing environment [Pedersen, 2008].

A typical PPC traffic model is depicted in Figure 1.4. An Advertiser can be a com-

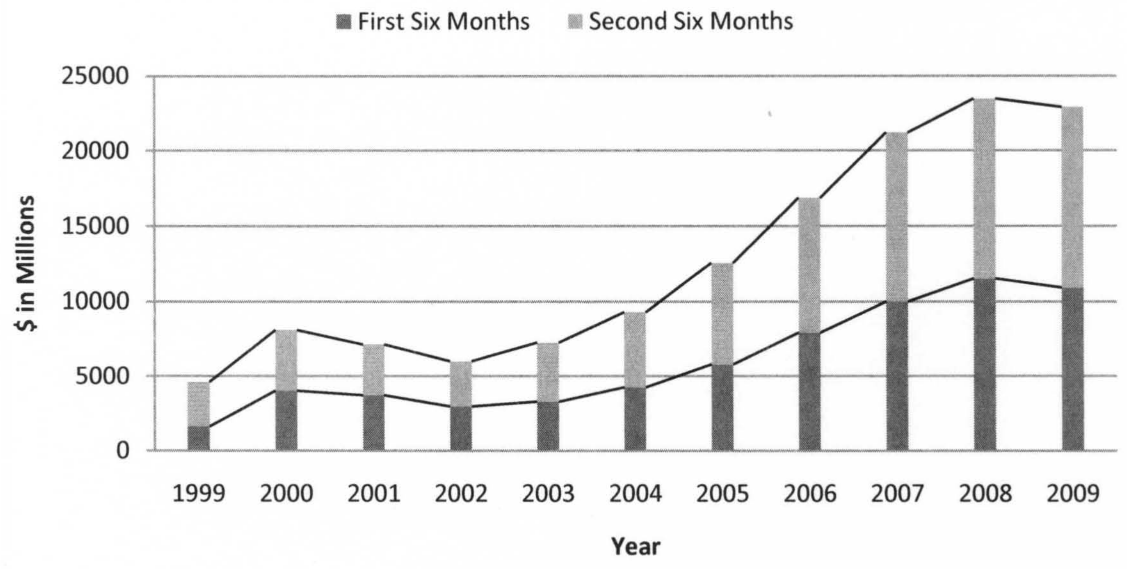


Figure 1.3: Growth of PPC model

pany or an individual who would like to display their advertisements in other websites. Publishers are the websites that accept contracts through advertisers to display advertisements. The commissioner is an independent entity that has agreements with both the advertiser and the publisher. It can be a search engine or other advertisement agent. The commissioner keeps track of the advertisers' budgets so that they are not over-spent. Often, the advertiser site does not administer the advertisement pay-per-click model itself, for example referred traffic by other sites, but rather employs a third party ad network, here referred to as the click fraud detection service, to administer the pay-per-click or click-through program on its behalf.

The web user can be a human or software, possibly the source of click fraud that uses services of the Internet. A user click propagates through the model as described below.

1. First a Web user requests a web page in the publisher site. The requested page is loaded along with the advertisement on the Web users' browser.

2. If the Web user clicks on an advertisement hypertext link (for example a banner ad or logo) in that page, the publisher redirects the Web user request to commissioner's server. The commissioner logs the click for accounting purposes.

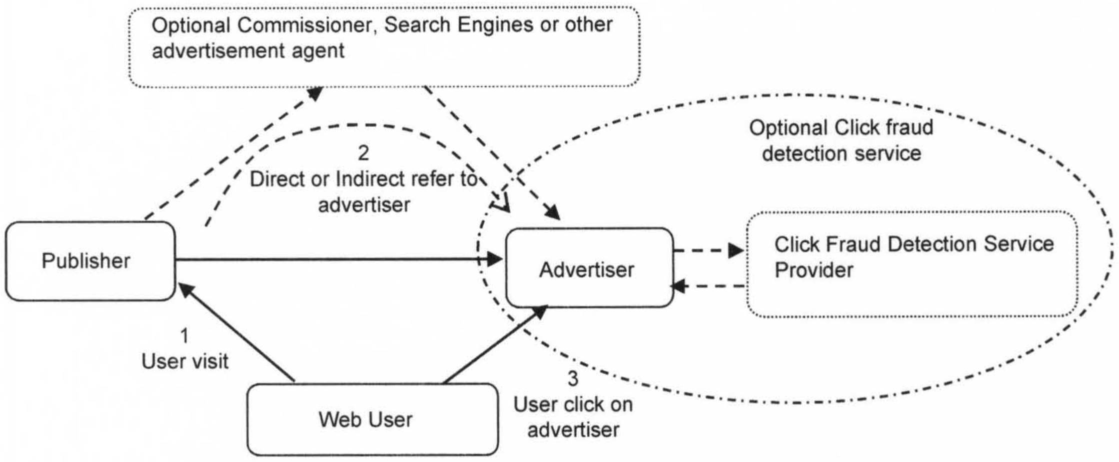

Optional Commissioner, Search Engines or other advertisement agent
Optional Click fraud detection service
2
Direct or Indirect refer to advertiser
Publisher
Advertiser
Click Fraud Detection Service Provider
1
User visit
3
User click on advertiser
Web User

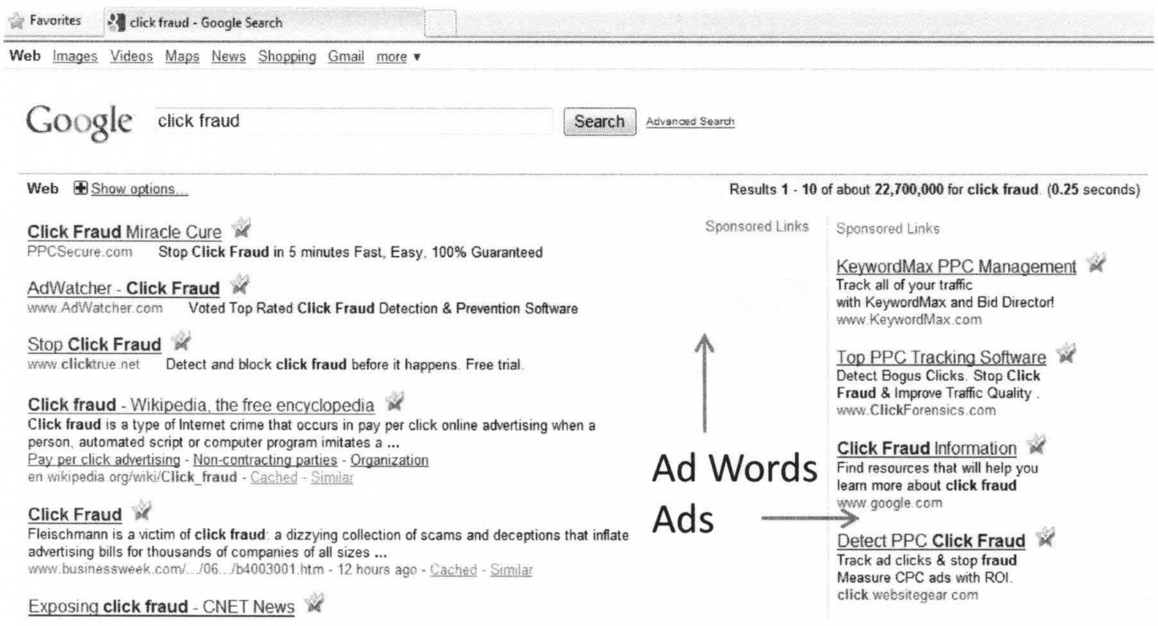
Figure 1.4: PPC Traffic model

3. The commissioner then redirects the Web user's browser to the advertiser site.

The publishers are paid based on the click traffic they drive to the advertiser's web site. The commissioner earns a percentage of this revenue. Sometimes these payments are based on number of sales generated in the advertiser's website, rather than the volume of traffic driven by the publisher.

Pay-per-click metering is a popular payment model for advertising on the Internet. The model involves and advertiser who contracts with a specialized entity, which we refer to as a syndicator, to distribute textual or graphical banner advertisements to publishers of content. These banner ads point to the advertiser's web site. When a user clicks on the banner ad on the publisher's webpage, she is directed to the site to which it points. Search engines such as Google and Yahoo are the most popular syndicators, and create the largest portion of pay-per-click traffic on the Internet today. These sites display advertisements on their own search pages in response to the search terms entered by users and charge advertiser for clicks on these links (thereby acting as their own publishers) or, increasingly, outsource advertisements to third-party publishers. Advertisers pay syndicators per referral, and the syndicators pass on a portion of the payments to the publishers.

The pay per click model comes in two forms. The first form is search engine advertising. An example of Google ads is shown in Figure 1.5. To navigate the World Wide Web and find desired sites, users input keywords into search engines like Google or Yahoo!. The engines, which index millions of online websites, then look for pages whose content matches that of the keywords. The best results are sent back to the user. In search engine advertising, firms like Google auction keywords to advertisers, who bid by amount paid per click. The ads of top bidders then appear alongside the search results for the auctioned keyword. By picking and choosing which keywords to couple ads with, advertisers selectively target the types of users to market to.



Results **1** - **10** of about **22,700,000** for **click fraud**. (**0.25** seconds)

Click Fraud Miracle Cure
PPCSecure.com Stop **Click Fraud** in 5 minutes Fast, Easy, 100% Guaranteed

AdWatcher - **Click Fraud**
www.AdWatcher.com Voted Top Rated **Click Fraud** Detection & Prevention Software

Stop **Click Fraud**
www.**click**true.net Detect and block **click fraud** before it happens. Free trial.

Click fraud - Wikipedia, the free encyclopedia
Click fraud is a type of Internet crime that occurs in pay per click online advertising when a person, automated script or computer program imitates a ...
Pay per click advertising - Non-contracting parties - Organization
en.wikipedia.org/wiki/**Click**_**fraud** - Cached - Similar

Click Fraud
Fleischmann is a victim of **click fraud**: a dizzying collection of scams and deceptions that inflate advertising bills for thousands of companies of all sizes ...
www.businessweek.com/.../06.../b4003001.htm - 12 hours ago - Cached - Similar

Exposing **click fraud** - CNET News

Sponsored Links

Sponsored Links

KeywordMax PPC Management
Track all of your traffic
with KeywordMax and Bid Director!
www.KeywordMax.com

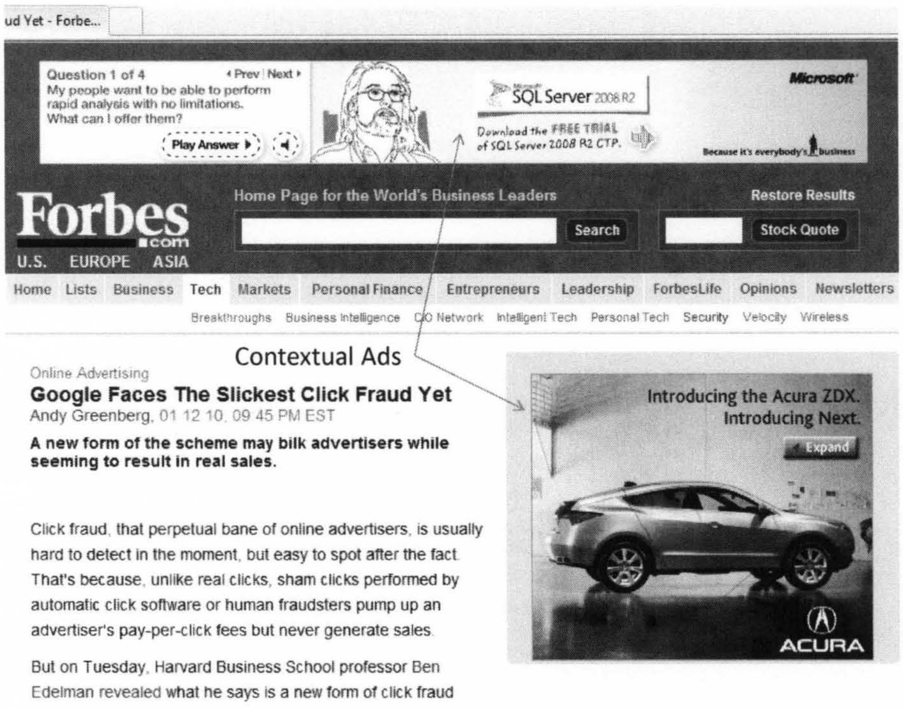
Top PPC Tracking Software
Detect Bogus Clicks. Stop **Click Fraud** & Improve Traffic Quality .
www.**Click**Forensics.com

Click Fraud Information
Find resources that will help you learn more about **click fraud**
www.google.com

Detect PPC **Click Fraud**
Track ad clicks & stop **fraud**
Measure CPC ads with ROI.
click.websitegear.com

Figure 1.5: Search Engine Advertising

PPC's second form is contextual advertising. Unlike search engine advertising where ads are placed alongside website search results, contextual advertising places ads into websites themselves. An example is shown in Figure 1.6. Sometimes, marketing firms contact site publishers directly to host ads for a negotiated fee. More commonly, however, intermediaries called advertising networks will help publishers find ads to host. Advertising networks maintain ad inventories that include various graphics like banners and text links. Ads within inventories are designed to match an advertiser's message and are stored on network servers. Advertising networks then provide publisher with bits of HTML code, the markup language for creating webpages, to script into host sites. When a web user


ud Yet - Forbe...
Question 1 of 4
Prev Next
My people want to be able to perform rapid analysis with no limitations. What can I offer them?
Play Answer
SQL Server 2008 R2
Download the FREE TRIAL of SQL Server 2008 R2 CTP.
Microsoft
Because it's everybody's business
Forbes.com
U.S. EUROPE ASIA
Home Page for the World's Business Leaders
Search
Restore Results
Stock Quote
Home Lists Business Tech Markets Personal Finance Entrepreneurs Leadership ForbesLife Opinions Newsletters
Breakthroughs Business Intelligence CIO Network Intelligent Tech Personal Tech Security Velocity Wireless
Contextual Ads
Introducing the Acura ZDX.
Introducing Next.
Expand
ACURA

Online Advertising

Google Faces The Slickest Click Fraud Yet

Andy Greenberg, 01 12 10, 09 45 PM EST

A new form of the scheme may bilk advertisers while seeming to result in real sales.

Click fraud, that perpetual bane of online advertisers, is usually hard to detect in the moment, but easy to spot after the fact. That's because, unlike real clicks, sham clicks performed by automatic click software or human fraudsters pump up an advertiser's pay-per-click fees but never generate sales.

But on Tuesday, Harvard Business School professor Ben Edelman revealed what he says is a new form of click fraud

Figure 1.6: Contextual Advertising

visits the site, the code displays inventory ads from the network server for the user to view. Revenues from per click ads are then shared between the advertising network and publisher.

In every growing line of business PPC Internet advertising is getting more and more popular mainly due to couple of reasons. First, search engines have eased the setup of this kind of a marketing account. Second, once setup it can generate traffic very quickly, depending on the level of competition. Third, this type of marketing delivers ads to users who are already searching for the product or services that an advertiser is offering, meaning they are receiving mainly qualified traffic.

Increasing popularity of PPC inevitably increased the competition among advertisers for popular keywords, thus increased the bid values for these keywords. As a result search engines continue to make big profits. Figure 1.7 shows the total revenue generated by popular CPM (Casale Media^{TM} , Burst Media ^{TM} , Value Click^{TM} , and Tribal Fusion^{TM}) and PPC((Google^{TM} , Yahoo!^{TM} , bing^{TM})) companies in the recent years. It can be seen that PPC model is growing its popularity compare to the CPM over the years.

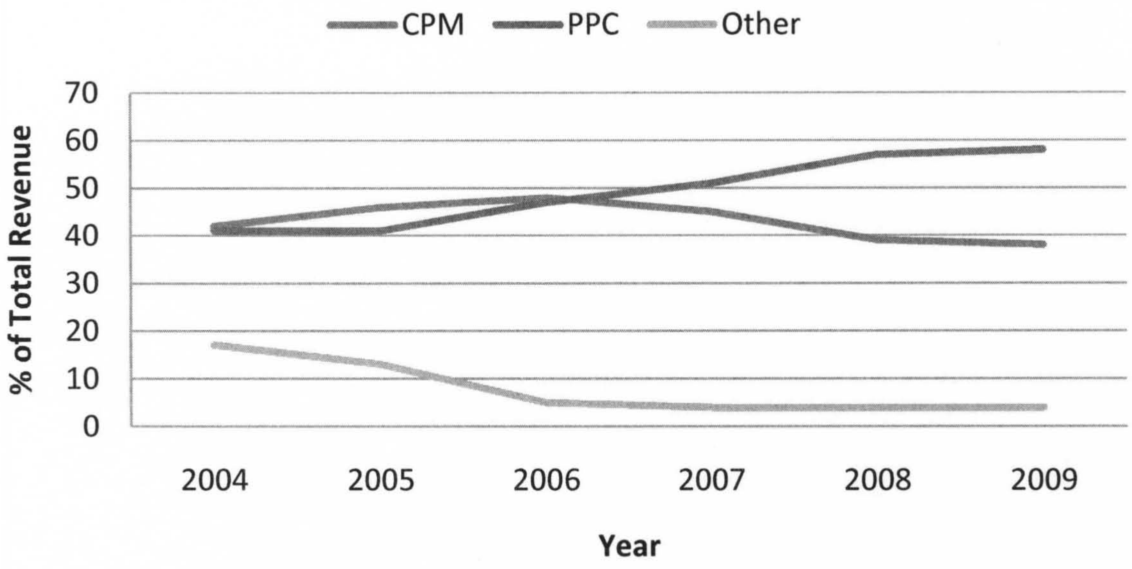


Figure 1.7: Internet Ad Revenue by Pricing Model

However, the enormous growth of PPC advertising helped a group of fraudsters to profit from that growth. This became obvious when companies started to report that their fraudulent traffic was more than 50% of total traffic, and losses were in the range of \$5,000 to \$300,000. These fraudsters were using the inherent drawback of the PPC advertising. PPC financial model totally depends on exchange of HTTP requests from advertiser and publisher. For a syndicator or publisher's server a "click" is simply a browser request for a URL associated with a particular ad. The server has no way to determine if a human initiated the action and, if a human was involved, whether she acted knowingly and with honest intent. Fraudsters soon started to manipulate PPC adverts with artificially generated clicks, a process which is known as *click fraud* [Juels et al., 2007].

The major companies treat pay per click fraud seriously. But the situation is a little similar to the battle against viruses; as a signature for a virus is released, a new virus or strain appears. In order to remain undetected, professional inflators of clicks closely simulate real visitor behavior and visitor parameters. Obviously, click fraud is becoming a significant threat to the rapid growth of on-line marketing sector, and as Google CFO George Reyes stated recently, "it threatens their entire business model where the paid-search has been the primary revenue generator" [Goodman, 2005].

1.4 Goals of the Dissertation

This research proposes new methodologies which can protect Pay-Per-Click advertisers from click fraud and improve their Return on Investment (ROI). The proposed solutions also provide an arbitration system for advertisers and PPC publishers whenever a click fraud argument arises. Advertisers can gain their confidence on PPC advertisement by having a channel to argue the traffic quality with big search engine publishers. General consumer will gain their confidence on internet business model by reducing fraudulent activities which are numerous in current virtual internet world.

In the fast growing Internet advertising industry, the pay-per-click business model, in which advertisers pay for click-through to their website, is the dominant Internet advertising model. Click fraud becomes a serious problem as the pay-per-click (PPC) business model gained its popularity. We developed novel data mining methodologies and tested our proposed solutions with the implementation of Collaborative Click Fraud Detection and Prevention (CCFDP) system to detect and prevent click fraud. This system uses a unique two-step logging structure to authenticate every click. In contrast to most of the existing click fraud solutions, which only use client side logging process, we match up every click from both server and client side.

Our solutions predict click fraud by finding the abnormalities in a click considering its short term and long term behavior. In some of the data mining modes a click is considered as a point in a large feature space, while in some data mining models a group of clicks are analyzed together for identification of behavioral characteristics. At the same time, it has the functionality of blocking suspicious clicks. The results of this system will booster the internet economy by eliminating the shortcoming of PPC business model. General consumer will gain their confidence on internet business model by reducing fraudulent activities which are numerous in current virtual internet world.

We implemented two versions of the CCFDP system, which achieved different goals. The initial version uses a rule based system to calculate overall click fraud score for each click. CCFDP improved version enhances the previous version by providing a traffic quality score calculation method based on fusion of evidence from several data mining models. Each data mining model generates a partial score for a click based on its analysis. These partial scores are then combined to calculate the total quality score.

We accumulated large click data set, approximately 250,000 clicks from an actual ad campaign. We compared different website contents, and different origins, e.g. Natural Traffic (Non-paid traffic) and Pay-Per-Click traffic (Paid traffic), thus, finding the suspicious activities. We also compared the result of proposed solution with that of Google Adwords reports.

Detecting duplicates in click data streams is an important task to fight against click fraud. Therefore, in this dissertation we also considered the problem of detecting duplicates in click data streams. Our solution uses a modified version of the Counting Bloom Filter. The Temporal Stateful Bloom Filter (TSBF) extends the standard Counting Bloom Filter by replacing the bit-vector with an array of counters of states. These counters are dynamic and decay with time. We conducted a comprehensive set of experiments, using synthetic and real world data. Results are compared with Buffering techniques used in the initial CCFDP system.

Nowadays, almost every task involving Web traversing and information retrieval depends on Web robots. Web robots are software programs that automatically traverse the Webs hypertext structure. They proliferate rapidly aside with the growth of the Web and are extremely valuable and important means not only for the large search engines, but also for many specialized services such as investment portals, competitive intelligence tools, etc. While many web robots serve useful purposes, recently, there have been cases linked to click fraud committed by these Web robots. In this research we detail the architecture and functionality of the Smart ClickBot, a sophisticated software bot that is designed to commit click fraud. It was first detected and reported by CCFDP improved solution in March, 2010. We discuss the machine learning algorithms used to identify all clicks exhibiting Smart ClickBot like patterns. We disclose the results of our investigation of this bot to educate the security research community and provide information regarding the novelties of the attack.

1.5 Organization of the Dissertation

This dissertation is organized as followings. Chapter 2 introduces the concept of click fraud in online advertising. Chapter 3 compares and contrasts the current commercial solutions for click fraud detection in advertising networks. Chapter 4 reviews the research in the domain of click fraud detection, while Chapter 5 introduces the architecture and operation processes of the Collaborative Click Fraud Detection and Prevention (CCFDP) System. The improved version of the CCFDP system is introduced in Chapter 6. Methodologies for Smart ClickBot detection and use of Temporal Stateful Bloom filters for fast detection of duplicates are introduced in Chapter 7. Conclusions and future work are given in Chapter 8 and Chapter 9 respectively.

CHAPTER 2

CLICK FRAUD IN ONLINE ADVERTISING

2.1 Introduction

Click fraud, is a by-product of the "pay per click" online advertising model. It is the act of clicking ads with fraudulent or malicious intent to generate illegitimate revenue or hurt competitors. Because fraudulent clicks do not represent genuine consumer interest, they hold zero economic value [Kintana et al., 2009]. Advertisers are charged on a per click basis, so an excessive number of bad clicks can severely inflate an organization's advertising expenditure. Since online advertising networks act as brokers of multi-billion dollar online advertising revenue streams, click fraud has become a major concern for publishers too.

Recently click fraud has become a subject of some controversy and increasing litigation due to the advertising networks being a key beneficiary of the fraud. On one hand, the brokers do not wish to lose customers. On the other hand, they are not able to provide full details of the clicks, their rates, and origin IP addresses to advertisers. Even worse, the advertisers would wish to claim that all clicks are fraud, hence avoiding click charges. For instance, broker loses money to undetected click fraud when it pays out to the publisher (third party website), but it makes more money when it collects fees from the advertiser [Haddadi, 2010]. Because of the spread between what the broker collects and what it pays out, click fraud may directly profit the broker. This also provides an incentive for the publishers to hire click fraud botnets or human teams to increase their revenue.

Estimates of click fraud prevalence range from 5-50% of all internet clicks, with values from 15-35% quoted most often. Even low end estimates pose huge concerns for the rapidly expanding online advertising industry.

Click fraud benefits a fraudster in at least three ways. First, a firm may click numerous times on competitor ads to drain at profit margins and thus gain a market edge. Competitor click fraud is not committed for any monetary gain, but rather in order to harm one's competitor. A prerequisite for the fraud is that the competitor in question has signed up as advertiser in a PPC scheme. The fraudster, knowing that every click on the ads costs the competitor good money, clicks repeatedly on his ad to cause harm. He might perform this task himself or use more sophisticated means, such as hiring a group of people or using a software.

Second, ad space suppliers like search engines, advertising networks, and publishers may make bad clicks to generate more revenues for themselves at the advertiser's expense. Publishers can in principle derive financial benefit from click fraud in the short term, as they receive revenue for whatever clicks they deem "valid". In the long term, however, as customers become sensitive to losses, and syndicators rely on third party auditors to lend credibility to their operations, click fraud can jeopardize syndicator advertiser relationships. Thus syndicators ultimately have a strong incentive to eliminated fraudulent clicks. Today they employ a series of filters to week out suspicious clicks. These filters are trade secrets of individual companies, as their disclosure might prompt new forms of fraud. For example it is likely that syndicators use IP tracing to determine if an implausible number of clicks is originating from a single source. While heuristic filters are fairly effective, they are of limited utility against sophisticated fraudsters, and subject to degraded performance as fraudsters learn to defeat them [Juels et al., 2007].

Third, a fraudster can modify the ranking of advertisements by a combination of impressions and clicks. An impression is the viewing of the banner. Impression fraud occurs when fraudsters manipulate the number of page impressions for a given search term. When an advertiser's relative click-through rate (CTR[1]) decreases, his or her search term

[1]CTR is a way of measuring the success of an online advertising campaign. A CTR is obtained by dividing the "number of users who clicked on an ad" on a web page by the "number of times the ad was delivered" (impressions).

can be suspended because of low CTR performance. This creates a window of opportunity for other advertisers. By committing impression fraud, they are able to obtain higher search rankings at lower costs due to the crippled competition.

Fraudsters take the advantage of the lack of verifiable human engagement in PPC requests, in order to fabricate ad traffic. It can take a number of forms. One type of click fraud relies on real clicks, whether intentional or not. An example of the former is a so called click farm, which is a term denoting a group of low wage workers who click for a living. Another example involves deceiving or convincing users to click on advertisements. An example of an unintentional click is one generated by a malicious cursor following script that places the banner right under the mouse cursor. This can be done in a very small window to avoid detection. When the user clicks, the click would be interpreted as a click on the banner, and cause revenue generation to the attacker. A related abuse is manifested in an attack where publishers manipulate web pages such that honest visitors inadvertently trigger clicks. This can be done for many common PPC schemes, and simply relies on the inclusion of a javaScript component on the publisher's webpage, where the script reads the banner and performs a get request that corresponds to what would be performed if a user had initiated a click.

2.2 Types of Fraud

Based on the type of attack we can identify three broad categories in the realm of clicks fraud. They are active human, active software and passive software click fraud. In the following section we briefly discuss each of these types.

2.2.1 Active Human Click Fraud:

In this category clicks are generated by human activity and the entire web request process is completed, which means that not only the web page is loaded, but also the images, flash, javascipt code etc. are loaded. Human may have activity on the web page such as mouse click, and scroll bar activities and the page view time is at least couple of seconds. This type of fraudulent clicks are generated mostly by people in developing countries who are hired for a nominal wage.

2.2.2 Active Software Click Fraud:

In this kind, a software initiate the click fraud. In most of the cases, the web page request is not complete and only the initial text web page is requested without the images, javascript code or videos. No mouse or keyboard activities are followed and the page view time is less than one second. A web page request to a server usually contains multiple follow up requests. It first loads the texts in the web page followed by images, multimedia, etc. In click agent software these requests are less likely to happen and such clues can be utilized to identify software clicks. Although some smart click agent software can generate these follow up requests, they are still different from the browser requests generated by real users, such as mouse click, mouse movement, page view time etc.

2.2.3 Passive Software Click Fraud:

These frauds correspond to Adware and spyware run on the background of the client computer without being known by the user. It hijacks browser sessions and sends out web requests to multiple ad servers. These spyware or adware are installed on client computers without the consent of users. Such software might pops up an advertise window, sometimes pops under, or might not pops up windows at all. The click fraud of this type also committed by software of type 2. The difference is, in type 2, fraud is a user initiated click and it is active. But in type 3 click fraud is originated from different client computers and it is passive, which means the client user is not aware of any click fraud activities. To differentiate if the clicks are of type 2 or 3 the pay per click provider should check if the ad was actually placed on the referring page or the originating search. Because in type 3 it is not necessary to have an advertisement displayed in the browser. Most of the background click robots are programmed to send requests without being clicked on an advertisement. So the pay per click provider should check if a real browser clicked the advertisement or not. This could be done by executing a java script code, or doing redirects that do not occur every time. Another possibility to detect if the click is of type 2 or 3 is having log records. Advertisers can always check, are images being downloaded in the requested webpage? Are any other pages clicked from the initial page? etc. The recent addition to type 3 is *click bots* or *botnet* attacks.

The most novel characteristic of the clickbot is that it is built to conduct a low noise click fraud attack against syndicated search engines. In our research we are focusing on detection and prevention of type 2 and type 3 click fraud with the emphasis on detecting click bot attacks.

Besides committing click fraud bots are widely used in few other types fraudulent activities:

1. Denial-of-service (DOS): Bots autonomously access an Internet system or service in a way that appear legitimate, but much more frequently than normal use and cause the system to become busy.

2. Spread Spyware: Spyware are used to send information to its creators about a user's activities.

3. Harvest E-mail: E-mail spam are e-mail messages disguised as messages from people, but are either annoying ads or malicious in nature.

4. Access number replacement: Access number replacements are where the botnet operator replaces the access numbers of a group of dial-up bots to that of a victim's phone number. Given enough bots partake in this attack, the victim is consistently bombarded with phone calls attempting to connect to the internet. Having very little to defend against this attack, most are forced into changing their phone numbers (land line, cell phone, etc).

5. Fast flux: Fast flux is a Domanin Name System (DNS) technique used by botnets

to hide phishing[2] and malware delivery sites behind an ever-changing network of compromised hosts acting as proxies.

Immaterial of the fraudulent activity (DOS, click fraud etc.) bots are built on common infrastructures. Following discussion will only focus on bots used for committing click fraud. Based on how they are built and which communication protocols they use, we can classify click bots into three main categories.

- Internet Relay Chat (IRC) click bots
- Peer-to-Peer (P2P) click bots
- HTTP click bots

2.3 IRC click bots

The initial deployment of IRC bots, dated back in 1993. These computer bots were used to assist the Internet Relay Chat channel management. Internet Relay Chat is a chat system that provides one-to-one and one-to-many instant messaging over the Internet. It is based on Client/Server architecture. Administrator can create a channel[3] and manage its clients. Users can join a named channel on an IRC network and communicate with groups of other users.

Internet community have slowly started experimenting these computer bots to use to commit click fraud.

An IRC based click bot is controlled by a bot-master and there are many bots attached to one bot-master. A bot-master can operate bots via command and control (C&C). Command and control activity is defined as a platform for transmitting the commands of a bot-master to all bots. A bot-master takes advantage of Internet Relay Chat because

[2]Phishing is the fraudulent process of attempting to acquire sensitive information such as usernames, passwords and credit card details by masquerading as a trustworthy entity in an electronic communication.

[3]The basic means of communication in an established IRC session is a channel. Channels in a server can be displayed using the IRC command LIST that lists all currently available channels.

this makes it easy to operate Command and Control. A typical bot-master and bot relationship is shown in Figure 2.1.

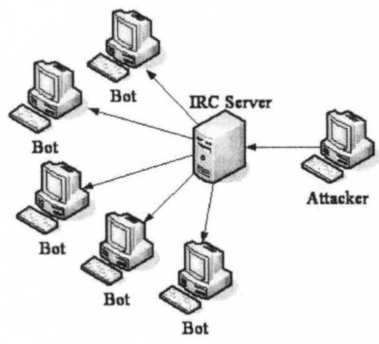


Figure 2.1: IRC Botmaster controlling several Clients

These are the most easy to build and most widely used click bots used by the fraudsters. A typical formation of botnet can be described by the following steps, as shown in Figure 2.2.

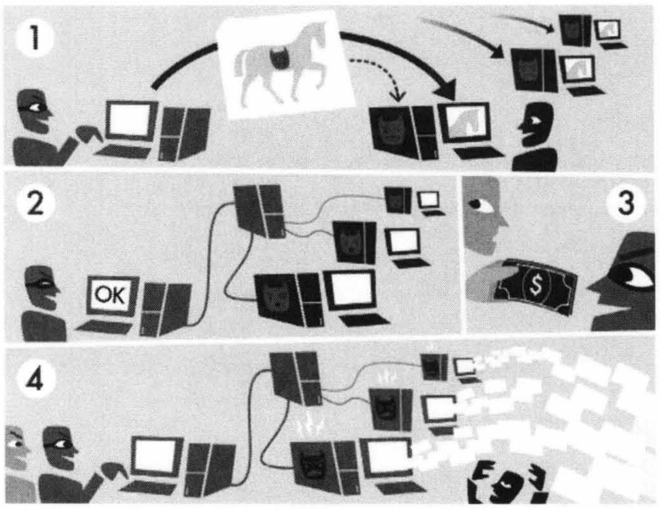


Figure 2.2: Formation and Exploitation of Botnet

(i) A botnet operator (Administrator of the channel) sends out viruses or worms, infecting ordinary users' computers, whose payload is a malicious application of the bot (its clients).

(ii) The bot on the infected PC, logs into a particular channel in the Command and Control server specified during installation.

(iii) A fraudster purchases access to the botnet from the operator.

(iv) The fraudster sends instructions via the IRC server to the infected PCs, causing them to click on specific advertisements.

IRC bots are susceptible to center point failure. If a malware detection program finds the Bot master and removes it the whole network will be shut down.

2.4 P2P click bots

A peer-to-peer network is a network in which any node in the network can act as both a client and a server (see Figure 2.3). P2P bots are developed to minimize the hazard of center point failure in IRC bots. Peer-to-peer botnets are distinctive from centralized C&C botnets in that they focus on resiliency through the uses of a peer-to-peer network. However, peer-to-peer botnets are similar to centralized botnets in most other aspects.

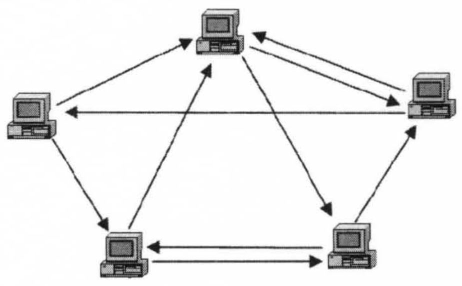


Figure 2.3: Peer-to-Peer Botnet

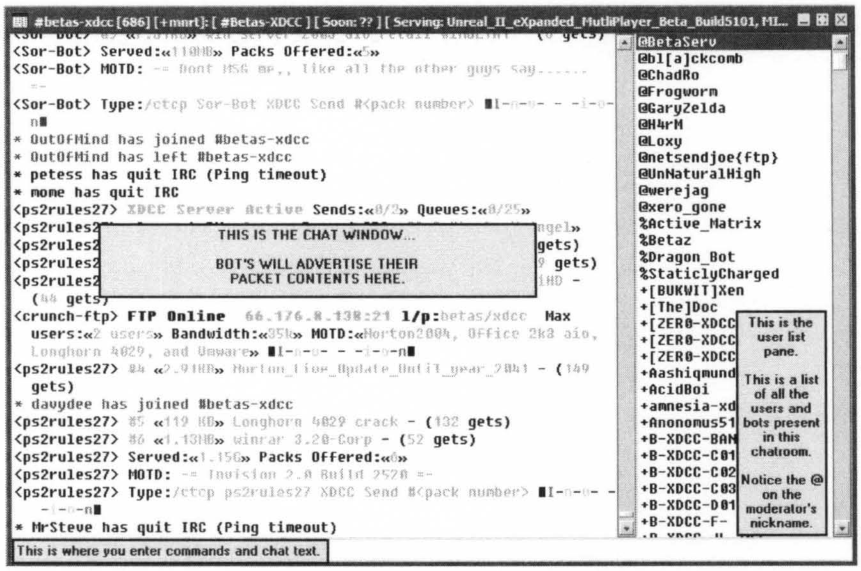
Both IRC Bots and P2P Bots use IRC as the channel of communication. Figure 2.4 shows how a Bot master interacts with its clients in an actual chat window. Steps involve in the process is as follows.

1. Botmaster connects to an IRC Channel with a nick name. In Figure 2.4 the nickname @Frogworm

2. All clients connect to the same channel through different nick names (known to the botnet). In Figure 2.4 nicknames B-XDCC-C02, B-XDCC-C01, B-XDCC-C03.

3. Communicate using natural language words

4. Clients carry out commands



```
#betas-xdcc [686] [+mnrt]: [ #Betas-XDCC ] [ Soon: ?? ] [ Serving: Unreal_II_eXpanded_MutliPlayer_Beta_Build5101, MI...
<Sor-Bot> Served:«110MB» Packs Offered:«5»
<Sor-Bot> MOTD: -= Dont MSG me,, like all the other guys say......
  =-
<Sor-Bot> Type:/ctcp Sor-Bot XDCC Send #<pack number> ▮I-n-u- - -i-o-
  n▮
* OutOfMind has joined #betas-xdcc
* OutOfMind has left #betas-xdcc
* petess has quit IRC (Ping timeout)
* mome has quit IRC
<ps2rules27> XDCC Server Active Sends:«0/2» Queues:«0/25»
<ps2rules2                                                ngel»
<ps2rules2                                                gets)
<ps2rules2                                                9 gets)
<ps2rules2                                                1MD -
  (44 gets)
<crunch-ftp> FTP Online  66.176.8.138:21 1/p:betas/xdcc  Max
  users:«2 users» Bandwidth:«35k» MOTD:«Norton2004, Office 2k3 aio,
  Longhorn 4029, and Vmware» ▮I-n-u- - -i-o-n▮
<ps2rules27> #4 «2.91MB» Norton_Live_Update_Until_year_2041 - (149
  gets)
* davydee has joined #betas-xdcc
<ps2rules27> #5 «119 KB» Longhorn 4029 crack - (132 gets)
<ps2rules27> #6 «1.13MB» winrar 3.20-Corp - (52 gets)
<ps2rules27> Served:«1.15G» Packs Offered:«6»
<ps2rules27> MOTD: -= Invision 2.0 Build 2520 =-
<ps2rules27> Type:/ctcp ps2rules27 XDCC Send #<pack number> ▮I-n-u- -
  -i-o-n▮
* MrSteve has quit IRC (Ping timeout)
```

```
@BetaServ
@bl[a]ckcomb
@ChadRo
@Frogworm
@GaryZelda
@H4rM
@Loxy
@netsendjoe{ftp}
@UnNaturalHigh
@werejag
@xero_gone
%Active_Matrix
%Betaz
%Dragon_Bot
%StaticlyCharged
+[BUKWIT]Xen
+[The]Doc
+[ZER0-XDCC
+[ZER0-XDCC
+[ZER0-XDCC
+Aashiqmund
+AcidBoi
+amnesia-xd
+Anonomus51
+B-XDCC-BAN
+B-XDCC-C01
+B-XDCC-C02
+B-XDCC-C03
+B-XDCC-D01
+B-XDCC-F-
```

THIS IS THE CHAT WINDOW...
BOT'S WILL ADVERTISE THEIR PACKET CONTENTS HERE.
This is the user list pane.
This is a list of all the users and bots present in this chatroom.
Notice the @ on the moderator's nickname.
This is where you enter commands and chat text.

Figure 2.4: Bots communication in IRC channel

2.4.1 HTTP click bots

HTTP click bots are the newest type of botnet. HTTP botnets use HTTP requests to receive and import commands instead of a persistent IRC or P2P connection. It also follows the common Server/Client structure, where the server(bot master) broadcasts the commands and clients(bots) carry out the commands. Formation of the botnet is similar to IRC and P2P where, a botnet operator sends out viruses or worms, infecting ordinary users' computers. It is still in its evolving process and soon we will have many varieties of it.

The code base for the HTTP Bot uses an HTTP post that contains information describing the bot. It simply does a GET request and receives back similar series of commands from its bot master. It is a single request that (a) identifies itself (b) receives the command from bot master and (c) reply.It then sort of goes quiet or launches an attack. The point here is that it is not a persistent connection that is necessary for P2P and IRC-based botnets. That means the HTTP botnet goes stealth, making it harder to find.

2.5 clickbot.A

In this section we discuss the implementation and functionality of *clickbot.A*.

The Clickbot.A client, or bot, is an Internet Explorer (IE) Browser Helper Object (BHO[4]). Similar to other browser helper objects, it runs within the process space of the browser, and is capable of accessing the entire DOM (document object model) of a web page. The Document Object Model is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. Clickbot.A was most probably written as a BHO so that its HTTP requests would mimic those generated by legitimate IE clients, and the work of accessing and parsing web pages would automatically be handled for the bot master. HTTP seems like a reasonable choice for fraudsters to build botnets for click fraud. The bots need to communicate using HTTP to click on ads, so why support an additional protocol if HTTP can be reused for command and control anyhow? If one was to attempt to use an IRC-based botnet to conduct click fraud, for example, the bot binary may have to support two protocols (HTTP and IRC). The binary for the bot may be smaller if it only has to support one communication protocol, and HTTP has the additional advantage that firewalls typically let HTTP traffic pass through freely, whereas some firewalls might restrict IRC traffic [Daswani and Stoppelman, 2007].

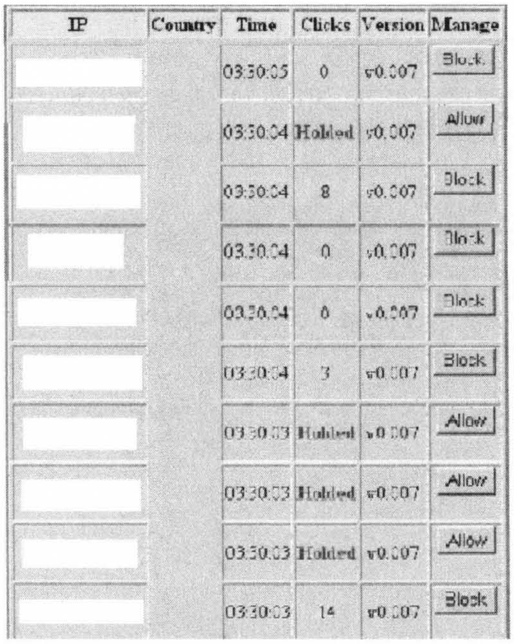
Once the bot is successfully installed it goes through the following three step process.

1. Contact its botmaster to register in botmaster server. The Clickbot.A botmaster

[4]A Browser Helper Object (BHO) is a DLL module designed as a plugin for Microsoft's Internet Explorer web browser to provide added functionality. The Adobe Acrobat plugin that allows Internet Explorer users to read PDF files within their browser is a BHO. Alexa Toolbar that provides a list of web sites related to the one you are currently browsing, or the Google Toolbar that adds a toolbar with a Google search box to the browser user interface are examples of some more BHOs.

was implemented as an HTTP based web application written using PHP, and used a MySQL database.

The bot registers with the botmaster using an URL such as
http://example.php? action=register& ver=0.007& id=GUID.
Note that the *action* parameter specifies the operation that the client requests (in this case to "register" with the botmaster). In addition to the action parameter, the client always reports its version via the *ver* parameter (ver=0.007), and a globally unique identifier via the *GUID* parameter to the botmaster server. So the botmaster can uniquely identifies each bot. Once a bot is successfully registered with the botmaster a raw will appear in botmaster's MySQL database as in Figure 2.5.



IP	Country	Time	Clicks	Version	Manage
		03:30:05	0	v0.007	Block
		03:30:04	Holded	v0.007	Allow
		03:30:04	8	v0.007	Block
		03:30:04	0	v0.007	Block
		03:30:04	0	v0.007	Block
		03:30:04	3	v0.007	Block
		03:30:03	Holded	v0.007	Allow
		03:30:03	Holded	v0.007	Allow
		03:30:03	Holded	v0.007	Allow
		03:30:03	14	v0.007	Block

Figure 2.5: Botmaster Administration Console. Note that entries in the IP address and country columns have been sanitized for privacy purposes.

2. Learn about doorway site: A doorway site is a web site set up by the bot operator(who is a human), to function similar to a search engine. Doorway pages are Web pages designed and built specifically to draw search engine visitors to a website. They are standalone pages designed only to act as an entry point to a site. In

Figure 2.6 doorway sites 1,2, and 3 designed only to redirect traffic to service 1, service 2, and service 3 pages.

In clickbot.A, each doorway page is especially tuned to a particular keyword search. When an internet user does a search, the doorway pages will show up and lead them directly to the main site (or target website). This is a proven and effective way to increase traffic to the website. Doorway pages are typically very "light" pages, in that they contain little or nothing in the way of pretty formatting, colors, tables, Javascript, images, etc. Instead, the HTML code is optimized to be search engine friendly.

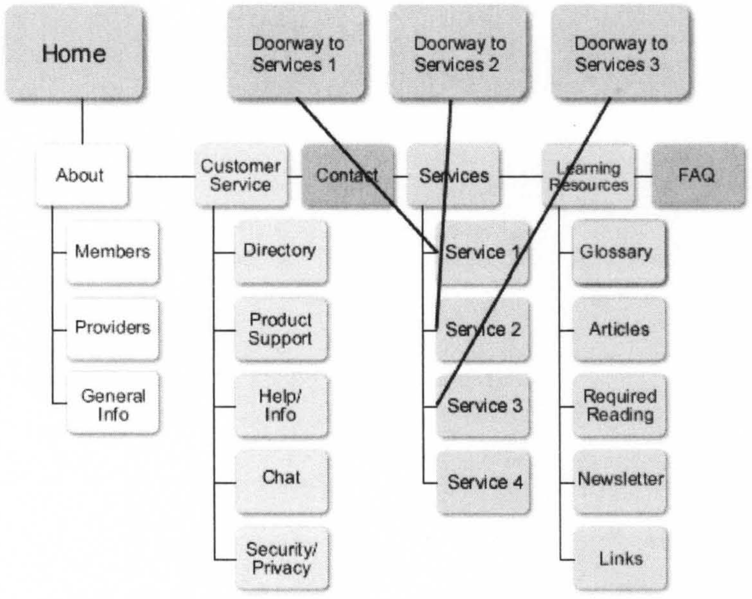


Figure 2.6: Example of doorway sites.

The bot runs an infinite loop in which it requests a doorway site and keyword, accesses the doorway site, and chooses a candidate link to click on from the doorway site. The bot repeats this loop for specified number of iterations which was set by the botmaster.

3. Place the HTTP request: The bot places a HTTP query to a doorway site. It will then receive search results containing clickable links. After communicating with the botmaster the bot issues a click. The doorway site in one version acts as a legitimate

search engine when applying to become a partner with a syndicator and serves as a location from which a bot can access URLs to click on and commit direct click fraud. In the other version the doorway site appears as a site that refers traffic to other websites and make a referrer deal. In the later version clickbot.A uses redirectors[5] Had the doorway sites not used redirectors, the web sites from which the bot operator would derive revenue could notice how many clicks were originating from the doorway sites, and could more easily investigate those sites based on information in their web logs. While one might expect that HTTP requests that do not specify referrers might be considered suspicious, there are many legitimate reasons that a referrer might not appear in an HTTP request. For instance, HTTP proxies that are used to conduct anonymous web browsing typically do not include referrers.

Click fraud is not difficult to commit and, consequently, it could threaten the existence of companies like Google^{TM}. Therefore, researchers in both academia and industry have proposed a series of models to detect it.

[5]Redirector is a customizable program, which returns a new URL replacing Client's original request. Redirectors allowed the attacker to strip the Referer fields in HTTP requests issued by Internet Explorer.

CHAPTER 3

OVERVIEW OF COMMERCIAL CLICK FRAUD DETECTION SYSTEMS

The disagreement about the click fraud traffic between publishers and advertisers has grown and resulted in class action law suits against big search companies [Goldman, 2007]. Recently most of the search companies have started to take click fraud seriously and came up with solutions that kept as secrets until today. But law suits against search companies have not stopped. In most of search engine solutions advertisers are placed in a disadvantage position to those big publishers. Advertisers can neither control the quantity and quality of traffic, nor negotiate the payment to big publishers if they suspect the authenticity of the traffic. Due to this reason third party click fraud solutions also started to bloom. Most of the advertisers who lost their trust on the search engine companies turn to these third party clients who manage their pay per click advertisement campaigns. Many of these third party companies developed click fraud detection services using their own matrics and offer them on the market. Non of them reveal the techniques use to detect fraudulent traffic and, they too, keep solutions as trade secrets. This created one additional problems. Which solution is correct? Is it the one that estimates a lot of click fraud? How should results from two different companies compared?

Many third party click fraud auditing firms often significantly overestimate the number of detected click fraud activities, reporting so called "fictitious" clicks, clicks which were never made on search engine ads. That causes a series of on going battles between search engines such as Google and small third party click fraud auditing firms.

So until today there is no universally accepted click fraud detection method. Most of the commercial solutions available are still not mature and do not discover most of the click fraud activities, and mainly they are reporting tools without mechanisms in preventing click fraud functionality.

Several commercial solutions, e.g. Clicklab, LLC, Web Traffic Intelligence, Inc. etc., are available for click fraud detection. They all use similar technology by adding a sampler or collecting javascript or iframe code on a page to track all activities. The code will run on the client computer when pages are viewed. Whenever the javascript or iframe is executed on client browser, it sends back information to the logging server. The most common client side parameters include client IP, client user agent, client browser settings, client computer settings, link-out click, user activity etc. Figure 3.1 shows the typical process of commercial click fraud solutions. These solutions include the following steps:

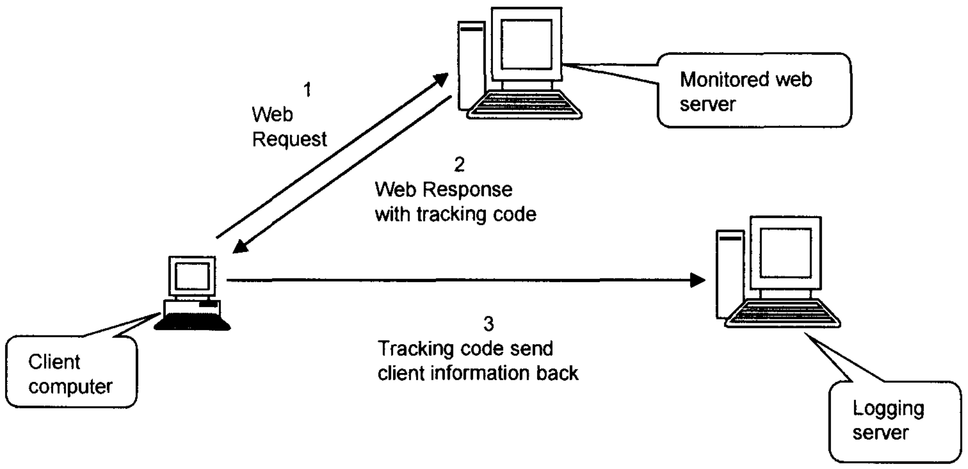

Monitored web server
1 Web Request
2 Web Response with tracking code
Client computer
3 Tracking code send client information back
Logging server

Figure 3.1: The tracking code is an essential component of all commercial click fraud detection systems

1. A tracking javascript or iframe code was added on each tracked page's bottom.
2. A client computer requests a web page. (Figure 3.1 message 1)
3. The web server response the web page with tracking code. (Figure 3.1 message 2)
4. The tracking code executes on client computer and sends the detailed tracking information back to log server. (Figure 3.1 message 3)

Those companies provide web reports as well as paper reports to their subscribers. The reports normally include the statistics on traffic origin IP, traffic user agents, page view time, heuristic fraud score, etc. All these information are used for detection of click fraud activities, and they can be classified in to two levels:

First Level: Static client parameters such as IP origin, user agent, monitor display setting, web browser setting, java and javascript enabled, web page title etc.

Second Level: Dynamic client parameters such as mouse clicks, mouse location, key stroll, scroll bar clicks, page view times, even client side clip board message etc.

The difference between these two classes of parameters may be recognized in the message 3 given in Figure 3.1. If the tracking code just sends back once to the logging server, this is static client parameters collecting. While the tracking code keeps sending back user's activities, the logging server will have dynamic client parameters.

Figure 3.2 lists the top ten most popular click fraud commercial solutions as of March, 2010.

A significant problem of most of these existing commercial solutions, is that they do not have a way to prevent click fraud dynamically. The javascript or iframe code can not block the page being loaded. Then those solutions are just click fraud reporting system not real click fraud detection and prevention system. There are some other problem exist in the commercial solutions. For example, the www.whosclickingwho.com, and Click Defense Corporate do not have real time user activity information, such as mouse movement, key stroke etc., which is very important to detect post user activities after a click occurs.

There is an ongoing industry-wide effort to develop tools that will effectively detect and block many common click fraud attacks. Most of attacks discovered and reported so far have been malware-based attacks that rely on automated scripts, individuals hired by competitors or proxy servers used to generate clicks for paid advertisements.

These commercial solutions are third-party add-on click fraud solutions. The search engine companies, such as Google, Yahoo, implements their own click quality control system. For example: Goolge's Click Quality team tries to control the click quality in two ways:

Prevention: Discouraging invalid clicking activities on its Network by making life for unethical users more difficult and less rewarding

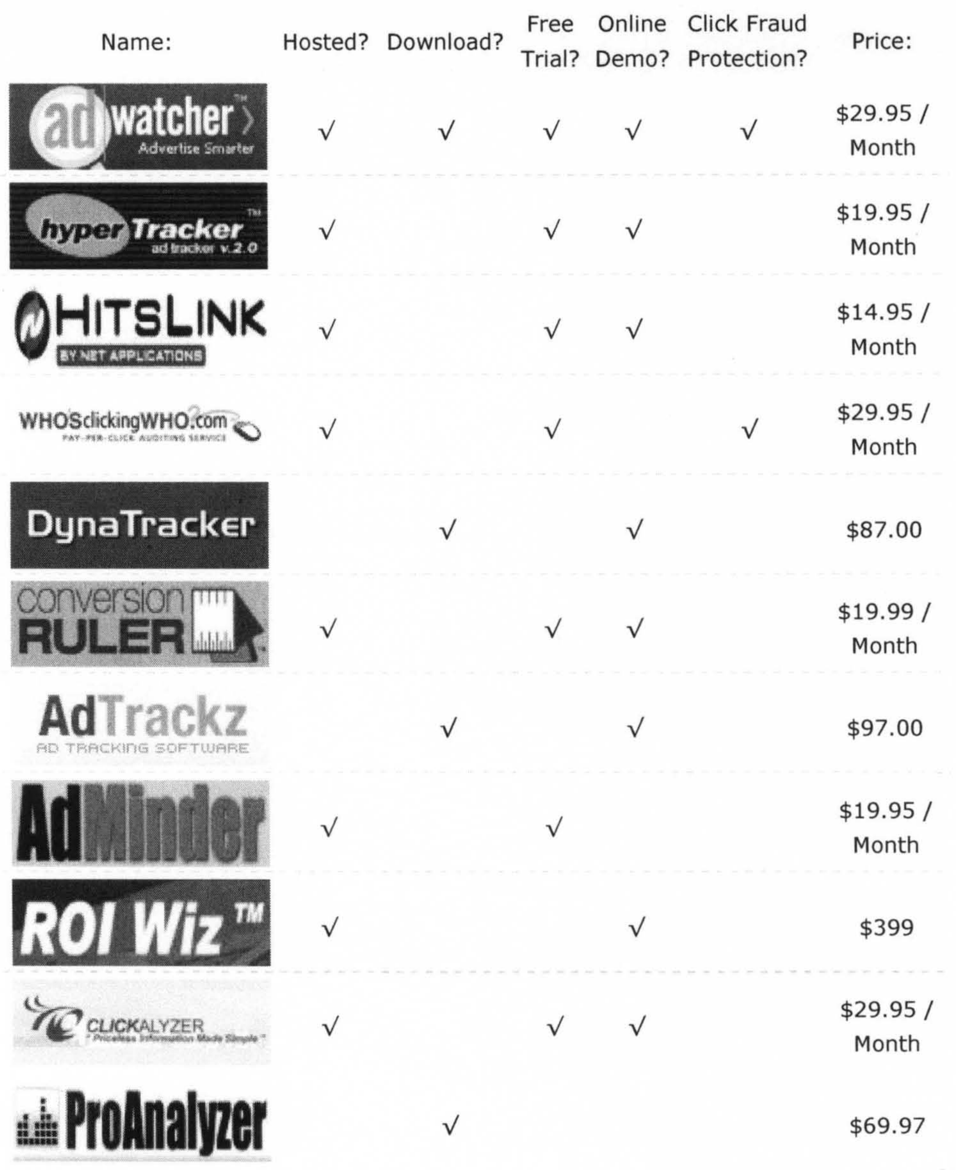
Detection: Detecting and removing invalid clicks and the perpetrators.

In addition to launching an extensive effort to detect and remove invalid clicks, Google also tries to build other mechanisms for preventing invalid clicking that reduce inappropriate activities on the Google Network even before invalid clicks are made.

Some of these preventive activities include:

(a) Making hard to create duplicate accounts and open new accounts after the old ones are terminated, (b) Making hard to register using false identities, and (c) Development of certain mechanisms that automatically discount fraudulent activities, i.e., advertisers pay less for invalid clicks since certain invalid clicking patterns would automatically reduce costs that advertisers pay for these clicks etc.

However. due to the nature inherent weakness of Google's (or any other search engine), which does not have enough data on post-click user activities, it is hard or even impossible to determine the true intent of a click. There are some other weaknesses in these online procedure: (a) lack of deployment of data mining methods, (b) lack of online real time solutions, (c) lack of using the conversion data and lack of more advanced types of filters, (d) use only one aspect of the click information which is the server side click information.



Name:	Hosted?	Download?	Free Trial?	Online Demo?	Click Fraud Protection?	Price:
adwatcher™ Advertise Smarter	√	√	√	√	√	$29.95 / Month
hyperTracker™ ad tracker v.2.0	√		√	√		$19.95 / Month
HitsLink BY NET APPLICATIONS	√		√	√		$14.95 / Month
WHOSclickingWHO?.com PAY-PER-CLICK AUDITING SERVICE	√		√		√	$29.95 / Month
DynaTracker		√		√		$87.00
conversion RULER	√		√	√		$19.99 / Month
AdTrackz AD TRACKING SOFTWARE		√		√		$97.00
AdMinder	√		√			$19.95 / Month
ROI Wiz™	√			√		$399
CLICKALYZER "Priceless Information Made Simple"	√		√	√		$29.95 / Month
ProAnalyzer		√				$69.97

Figure 3.2: Top 10 ad tracking & PPC tracking tools - best ad tracker (2009)

CHAPTER 4

REVIEW OF RESEARCH IN DOMAIN OF CLICK FRAUD DETECTION

Research activities in click fraud detection can be divided into two groups. One group proposes alternative business models for Pay-Per-Click model, while the other group tries to find solutions for click fraud in the Pay-Per-Click payment model.

4.1 Alternatives for Pay-Per-Click model

4.1.1 Cost-Per-Action (CPA)

In the Cost-Per-Action model, advertisers do not pay for clicks, but rather for specific actions that are performed on the advertisers page after the click. These actions include, for example, Generating email opt-ins, Producing product sales, and Signing up subscriptions etc. Therefore, we can define *Completed Actions* as:

$$\begin{aligned} Completed\ Actions = Total\ Number\ of\ Generating\ email\ opt-ins\ + \\ Producing\ product\ sales + Signing\ up\ subscriptions \end{aligned} \tag{4.1}$$

These actions produce measurable outcomes for a company. Ideally, the actions, which are being measured are those most closely tied to the growth of the business; therefore product or service sales are the most common actions tracked. These outcomes are usually known as conversions.

Website conversion is a process of turning website visitors into prospects and customers. The conversions or conversion rate evaluates: (1) the quality of the visitors attracted by the advertising strategies and (2) how satisfied the visitors are interacting with the website.

conversion rate is calculated as follows:

$$Conversion\ Rate = \frac{Completed\ Actions}{Total\ Number\ of\ Visitors} \tag{4.2}$$

It is important to use the same time range when gathering the completed actions and the total number of website visitors.

For example, if in the last month 1,000 visitors visited the website and 10 have purchased a product, the "sales" conversion rate is 1.0%. i.e. for every 100 visitors to the website, 1% is satisfied.

CPA or "cost per action" is the advertising cost somebody pays for one completed action. As with the conversion rate, an action may be generating an email opt-in, producing a product sale or downloading a white paper. For example, if in the last month a company spent $1,000 on advertising to generate 2,000 visitors and 20 of them subscribed to a newsletter, the cost per action for a newsletter subscription is, $1,000 ad cost per 20 subscriptions, i.e. $50.00 Cost per Action.

$$Cost\ per\ Action = \frac{Advertising\ Cost}{Total\ Completed\ Actions} \tag{4.3}$$

Although CPA does prevent publisher and competitor click fraud, it leaves room for advertiser fraud. Since the publisher (and the commissioner, if he exists) have no way to confirm whether or not a specific action has taken place they depend on the advertiser to report the customers action truthfully. Additionally, the publisher has to rely on the advertisers ability to produce both efficient advertisements and if the action in question is making a purchase worthwhile products on the target side in order to make a profit. This means that the publisher does not profit directly from the advertising space and traffic he/she provides anymore. Furthermore, a user might click on an ad on the publishers website without making a purchase on the target site, only to return to the target site and make the purchase later thus robbing the publisher of his well-deserved commission. For the advertiser, on the other hand, the CPA model is very advantageous: since he only has to pay when he actually does make a sale, there is virtually no risk on his side.

However, it is still a model that is based on trust. The only difference being that, this time, it is the other party that has to trust, and as such it is a model that publishers might be wary of adopting.

4.1.2 Pay-Per-Impression

An alternative to the PPC model of advertising is the old Pay-Per-Impression(PPI) model. It is also know as Cost-Per-Mille(CPM). It remains popular on major Internet portals, such as Yahoo.com, msn.com, and aol.com [Edelman et al., 2007]. Cost per Mille is slightly different than PPC. Much like traditional advertising, the display of an ad (impression) is nothing more than a single appearance of an ad. Viewers are not required to take action, for example click the ad, in order for payment to be due. Banners or other advertisements are placed, according to the terms of a contract. Payment is typically based on a predetermined number of impressions, generally set at 1000, thus naming it Cost per Mille(thousand). While advertisers are paying for each instance of the advertisement, it is generally less expensive than Cost per Click.

The dollar figure of "Cost per Thousand" is used to evaluate the cost to reach a thousand persons in a media buy. PPIs are calculated by multiplying the cost of an ad by 1,000, then dividing that number by the total audience.

$$CPM = \frac{Cost\ *\ 1,000}{Total\ Audience} \tag{4.4}$$

Advertisers are drawn towards CPM because of the enormous amount of visibility it can bring; the ad is seen often even when the ad itself is not clicked. It is strongly believed that the ad is still read and that exposure counts over the long run. If the ad is attractive and grabs attention, the Click Through Rate (CTR) will be higher, thus generating even better results.

Unfortunately, Pay-Per-Impression is not fraud resistant either. The technical methods that make click fraud possible can be easily adapted to so-called impression fraud. Instead of simulating a click, though, the script repeatedly requests the website on which the ad is displayed and consequently artificially increases the number of impressions.

4.1.3 Pay-Per-Percentage of Impressions

Pay-Per-Percentage of Impressions is an alternative to Pay-Per-Click that was suggested by Goodman [Goodman, 2005]. In his paper he describes this model as follows. In this system, an advertiser picks a keyword, e.g. "cameras" and purchases, perhaps through bidding, a certain percentage of all impressions for that keyword. For instance, an advertiser might pay $1.00 to Bing Search. In return, the advertiser might receive 10% of all impressions for "camera" for 1 week. What does this mean? It means that for 1 week, one out of ten times that someone searches for the word "camera", they will see the ad. The costs of advertising are thus fixed and do not depend on whether or not the ad is clicked. They do not even depend on the number of impressions. If there are R real impressions over the week, and Fake impressions, that the advertiser will receive 0.1 x R real impressions and 0.1 x F fake ones. Consequently, this system is not susceptible to competitor fraud of any kind (neither click nor impression fraud). However,evaluating how much a certain percentage of impressions on a specific website is worth remains problematic. If such an evaluation is based on the average number of impressions of a site it remains vulnerable to impression fraud. Goodman himself admits that "the pay-per-percentage model is not appropriate for all kinds of affiliate advertising; in particular, it is most appropriate for high volume sites". He recommends using a rating company to estimate the traffic on these sites, basing the cost-per-percentage on the estimates. For the publisher, this model is just as advantageous as Pay-Per-Impression, since his payment depends solely on the services he/she provides. Although the advertiser is safe from competitor fraud, this scheme leaves him susceptible to publisher Impression fraud if he/she does not advertise on a trustworthy site. Furthermore, the model is worse for him/her than Pay-Per-Click because the price is not based on the effectiveness of the ad.

4.2 Related research for click fraud detection in PPC model

4.2.1 Duplicate Detection

Metwallys group at UCSB proposed a solution, based on Bloom Filters, to detect duplicates in data streams[Metwally et al., 2005b]. [1] Their technique could be utilized in various web-based applications including click fraud detection. Assuming that these streams could be click activities, duplicates represent one type of a click fraud in advertisement networks. A comprehensive set of experiments, using both real and synthetic click streams, showed high detection rate of duplicates, and very low error rate.

In the proposed method they start by allocating M bits, where M is $O(N)$, and N is the estimated size of the processed window. As illustrated in Figure 4.1, using $d = 17$ independent hash functions, they test every new element on the Bloom Filter structure of the previously observed elements, and then insert it into the Bloom Filter structure. Before setting any of the d cells to 1, the cell is tested whether it has been set before to 1, or not. The element is not counted as a duplicate if at least 1 bit was switched from 0 to 1, and is considered to be a duplicate otherwise. Both M and d can be determined according to the required error rate, and the expected window size.

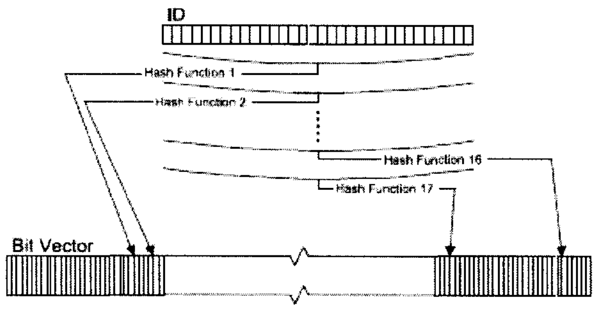


Figure 4.1: The Classical Bloom Filter

The original Bloom Filters use one hash space and hashes all the elements onto it. However, when the authors developed the proposed solution, independently of Bloom

[1] A Bloom Filter is a data structure that was proposed to detect approximate membership of elements. Given two sets X, and Y, the Bloom Filter algorithm would loop on every element in set X, to check if it belongs to set Y.

Filters, they used separate space for different hash functions, as sketched in Figure 4.2.

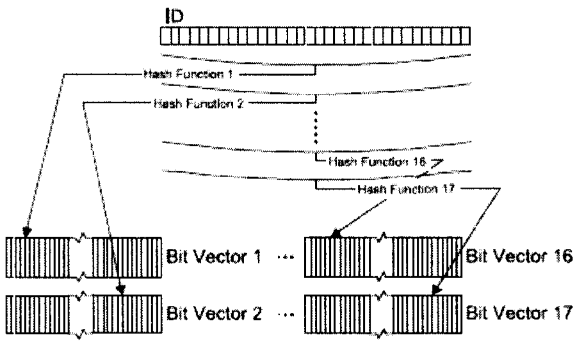


Figure 4.2: The Classical Bloom Filter

In order to differentiate between authentic and fraudulent clicks, the advertising commissioner "tracks individual customers by setting cookies. Duplicate clicks within a short period of time, a day for example, raise suspicion on the commissioners side" [Metwally et al., 2005b]. Duplicate detection can, no doubt, detect amateur click fraud, where the fraudster operates from one or a handful of computers. However, it is clearly inadequate when it comes to detecting distributed click fraud, where millions of computers simulate clicks on an advertisement from all over the world, since it relies on cookies (text files which the commissioner stores and accesses on the users computer) for detection. In the case of distributed click fraud, every computer in the attackers vast network will have its own individual cookie. Moreover, an attacker who knows what he/she is doing will just delete the cookies after each "click", leaving no duplicates to be detected.

4.2.2 Association Rules

Metwally, Agrawal and Abbadi have also proposed a solution to the referrer click fraud. They propose encouraging ISPs (Internet Service Providers) to provide the data stream necessary to detect this kind of click fraud. This data stream would contain the HTTP requests to page P, which might or might not be fraudulent. They would devise an algorithm to detect associations between one or more sites that refer to P very frequently, and clicks on an ad on P. If strong associations are found, it is very probable that P is using one or more "decoy" websites in order to commit undetected click fraud [Metwally et al., 2005a].

In [Metwally et al., 2005b] the same authors extend the ideas of streaming data analysis, and they established schemas that would detect fraud attacks of many classes based on their classification of click fraud. Special attention is given to the problem of detecting automated click fraud activities. They develop the algorithm called *Streaming Rules* which reports association rules, using limited processing per element and minimal space. The algorithm has a possibility to detect more sophisticated attacks such as the one identified by Anupam et al. [AnupamL et al., 1999]. The system is tested with synthetic data but also with ISP logs from an anonymous ISP where some suspicious relationships are detected. The focus in their research is on the commissioner part of the click fraud detection process, and it is very specific because commissioners are optional in the Internet advertising network and many advertisers directly put advertisements on publishers' sites. In another paper [Metwally et al., 2007] the authors are analyzing coalition hit inflation attacks. Still, they agree that the area is open for other classifications of hit inflation attacks, for new techniques to detect automated behavior, and more effective real world solutions.

4.2.3 Classification of URLs

Malicious Web sites are a cornerstone of Internet criminal activities. Referrer click fraud is one form of URL related fraud. As a result, there has been broad interest in developing systems to prevent the end user from visiting such sites. Researchers have address the problem of URL classification, using statistical methods to discover the tell-tale lexical and host-based properties of malicious Web site URLs.

Justin Ma et al. have described an approach for classifying URLs automatically as either malicious or benign based on supervised learning across both lexical and host-based features [Ma et al., 2009]. They argue that this approach is complementary to both blacklisting, which cannot predict the status of previously unseen URLs and systems based on evaluating site content and behavior, which require visiting potentially dangerous sites. Further, they show that with appropriate classifiers, it is feasible to automatically sift through comprehensive feature sets (i.e., without requiring domain expertise) and identify the most predictive features for classification.

The work by Garcra et al. use logistic regression over 18 hand-selected features to classify phishing URLs [Garcra et al., 2007]. The features include the presence red flag key words in the URL, features based on Google's Page Rank and Google's Web page quality guidelines. In their experiments they achieved a classification accuracy of 97.3% over a set of 2,500 URLs. McGrath and Gupta do not construct a classifier but nevertheless perform a comparative analysis of phishing and non-phishing URLs [McGrath and Gupta, 2008]. The features they analyze include IP addresses,WHOIS records (containing date and registrar-provided information only), geographic information, and lexical features of the URL (length, character distribution, and presence of pre-defined brand names).

Provos et al. perform a study of drive-by exploit URLs and use a patented machine learning algorithm as a pre-filter for VM-based analysis [Provos et al., 2008]. They extract content-based features from the page, including whether IFrames are "out of place," the presence of obfuscated javascript, and whether IFrames point to known exploit sites. CANTINA classifies phishing URLs by thresholding a weighted sum of 8 features (4 content-related, 3 lexical, and 1 WHOIS related) [Zhang et al., 2007]. Among the lexical features, it looks at dots in the URL, whether certain characters are present, and whether the URL contains an IP address. The WHOIS-related feature CANTINA examines the age of the domain.

4.2.4 Non standard approaches for click fraud detection in PPC model

There are a number of other solutions proposed in the literature for avoiding click fraud in PPC advertisement model. One suggestion is to charge based on user's actions, i.e., the publisher gets a premium only after the successful conversion of the ad, meaning the user's visit to the advertiser website and performing an action such as buying an item or signing up for a service. There are a number of basic attempts. Such as by means of tracking cookies, however these efforts make up a negligible portion of the current advertising revenue on the Internet. Immorlica et al. [Immorlica et al., 2005] analyze the click fraud learning algorithms to compute the estimated click-through rate. They focus on a situation in which there is just one ad slot, and show that fraudulent clicks can not increase the expected payment per impression by more than $O(1)$ in a click-based algorithm. However the complexity of the inferred algorithm and the need for click-through rate estimation would make it impractical as it also deviates from the pay per click model, to pay per view model, which is the least desired model in the modern advertisement world where bidding for space is of critical importance.

Gandhi et al. [Gandhi et al., 2006] from Indiana University at Bloomington proposed a new type of camouflaged click fraud attack on the advertising infrastructure so called "badvertisement". This stealthy attack can be thought of as a threatening mutation of spam and phishing attacks. The target of this attack is the unwitting advertiser and it could be very serious with significant revenue potential for its perpetrators. The attack was experimentally verified by corrupting the JavaScript file that is required to be downloaded and executed by a clients web browser to publish sponsored advertisements.

The Jacobsons paper [Jakobsson et al., 1999] discusses the measures for the visibility and degree of success of an ad. It had be found that traditional methods perform very poorly in an Internet setting, due to lack of trust, lack or reliable metering methods, and a lack of direct feedback. Therefore, they introduce the concept of e-coupons. E-coupons can be viewed as the electronic counterpart of traditional coupons in mailboxes or newspapers. In order to benefit from e-coupon, a customer needs to interact with the merchant, which allows checking the validity of the e-coupon and represents direct feedback of the impact of the ad campaign. The authors argue that the proposed scheme is safe and meets the strongest security requirements.

Another solution, proposed by Juels et al., tries to authenticate valid clicks. In [Juels et al., 2007], they proposed a credential-based approach to identify premium clicks (i.e. good clicks) instead of excluding invalid clicks. If a user has committed legitimate behaviors (e.g. purchases), the clicks from his/her browser are marked as premium clicks and cryptographic credentials are stored in the browsing cache for authentication. This approach, however, is still subject to the attack presented in this paper, where click fraud may be committed in a browser used by a legitimate user. If credentials have been stored due to the legitimate behaviors from that user, fraudulent clicks will also be identified as premium clicks.

Haddadi et al. presented a simple detection strategy, what they called as using Bluff ads [Haddadi, 2010]. These are sets of irrelevant ads displayed amongst user's targeted ads, which should never be clicked on. Together with threshold detection, IP address monitoring and profile matching techniques, bluff ads can be used to make it more complicated for the botnet owners to train their software, or a human operator. The bluff ads also may have a comfort factor of decreasing the user's negative perceptions by reducing the number of accurately targeted ads.

Important discussion about a click fraud concept and its interpretation are also given in [Tuzhilin, 2006]. Thuzilin claims that between the obviously clear cases of valid and invalid clicks, lies the whole spectrum of highly complicated cases when the clicking intent is far from clear and depends on a the range of factors, including the parameter values of the click. This intent cannot be operationalized and detected by technological means with any reasonable measure of certainty. Therefore, the invalid click detection methods need to be developed without a proper formal specification of invalid clicks. The standard commercial methodologies, which are measuring mainly the true rate of invalid clicks, are only guesstimates at best.

Many researchers [Banerjee and Ghosh, 2001], [Buchner et al., 1999] have been carrying out similar research for web usage and user behavior analysis. Web usage characteristics could be indicators for click fraud detection, especially for software clicks. Web usage mining is the application of data mining techniques to discover usage patterns from Web data, in order to understand and better serve the needs of Web-based applications [Cooley et al., 1999]. In the same paper, the Web usage mining is parsed into three distinctive phases: preprocessing, pattern discovery, and pattern analysis. It also clarified the research sub direction of the Web usage mining, which facilitates the researchers to focus on each individual process with different applications and techniques.

4.3 Summary of Current Industrial and Research Solutions

We briefly categorize the commercial solutions and research activities in three categories based on the data they use:

1. Server side approaches such as Metwally et al.[Metwally et al., 2005b], Mahdian et al.[Mahdian, 2006], Banerjee et al.[Banerjee and Ghosh, 2001], Google etc. They are using the data direct from server without caring much about the users activities. These methods analyze the IP, double clicks, or web pages from the server side or ISP side.

2. Client side approaches such as Gandhi [Gandhi et al., 2006], Whosclickingwho.com, and Anupam [AnupamL et al., 1999] etc. They study the client side user activities to propose solutions for the click fraud problem.

3. Non standard approaches such as Jacobson [Jakobsson et al., 1999] and Goodman [Goodman, 2005] etc. proposed different approaches to the existing problem. However, these solutions are not standard and not widely accepted by the existing internet industry.

Limitations and drawbacks of existing solutions, fall in to each category stated above, are discussed below.

Even though commercial solutions and research trends exist, there are unaddressed threats to the pay-per-click model. For example, most of the commercial solutions can not detect software click fraud. If click traffic is generated by robotic software, the software may not execute the javaScript or may not load iframe tags. Since, there are no user activities, category 2 solutions will not have any information about sources of such clicks. Since category 1 solutions do not collect client side data, they will not know whether click has user activities or not. Then the detecting hosts in both category 1 and 2 solutions will have partial evidence about any of these traffic; thus, will not report any kind of fraudulent click.

Software click bots provide the largest threat to the PPC advertising model. Despite a few early efforts towards identifying special classes of click-bot attacks such as clickbot.A and Bahama bot, there has been a little work on studying bot-generated click traffic. A number of challenges make this task difficult. First, the amount of data to process is often huge, on the order of terabytes per day. Thus any method that mines the data for identifying bot traffic has to be both efficient and scalable. Secondly, most of these data are not disclosed due to privacy, security and business policy issues. Furthermore, with many bot-net hosts available, attacks are getting increasingly stealthy with each host submitting only a few clicks to evade detection. Therefore, click bot detection methods cannot just focus on aggressive patterns, such as in Bahama bot, but also need to examine the low rate patterns that are mixed with normal traffic. Third, attackers can constantly craft new attacks to make them appear different and legitimate; thus we cannot use the training-based approaches that derive patterns from historical attacks. Finally, with the lack of ground truth, evaluating detection results is non trivial and requires different methodology and metrics than the detection methods.

Most of the existing solutions do not collect real time user activity information, such as mouse movement, keyboard pressed etc., which is very important in detecting adware, malware, and simulated clicks. When a user session is observed, if there are user activities reported, they provide positive evidence for the existence of a human user. Such evidence can be used instead of asking a user to verify a “CAPTCHA”, which more and more websites find annoy the actual human user.

Another significant problem of the existing commercial solutions is that they do not have a way to prevent click fraud dynamically. Since most of these solutions are merely click fraud reporting solutions they provide reports at the end of the day, at the end of the week or sometimes at the end of the month. By the time they found out about the fraud it may be too late to take necessary actions against the fraudulent sources because fraudsters will change their method of attack to go undetected.

Most PPC service providers currently approach the problem of click fraud by attempting to automatically recognize fraudulent clicks and discount them. Fraudulent clicks are recognized by machine learning algorithms, which use information regarding the navigational behavior of users to try and distinguish between human and robot generated clicks. Such algorithms are mainly built using rule based techniques and most of them are classification systems, even though a few score based systems are also reported.

Most of the click fraud solution providers, including search engines and third party solution providers, claim their rule-based expert system is the best among the others taking the advantage of keeping "rules" as a secret weapon. They do not disclose information about the set of rules due to fear of competition. This situation even led to multi-million dollar settlements in recent years. Due to the lack of verifiability of click fraud solutions, it is inevitable that the trust between service providers and advertisers is degraded. Since real-world click fraud solutions are usually kept secret for fear of competition, it is practically impossible to study many of them in a single context. If a mechanism for the modeling of knowledge and validation (KV) for rule based expert systems exists, solution providers will be able to use it to verify their systems without revealing the implementation details of the rule base.

We tried to address these issues in the Pay Per Click advertising model. Chapter 5 and 6 we introduce our proposed methodologies.

CHAPTER 5

COLLABORATIVE CLICK FRAUD DETECTION AND PREVENTION

Sometimes it is hard to get solid evidence of the existence of click fraud based on the data collected. In some cases the true intent of a click can be identified only after examining deep psychological processes. For example, a person might have clicked on an ad, looked at it, went somewhere else but then decided to have another look at the ad shortly thereafter to make sure that he/she got all the necessary information from the ad. It is hard to say the second click is click fraud. However, we can evaluate the second click as less valuable than the first click, or give less quality score for any multiple clicks for a single user.

Initial phase of our research is based on the CCFDP system that was developed by Dr. Li Ge[Ge and Kantardzic, 2006]. In this research we do not try to determine the real intention of the web users and classify as binary (fraud and not fraud). We assign each click with a score value which estimates the quality of user activities, and also measures the difference between characteristics of a given click and averages collected through time in the click database. The CCFDP system integrates raw user activity data with the derived factors to determine the quality of each click. Preliminary list of factors, which describe user behavior and intentions, includes: Software clicks, No User Activities, Repeat Visitors, Suspicious User Agent Keyword, History Count, Blacklist Referrer, No Cookie or JavaScript allowed, IP and Permanent Cookie inconsistency, Web users location and IP location mismatch, IP location analysis, Page Activity Analysis, and Referrer Rate analysis. The CCFDP system establishes an arbitration system to evaluate the quality of every click referred from publishers, thus protecting advertiser from click fraud.

The CCFDP system takes collaborative approach to determine click quality by analyzing detailed user activities, matching the client and server side logs (Figure 5.1).

Since the CCFDP system logs both, we collect the data about the detailed activities of individual web user, which enable us analyzing web users behavior and estimate a quality score based on their activities. Recorded users activities on a web page can indicate their interests, browsing habit, and most important, estimating their intention. We use quality score to estimate the users intention instead of labeling clicks as fraud/not-fraud. After applying data mining method in the collected database, we analyze and quantify twelve factors for click score calculation. Three analyzes, History Factor Analysis, Individual Factor Analysis and Sequence Factor Analysis are implemented in the system for quality score calculation. The history click database for a site is used to build baselines to help determine the quality for new clicks while sequence factor analysis focuses on the relations between clicks. We study the user activities also by comparing baselines in databases for different website contents and different origins, e.g. Natural Traffic (Non-paid traffic) and Pay-Per-Click traffic (Paid traffic), thus, finding the suspicious activities.

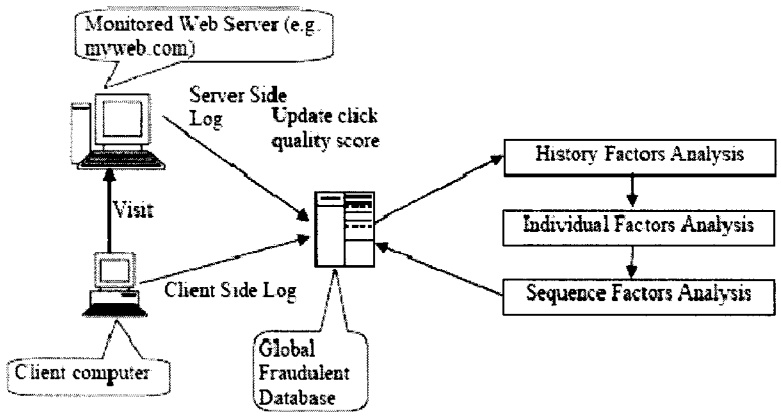

Monitored Web Server (e.g. myweb.com)
Server Side Log
Update click quality score
Visit
Client Side Log
Client computer
Global Fraudulent Database
History Factors Analysis
Individual Factors Analysis
Sequence Factors Analysis

Figure 5.1: Initial version of the CCFDP system

The core part of this system is the Global Fraudulent Database (GFD), which stores the real-time server side log, client side log, and computed fraud score for the given click. The fraud score is not based only on a single click characteristics; it is based on the time and space context of the click event. When a web user visits the monitored web site, both the server side log and client side log data are stored in the Global Fraudulent Database. A quality score for each click is computed and normalized from 0 to 1, and it is used to indicate the quality of a click. A score of 0 means valid click and a score of 1 means click fraud. The click score is usually a fractional value between 0 and 1 representing ambiguity in classification (fraud/not-fraud) of a current click. Summation of scores for all clicks on the current site gives the integrated score for the quality of click traffic on the site.

In the Section 5.1 we detail the hardware architecture of the CCFDP system, while Section 5.2 is dedicated to the operation architecture of the system. Section 5.3 explains the data collection process and Section 5.4 describes the actual click scoring process.

5.1 NetMosaics Hardware architecture

The NetMosaics hardware is currently implemented at NetMosaics.com Louisville data center. It is implemented as a dedicated Windows©server and follows most current industrial standards for web server hosting. Figure 5.2 shows the logical network diagram of the NetMosaics.

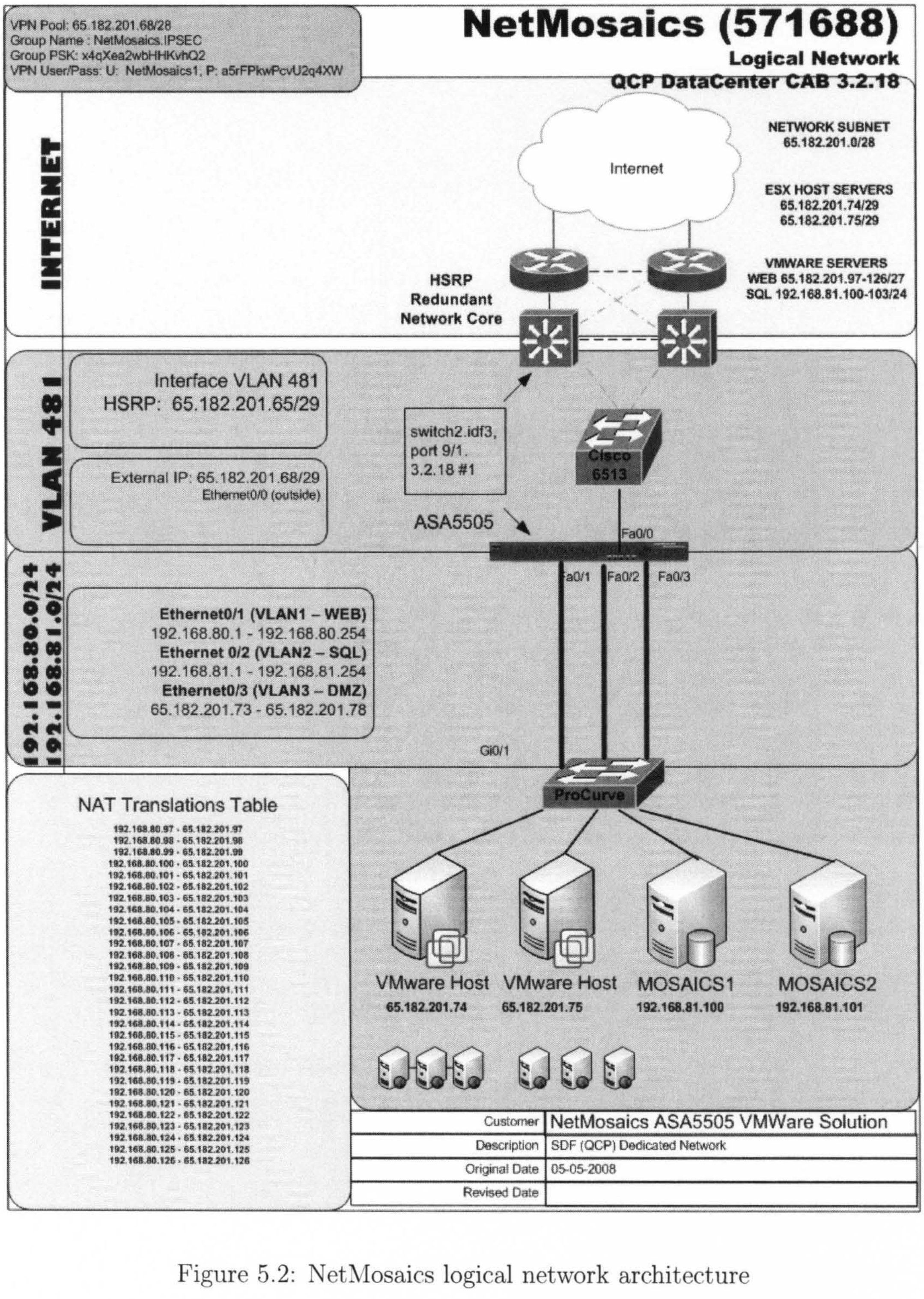
5.1.1 Reading the NetMosaics hardware diagram

1. Network Subnet:

   The network subnet is 65.182.201.0/28. A subnetwork, or subnet, is a logically visible, distinctly addressed part of a single Internet Protocol network. The process of subnetting is the division of a computer network into groups of computers that have a common, designated IP address routing prefix. Subnetting breaks a network into smaller realms that may use existing address space more efficiently, and, when physically separated, may prevent excessive rates of Ethernet packet collision in a larger network.

2. HSRP Redundant Network Core:

   HSRP stands for Host Standby Router Protocol. It is a Cisco proprietary redundancy protocol for establishing a fault-tolerant default gateway, and has been described in detail in RFC 2281.



VPN Pool: 65.182.201.68/28
Group Name : NetMosaics.IPSEC
Group PSK: x4qXea2wbHHKvhQ2
VPN User/Pass: U: NetMosaics1, P: a5rFPkwPcvU2q4XW

NAT Translations Table

192.168.80.97 - 65.182.201.97
192.168.80.98 - 65.182.201.98
192.168.80.99 - 65.182.201.99
192.168.80.100 - 65.182.201.100
192.168.80.101 - 65.182.201.101
192.168.80.102 - 65.182.201.102
192.168.80.103 - 65.182.201.103
192.168.80.104 - 65.182.201.104
192.168.80.105 - 65.182.201.105
192.168.80.106 - 65.182.201.106
192.168.80.107 - 65.182.201.107
192.168.80.108 - 65.182.201.108
192.168.80.109 - 65.182.201.109
192.168.80.110 - 65.182.201.110
192.168.80.111 - 65.182.201.111
192.168.80.112 - 65.182.201.112
192.168.80.113 - 65.182.201.113
192.168.80.114 - 65.182.201.114
192.168.80.115 - 65.182.201.115
192.168.80.116 - 65.182.201.116
192.168.80.117 - 65.182.201.117
192.168.80.118 - 65.182.201.118
192.168.80.119 - 65.182.201.119
192.168.80.120 - 65.182.201.120
192.168.80.121 - 65.182.201.121
192.168.80.122 - 65.182.201.122
192.168.80.123 - 65.182.201.123
192.168.80.124 - 65.182.201.124
192.168.80.125 - 65.182.201.125
192.168.80.126 - 65.182.201.126

Customer	NetMosaics ASA5505 VMWare Solution
Description	SDF (QCP) Dedicated Network
Original Date	05-05-2008
Revised Date	

Figure 5.2: NetMosaics logical network architecture

3. Multi layer switch:

   A multilayer switch (MLS) is a computer networking device that switches on OSI layer 2 like an ordinary network switch and provides extra functions on higher OSI layers.

4. Cisco catalyst switch:

   The 13-slot Cisco Catalyst 6513 Switch chassis is ideally suited for high performance, high port density Fast Ethernet and Gigabit Ethernet aggregation in all parts of the network, including the access, distribution, and backbone layers as well as the server farm and data center environments. With up to 12 payload slots available, the 13-slot chassis offers industry-leading 10/100/1000 Gigabit Ethernet and 10 Gigabit Ethernet port densities while providing high levels of network resilience.

5. ASA 5505 Firewall:

   Cisco ASA 5500 Series Adaptive Security Provides intelligent threat defense and secure communications services.

6. Virtual LAN:

   A virtual LAN, commonly known as a VLAN, is a group of hosts with a common set of requirements that communicate as if they were attached to the same Broadcast domain, regardless of their physical location. In NetMosaics setup we have three virtual LANs.

   - Ethernet0/1 (VLAN1 - WEB): Not publicly accessible. Use NAT table
   - Ethernet0/2 (VLAN2 - SQL) : Not publicly accessible. Use NAT table
   - Ethernet0/3 (VLAN3 DMZ): Publicly accessible. DMZ, or demilitarized zone is a physical or logical subnetwork that contains and exposes an organization's external services to a larger untrusted network, usually the Internet.

5.2 The operational architecture of the CCFDP system

The operational architecture of the CCFDP system is given in Figure 5.3.

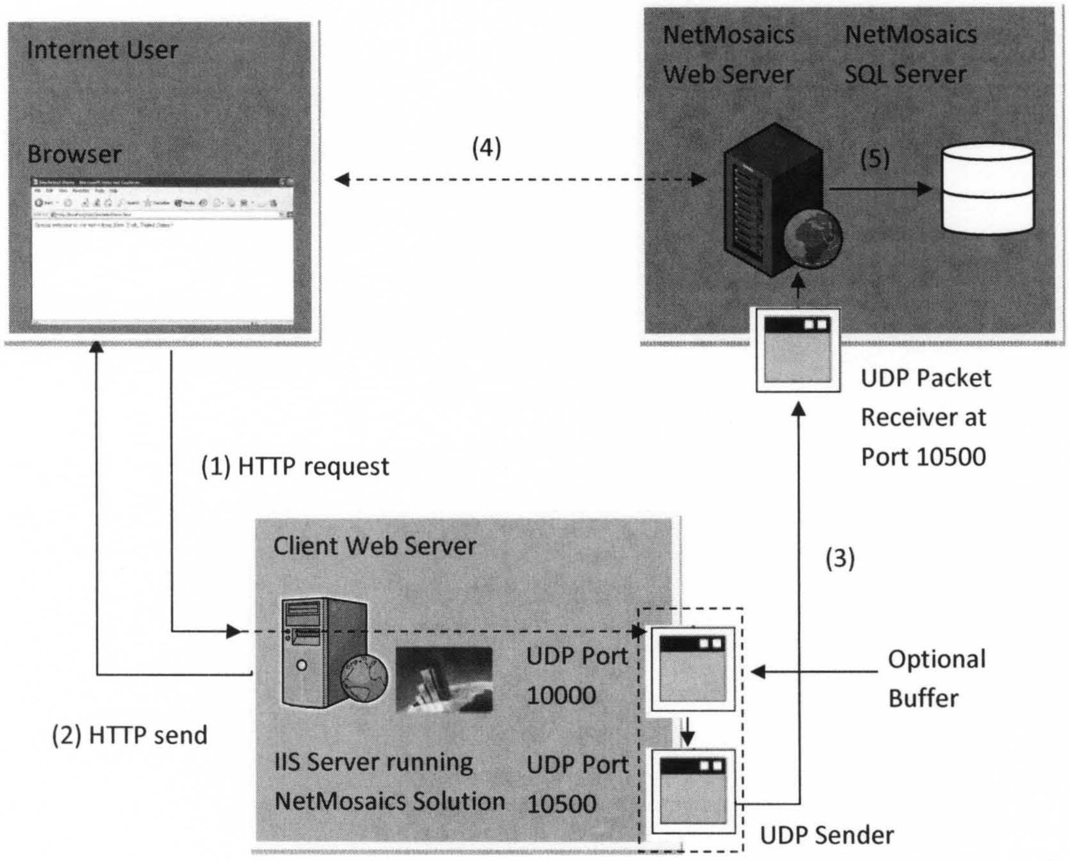


Figure 5.3: CCFDP operational architecture

The three main components in the CCFDP are:

1. The NetMosaics SQL server or the Global Fraudulent Database (GFD).

2. Monitored site which is a web server.

3. Client computer which could be normal user, click fraud user or software user.

We will explain each component in detail below.

5.2.1 Global Fraudulent Database (GFD)

GFD stores the server side log, client log, and fraud score report data. The database contains three main tables. They are serverlog, clientlog, and serverclienttracking tables. Tables "serverlogx" and "historyserverlogx" store server-side of click information and "clicktracking" and "historyclicktracking" store client-side of click information including the post click event data. The server side log and the client side log are matched by trackingID. In server side log, IP, User Agent and Referrer are three important parameters to detect and predict fraud. Other parameters include the logging time and the site name. Field "ID" is the primary key for serverlog table and its structure is shown in Table 5.1.

Table 5.1: Structure of the server side log table

Field Name	Field Type	Allow Nulls
ID	bigint	No
TrackingID	varchar(50)	Yes
IP	varchar(20)	Yes
Referrer	varchar(255)	Yes
UserAgent	varchar(255)	Yes
Location	varchar(512)	Yes
Site	varchar(50)	Yes
WebServerIP	varchar(20)	Yes
insertDate	datetime	Yes

The client side log includes client computer settings, such as screen width, screen height etc.; client activities inside the pages, such as mouse click, keyboard click, mouse over etc.; client browser setting, such as, javascript enabled and java enabled, allowing cookie etc. The IP Location table is the IP geographic location, which includes the owner of the IP, postal code, and latitude and longitude information. Field "ID" is the primary key for clientlog table and its structure is shown Table 5.2.

Table 5.2: Structure of the client side log table

Field Name	Field Type	Allow Nulls
Id	Int (4)	No
Html	Varchar(1000)	Yes
Html before	Varchar(1000)	Yes
Html after	Varchar(1000)	Yes
Mouse x	Int (4)	Yes
Mouse y	Int (4)	Yes
Clickitem x	Int (4)	Yes
Clickitem y	Int (4)	Yes
Size x	Int (4)	Yes
Size y	Int (4)	Yes
Event type	Varchar(20)	Yes
Time to click	Bigint(8)	Yes
trackingID	Varchar(50)	Yes
Keystrobecount	Int (4)	Yes
insertDate	datetime	Yes

In fraud detection process, the location is an important indicator of a click fraud. Therefore Some additional tables were used in calculation of the originating country for a given IP. These tables are: GeoIPCity, GeoIPCityBlocks, GeoIPCityLocation. For the purpose of finding the country only GeoIPCity table was used. Its structure is shown in Table 5.3.

The IPScore, ReferrerScore, CountryScore, and UserAgentScore tables dynamically update the activity scores based on the client side IP, Referrer, Country and User Agent parameters. The ReferrerScoreSoftClick, IPScoreSoftClick and UserAgentScoreSoftClick tables are software click fraud score based on the existence of server side log, and its match with client side. Figure 5.4 is the detailed diagram of the GFD.

Table 5.3: Structure of the GeoPICity table

Field Name	Field Type	Allow Nulls
startIpNum	Varchar(20)	Yes
endIpNum	Varchar(20)	Yes
Country	Varchar(100)	Yes
Region	Varchar(100)	Yes
City	Varchar(100)	Yes
postalCode	Varchar(20)	Yes
Latitude	Float(8)	Yes
Longitude	Float(8)	Yes
dmaCode	Varchar(10)	Yes
areaCode	Varchar(10)	Yes
intStartIP	Bigint(8)	Yes
intEndIP	Bigint(8)	Yes

5.2.2 Monitored site

CCFDP data collection works in 2 connected parts, server and client. They are separate software and may be run on separate servers, but they are coordinated by one central element. The central element that binds everything together is a server module that must be inserted into the website that is being tracked.

Currently, this module is developed in C#.NET that runs on IIS 7.0 (Windows^{TM} Server 2008) or later versions and C++ .NET that runs on IIS 6.5 and earlier versions. Both implementations are discussed later in this section. This module intercepts all incoming requests to a webpage, generates a unique ID for this user session, and attach a line of java script code into the outgoing webpage.

Notice that this way the original webpage never needs to be modified or updated. The server module takes care of all that behind the scene. In fact, the original webpage file

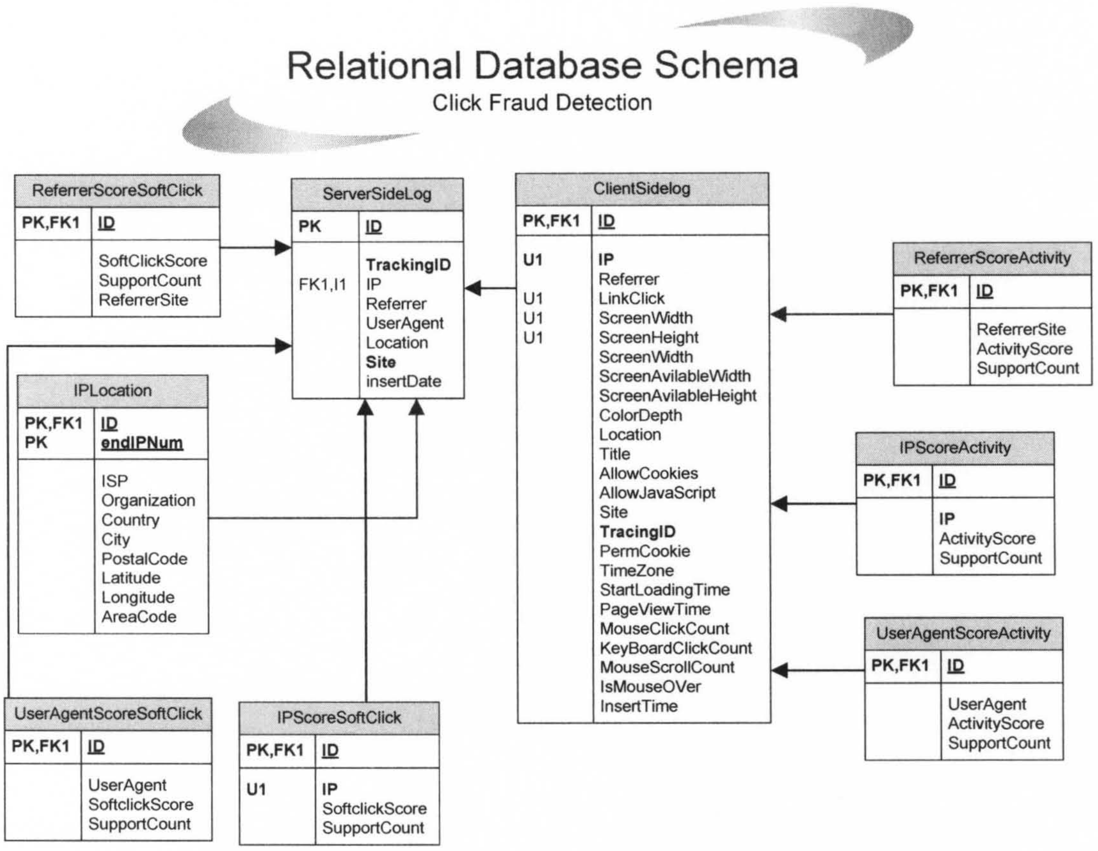


Figure 5.4: GFD detailed diagram

will never be changed, only the user of the webpage will see the javaScript code attached. The author of the webpage will never be aware of the javaScript code being attached.

After installing the NetMosaics module at the end of the source file, a javaScript line at the bottom of the file can be seen as similar to what is shown in Figure 5.5.

```
</body>
</html>
<script language="javascript" src="http://65.182.201.98/activitytracking.asp?
 cb=6015be3656d84551a235ef951e8d12c0&site=VDWS571688WEB3/D:\Websites\thebestmusicsitesorg"></script>
```

Figure 5.5: javaScript executed on client's webpage

The NetMosaics Module

When a user opens a browser and navigates to a website, he/she sends out an http request to a web server, as is defined in RFC 2616. The HTTP protocol is an application level protocol, which is above TCP/IP protocol. Figure 5.6 is a sample HTTP request sent out from the client computer.

```
GET /mypc/browserinfo/
HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-excel, application/vnd.ms
Referer: http://www.internetfrog.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; )
Host: www.internetfrog.com
Connection: Keep-Alive
Cookie: AreCookiesEnabled=476558864; permCookies=972049659; ASPSESSIONI=BFPFAEKAKIOELDK
```

Figure 5.6: Sample HTTP request

In Figure 5.6, the sample HTTP request contains most server side log parameters, except IP, which is a network layer parameter. Besides the server side parameters, we are interested in, some other parameters sent to the web server. They include HTTP version, accept file format, language, encoding, connection status, and cookies etc. Those parameters are defined in RFC 2626.

As the client computer sends the request to the web server (for example IIS), the server generates the page and forwards it to the client computer. We created an ISAPI filter(managed module in IIS 7) to record such requests and this process is displayed in Appendix A.

We are only concerned about the successfully generated pages, with HTTP success code 200 (From the example presented in Figure 5.7, we can find the response status code, which is HTTP/1.1 200 OK). At the same time, we only process requests for text/html. Other contents, such as image, css, video clip etc are supplementary and will be sent to the client computer directly. If the major text/html page is blocked, there should not be any requests for image, css, and video clip etc. The system sends logs to Global Fraudulent Database (GFD), and queries for the fraud score before the response is sent back to client computer. If the fraud score is higher than the threshold, a warning page will be sent to client computer instead. Otherwise, a unique 128-bit number, Globally Unique Identifier (GUID) will be added to the tracking javascript code at the bottom of the page. This tracking id is introduced in Hardware Architecture.

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 27 Sep 2005 18:04:35 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Content-Type: text/html
Set-Cookie: BrowserMirrorPersistent=9%2F27%2F2005+2%3A04%3A35+PM; expires=Wed, 27-Sep-2006 18:04:34 GMT; path=/
Set-Cookie: BrowserMirrorSession=9%2F27%2F2005+2%3A04%3A35+PM; path=/
Cache-control: private
```

Figure 5.7: HTTP Response header

UDPServer application

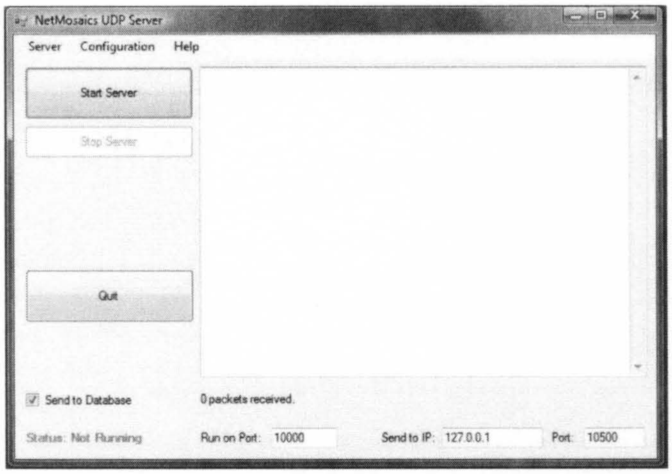
The UDPServer is an application that can receive local UDP packets and queue them in a buffer. This software is installed on the client's web server and set to receive packets from that web server's NetMosaics module. The packets are written to a database locally on the server, and then they are sent off to the main NetMosaics receiving server (client side database setup is discussed later in this section). This helps ensure no click data is lost if the NetMosaics server is unreachable or busy but does have the downside of having to install a local version of the NetMosaics database.

This application's second major task is to run a loop internally that checks if buffer database contains any entries, if there are entries, the application creates UDP packets from database entries, and sends those UDP packets to the network address and port specified at the bottom of GUI in Figure 5.8 (the port should be open).

Once "Start Server" button is pressed, the application is listening to any incoming UDP packets on the port that you have specified the UDP server to listen to. Once a packet arrives, the application reads the packet to see if it's server or a client packet. The application then copies the information of the packet into the "buffer" database. And goes back to waiting for more packets.

5.3 Data Collection process in CCFDP

The data collection is a five step process. Each of the five steps is shown in Figure 5.9 and described in detail below.



☑ Send to Database

0 packets received.

Status: Not Running

Run on Port: 10000

Send to IP: 127.0.0.1

Port: 10500

Figure 5.8: NetMosaics UDP Server

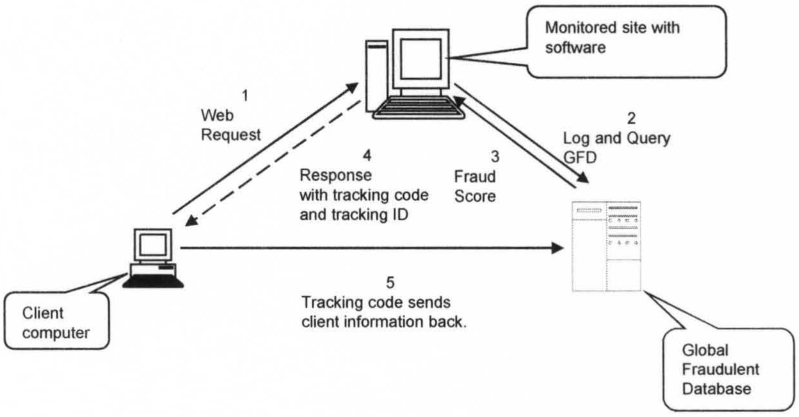

Monitored site with software
1
Web Request
2
Log and Query GFD
4
Response with tracking code and tracking ID
3
Fraud Score
5
Tracking code sends client information back.
Client computer
Global Fraudulent Database

Figure 5.9: Data Collection Process in CCFDP

1. A public internet user requests a webpage from client's website.

   The server side data collection starts at this moment. The server module (that waits on IIS) performs three primary tasks.

   - Intercepts user web page requests.
   - Collects server information using various API from IIS and Windows Server 2008, then creates a UDP packet from this information, and finally sends the packet over to UDP Net Mosaics Data Collection Server using UDP protocol (discussed in detail in step 3).
   - Attach client side tracking code.

2. The NetMosaics module executes on the IIS server, generates a permanent and a temporary tracking code and writes a JavaScript to the webpage. The server DLL module attaches a short string to outgoing HTML reply to the user. This module does not actually insert any final JavaScript code. (That would be inefficient and unscalable.) Instead, the module adds a reference to a JavaScript file. Thus, when the HTML page arrives to the users web browser the first thing that will happen is that web browser is going to download the reference and insert that reference into the HTML page, and only then the user sees the final page and JavaScript code that NetMosaics inserted finally runs. If the user looks at the source page of the received webpage, there should be a JavaScript that looks similar to the Figure 5.5.

   The attached javaScript code tells the web browser that before displaying a page to the user, the browser must go to the above address, load and insert that java script file into HTML code that the web user will finally see. The java script code and setup will be discussed later. This code will later record and send all client interactions with the webpage. Thus, anytime a user clicks or closes the page, a JavaScript event will occur, which will run Net Mosaics JavaScript code and send over a UDP packet that will contain the information about the event (for example, the location of the click).

   Most of the parameters in step 2 are defined in Hypertext Transfer Protocol HTTP/1.1 (RFC 2616 [(IETF), 1999]). We added two extra cookies and a tracking ID besides the RFC header for tracing purpose. A permanent cookie is the cookie we implant to client computer with the expiration date of 1 year and a session cookie will be expired whenever the client closes the connection session. We use those two cookies to identify client computers. Whenever the client computer connects to the same web site, the client permanent cookie will be sent to web server as part of the web request. A tracking ID will be added to the javascript code and sent to the client. The tracking code inside every page is presented in Figure 5.10.

   The number 521f5c9939d44463d886a905c2a2af8e9 in Figure 5.10 is the tracking ID.

```
<script language="javascript" src="http://www.clickfraudresearch.com/activity.js?site=mysite&trackingid=
521f5c9939d4463d886a905c2a2af8e9"></script>
<noscript><img src="http://www. clickfraudresearch.com/UserAct.js?site= mysite &js=false& trackingid =
521f5c9939d4463d886a905c2a2af8e9"  height=0 width=0></noscript>
```

Figure 5.10: javaScript code added to each web page

The purpose of this tracking id is to match the client side log with its corresponding server side log. In step 5 of the logging process, when the javaScript code executes on client side, it will collect the client side setting and log to the GFD.

Let us illustrate how the logging process works with an example. Suppose user A opens a browser and navigates to site www.mysite.com, the web browser sends the web request define in HTTP 1.1 to site www.mysite.com. Site www.mysite.com sends the web request parameters along with serialized tracking ID to GFD. GFD returns a fraud score S for a given click back to site www.mysite.com. If the fraud score S is less than a threshold value, site www.mysite.com sends the requested page and the tracking code to above client's browser. The client browser will display the page, and at the same time the above tracking code will execute on user As browser and report As activity to GFD. Since the same tracing ID appears in the two logs, it reveals the two log entries are connected.

3. The initial HTTP request packet (shown in a dashed arrow from process (1) to process (3) in Figure 5.9) and tracking ids are sent to port 10500 of the NetMosaics web server. This process is managed by the DLL modules. This can be achieved in two ways.

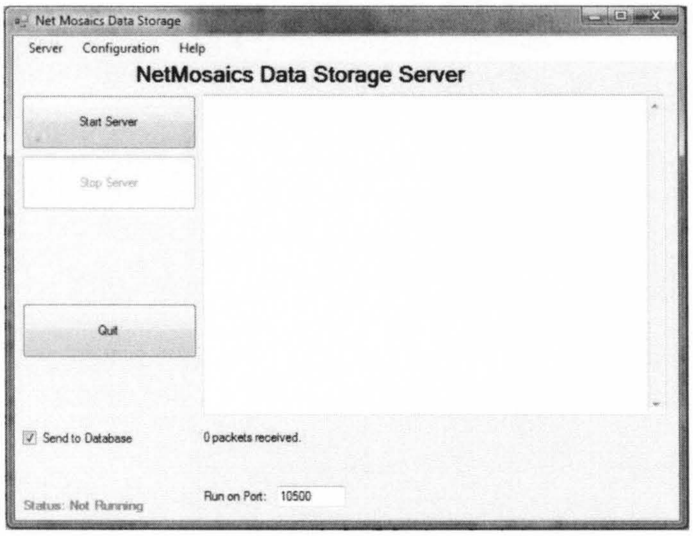
   - With a buffer database.
   - Without using a buffer database.

   (a) Data sent with a buffer database

   This is shown as the optional buffer in Figure 5.8. A buffer database is simply a mirror image of the final database. It is used to store the data until the data can be moved over to the final database server, in case the final server is experiencing slow performance or some network lag. Remember that this database is completely optional. This database is created on the client's web server.

(b) Data sent without a buffer database

The NetMosaics module installed inside the client's web server intercepts user web page requests, collects server information using various API from IIS and Windows Server operating system, then creates a UDP packet from this information, sends it over to UDP port 10500 of the NetMosaics data collection server through port 10000 of the local web server. As noted before, DLL modules create UDP packets and send them over network to the address and port you have specified (Figure 5.11).



0 packets received.

Run on Port: 10500

Figure 5.11: NetMosaics Data Storage Server

4. All packets are written to the NetMosaics SQL server.

DBTransfer Application: DBTransfer application runs on the NetMosaics hardware. The DBTransfer application receives UDP packets from servers all over the Internet running the NetMosaics Module. Packets are sent over port 10500 from the clients' web server module and are received on the port 10500 of the NetMosaics hardware. After they are received, they are written to the database so the algorithms can be run on the new rows.

5. The JavaScript executes on the user's browser and keep sending user activities to the NetMosaics web server.

5.4 Click Fraud Detection Model: Initial Version

The main characteristics of this system were,

(i) Integrates server side and client side activities to better detect click fraud activities.

(ii) Use more data from server and client compared to other available system, with the assumption that more data gives better analysis/results.

(iii) Introduce a fraud score between 0-1 instead of classifying clicks into valid/invalid.

Figure 5.3 shows the data sources used by the CCFDP initial version. Those sources can be divided into two categories as shown below. The logical structure of the click record is shown in Figure 5.12.

1. Direct sources which includes
   (a) Server computer
   (b) Client computer
   (c) Clicktracking
2. History data
   (a) Fraudulent Database
   (b) Blocking Database

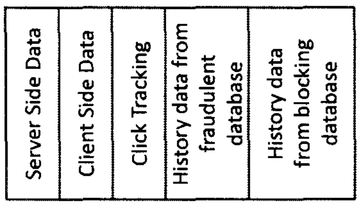

Server Side Data
Client Side Data
Click Tracking
History data from fraudulent database
History data from blocking database

Figure 5.12: Logical Structure of the click record

5.5 Click Fraud vs. Click Quality

Sometimes it is hard to get solid evidence of click fraud just based on the data collected. The click fraud is about the intention of the web users, who has no objective of buying the products or services advertised.

In other words, classification of clicks as valid or invalid is either impossible or not accurate. Just like Google, who does not like the concept of "fraudulent" click, and uses the term "invalid" click instead, we also found sometime it is difficult to determine the real intention of the clicks. The example from Dr. Tuzhilins [Tuzhilin, 2006] explains the difficulties in determining the intention of clicks: a person might have clicked on an ad, looked at it, went somewhere else but then decided to have another look at the ad shortly thereafter to make sure that he/she got all the necessary information from the ad. Is this second click invalid? To make things even more complicated, the second click may not be strictly necessary since the person remembers the content of the ad reasonably well (hence there is no real need for the second click). However, the person may not really like or care about the advertiser and decides to make this second click anyway (to make sure that he/she did not miss anything in the ad and his/her information is indeed correct) without any concerns that the advertiser may end up paying for this second click (since the person really does not care about the advertiser and his/her own interests of not missing anything in the ad overweigh the concerns of hurting the advertiser). Therefore, in some cases the true intent of a click can be identified only after examining deep psychological processes, subtle nuances of human behavior and other considerations in the mind of the clicking person. Moreover, to mark such clicks as valid or invalid, these deep psychological processes and subtle nuances of human behavior need to be operationalized and identified through various technological means, including software filters. Therefore, it is simply impossible to identify true clicking intent for certain types of clicking activities and, therefore, classify these clicks as valid or invalid.

Therefore, in the CCFDP initial version, we do not try to determine the real intention of the web users. We give each click a score which measure the difference between each click and average users activities. For example, the average user makes 5 clicks during the stay on a site, a user make no click on this site means the low quality of this visit.

Web users activities on a web page can indicate their interests, browsing habit, and most important, estimating their intention. Because of lack of recorded user activities, the searching engines, ads commissioners, and other advertisement agent have difficulty in detecting many kinds of suspicious clicks. Since the CCFDP system logs both client and server side logs, we have a record about each click with detailed activities. Analyzing web users activities CCFDP system gives a quality score.

We study the user activities in detail by comparing different website contents, and different origins, e.g. Natural Traffic (Non-paid traffic) and Pay-Per-Click traffic (Paid traffic), thus, finding the suspicious activities. To define precise score for each click, we analyze large number of potential factors which define suspicious activates and potential click fraud.

Based on this analysis, we selected and formalized set of factors which determine the quality of clicks. The factors are defined as follows:

1. Software clicks

   Software click means that a click has a server side entry without corresponding client side entry. If software requests a web page, there will be a server side log entry. However, the software neither request the following javascript code nor run the javascript code. There should be no corresponding client side log entry for this request. We exclude the search engine crawler traffic from click fraud detection although the search engine traffic is a form of software click. The software click rate is defined as:

$$R_{software\ click} = 1 - \frac{\sum Client\ side\ log}{\sum Server\ side\ log} \tag{5.1}$$

If the client user closes the browser or press the "Stop" before the web page is fully executed, there will be no client side log too. In normal situation for any site, this exception rate is low. If there is jump of Software Click Rate, the traffic is suspicious.

2. No User Activities

   The normal web page user will perform certain amount of activities during the viewing period. These activities could be mouse over, mouse click, keyboard input, scroll bar move, etc. If there are no activities inside a page, the click will likely be click fraud. The no activities rate for a site is defined as:

$$R_{No\ activities} = 1 - \frac{\sum No\ existing\ activities}{\sum Client\ side\ log} \tag{5.2}$$

   In some cases, the user closes the web browser before he makes any mouse or keyboard movements. In normal situation for any site, this exception rate is low, if there is jump in "No activities" rate, the traffic is suspicious.

3. Repeated Visitors

   There are two ways to count repeated visitors. Since we implemented a permanent cookie on the client computer, if the same user repeatedly visits the monitored web site, their permanent cookie will be recorded in our database. It is unlikely the two new assigned permanent cookies are identical. If the client browser does not accept cookies, the users' IP address can be an alternative way to identify repeated visitors. If the user keeps deleting cookies while visiting the website, factor (8) can be used.

   Sometimes, the AOL users or large intranet users may share the same gateway IP addresses when connecting to the internet. We may miscalculate the repeat visitor count for shared IP. In that case, IP and permanent cookies are complementary for repeat visitor detection. In some situations, the repeat visitors come from the same class "C" network (with net mask xxx.xxx.255.255) if they use a big proxy pool for outbound connection. For those visitors, we can detect them by IP location analysis.

It is normal for a user to repeatedly visit a site during a period of time. However, if the repeat count is substantially greater than the average repeat visitor count, we will flag the visitor as suspicious.

4. Suspicious User Agent Keyword

   Every normal web request carries a User Agent field. A lot of click fraud from adware or spyware uses suspicious keywords inside their User Agent field. We can immediately identify this kind of click fraud from the field. Some fraud includes well-known keywords such as Google, Yahoo, or Bing in their User Agent field, to give the impression that the traffic is from those sources. However, the IP is not from those companies based on the IP location match database.

5. History Count

   Javascript can report how many other web pages the current browser visited before visiting this page, although Javascript does not report the details of the visited web sites, this count is useful for detecting click fraud.

   First, if a site is referred by a search engine or other advertisement agent, the history count should not be 0. This is because the current browser must have the search engine or advertiser agent in its history. The conclusion is based on the fact the advertisement link does not bring up a new pop up browser. The condition is easy to be satisfied for many search engines, such as google.com, because neither google.com nor its Adsense program use pop up browser for their advertisement link.

6. No Cookie and JavaScript allowed

   Through our research, we find that for normal users, the rate to allow JavaScript and Cookie is very high for any site, from 95% to 99%. For some Per-Per-Click traffic, the setting is significantly different compared to normal traffic. The Per-Per-Click traffic may allow JavaScript and Cookie in 50% of the visitors. If the traffic does not allow Cookie and JavaScript, we will have a reason to suspect the intention of the user.

7. IP and Permanent Cookie inconsistency

   This criterion is directly linked to case (3) of repeated visitors. In normal cases, the IP and permanent cookie for a web user should be consistent. That means if a user revisits a website, the IP address and his permanent cookie should be a fixed pair. If a user cleans his cookies every time before visiting a monitored web site, we will log a new permanent cookie for each repeated visit. This will create an inconsistency between the IP and permanent cookie.

   There is still another case for IP and permanent cookie inconsistency, which is, a web user connects to the monitored site with the same permanent cookie, while his IP address is changing for every visit. Most likely the web user is changing proxy server for different visit. This IP and permanent cookie inconsistency should be flagged as suspicious.

8. Web users location and IP location mismatch

   In CCFDP system, Three location related parameters are logged, IP, web users local time, and time zone. The location indicated by those three parameters should be consistent by any web visitor. The web visitors location is calculated in three ways.

$$Location_{IP} = Longitude_{IP} \tag{5.3}$$

$$Location_{Time\ zone} = Time\ zone_{Java\ script} \tag{5.4}$$

$$Location_{Local\ time} = Hour(Log\ server\ time\ -\ java\ script\ local\ time\ -\ 5) \tag{5.5}$$

   If we find the IP time zone and Javascript time zone are significantly different, we can suspect the visitor uses a proxy server to connect to the web server. In the system, we set the time zone threshold difference to be +/-2 to accommodate for the day light saving time and other mobile situations.

9. IP location analysis

   IP country analysis is a very important method to detect click fraud, especially for those making advertisements world wide. The IP country distribution is relatively fixed for a web site. If we observe a substantial shift of the IP country distribution from its history distribution, the traffic is suspicious.

10. Page Activity Analysis

    For the page activity analysis, we focus on three parameters: Click Count, Page Depth and Page View Time. The average value for these three parameters over a period of time is calculated. The Click Count is the number of clicks during a web visitors staying on the web site. The average of Click Count for a web site over a period of time is defined as:

$$R_{Average\ click\ count} = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M_N} Clicks\ from\ the\ n^{th}\ visitor\ on\ the\ m^{th}\ page}{\sum_{n=1}^{N} M_N} \tag{5.6}$$

The Page Depth is the number of unique pages a visitor viewed for a web site. The average of Page Depth for a web site over a period of time is defined as:

$$R_{Average\ page\ depth} = \frac{\sum_{n=1}^{N} page\ depth\ for\ the\ n^{th}\ visitor}{N} \tag{5.7}$$

The Page View Time is the staying time on web page by a visitor. The average of Page View Time is defined as:

$$R_{Average\ page\ view\ time} = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M_N} Page\ view\ for\ the\ n^{th}\ visitor\ on\ the\ m^{th}\ page}{\sum_{n=1}^{N} M_N} \tag{5.8}$$

We constantly monitor the average click count for a web site. A dramatic change in any of the average values will indicate the change of activities and some suspicious event on a web site.

11. Referrer Rate analysis

    If a web site makes an advertisement with an advertisement agent, the time rate referred by a location is relatively fixed. For example, we observed the incoming traffic referred from "www.domainsponsor.com" was 12.3 per hour. The advertisement was with Google.com for "Music download" keywords and "www.domainsponsor.com" is Google.coms search partner. The definition of the referrer rate is:

$$R_{Referrer\ rate\ for\ website\ A} = \frac{\sum Traffic\ referred\ by\ site\ A}{Time\ period} \tag{5.9}$$

    The rate referred from a particular site is relatively fixed based on the facts that every site has its own visitor base. If we see a dramatic increase in referrer rate, the traffic is suspicious.

The total quality score is the weighted summation of all the scores for factors and it is normalized to range [0, 1].

5.6 Disadvantages of the CCFDP Initial Version

Despite the better performance compare to other commercial solutions. CCFDP initial version also has considerable drawbacks.

(i) Similar to existing commercial solutions, CCFDP initial version was run offline. As the amount of pay-per-click traffic grows over the years the demand for an online/real solution was apparent.

(ii) CCFDP initial solution did not use baselines that analyze trends and outliers in the history of traffic. Fraudsters came up with low noise click fraud attempts which then required analyze of traffic in the history.

(iii) Software clicks were treated generally in the initial version but sophisticated click bots came into action, which required special solutions.

(iv) Initial version did not have a traffic visualization methods. For example "geographic distribution of traffic" etc. It also did not have a reporting tool that generates automatic reports for a click campaign.

(v) Since the CCFDP was not launched online client cannot register or manage their click campaigns online.

5.7 Can the model be improved with context data?

As new technologies evolve to combat click fraud, fraudsters find new ways to achieve their targets. By taking the advantage of advances in malware, they try to profit from click fraud, while making it harder to detect. As the botnets get more sophisticated, they are able to perpetrate more click fraud. They are discovering new ways to distribute and this can be seen in the data we have being collecting continuously.

The current situation requires the use of proactive technologies, which can detect unknown threats by examining their behavior. Click bot detection methods cannot just focus on aggressive patterns, such as in Bahama bot, but also need to examine the low rate patterns that are mixed with normal traffic. Also, attackers can constantly craft new attacks to make them appear different and legitimate; thus we cannot use the training-based approaches that derive patterns from historical attacks. Therefore, with the lack of ground truth, evaluating detection results is non trivial and requires different methodology and metrics than the detection methods.

We continue to craft more and more ideas to detect new types of click fraud. We develop data mining methodologies and tested them through the CCFDP system. Some of these solutions are inspired by the recent research conducted in the area of click fraud detection and prevention. These new ideas are reflected in the second phase of this research. The main goal of the second phase of research activities is to provide our initial CCFDP system with additional functionality.

The improved CCFDP, which we discuss in the next chapter, will include automatic scoring of click traffic with extended set of context-based parameters which are not included in our initial version. These parameters are derived attributes showing significant differences or outliers from baselines empirical distributions. The robust, online methodology for modeling and comparison of web click distributions will be developed. New extended descriptions of web clicks (context-based) will enhance the quality of a click score value by better describing characteristics of each click and intentions of the user. It will be calculated automatically in the CCFDP extended version using the technique known as incremental, semi-supervised support vector machines (SSVM).

Also, new algorithms are proposed and developed that improve the time and space performances because the CCFDP improved version should support online prevention capabilities. The suspicious clicks will be blocked from reaching the advertised site based only on server data and dynamically, online maintained variety of "black lists". This proactive emphasis on the prevention of fraud clicks will reduce significantly the number of highly suspicious clicks, and directly produce financial benefits for advertisers. We believe that prevention mechanisms as an addition to existing but improved detection mechanisms are key to success of the CCFDP system.

The new version will efficiently and seamlessly meet the demanding requirements of a growing Internet advertising business by qualifying users clicks on advertising web sites, and discovering and documenting click fraud events in real time. We will show that using our CCFDP (Collaborative Click Fraud Detection and Prevention) model will give advertisers, publishers, and advertisement networks powerful analytical tools to improve the performance and quality of their revenue generating models, increase the satisfaction level of their customers and affiliates, and most importantly, raise the individual consumers level of trust and confidence in the integrity of conducting business and purchasing transactions online.

CHAPTER 6

CLICK FRAUD DETECTION WITH EXTENDED CONTEXT DATA

Most of Current research and industrial solutions for click fraud detection use only either client side or server side data. In the previous chapter we have shown that use of both client side and server side data enhance the click fraud detection probability. In this chapter we further consider the ways to extend the data about each click. Two main goals in this chapter are:

1. Formalizing new approaches in click fraud analysis by defining new concepts such as:
    - click context, click baselines and fusion of click data.
    - applying some theories to formalize these concepts and build new architecture to detect previously undetected activities.
2. Based on introduced new theoretical concepts and appropriate formalisms we extended and enhanced our CCFDP architecture. We proved using real world data set that click fraud detection with extended click context improves the detection capabilities.

6.1 Context of the click

We assume that use of more context data about the click may help to better estimate its quality. Among the data we collect is spatial in nature while some is temporal in nature. Therefore, we define the context of the click record considering both spatial and temporal properties of the click.

Spacial data is also known as geo-spatial data or geographic information. It is the data that identifies the geographic location of features and boundaries. For example Figured 6.1 shows the geographical location of two cities(Louisville and Frankfort) in the state of Kentucky. IP addresses are usually assigned based on the geographical location. Therefore, in Figure 6.1 the IP can be a representative of the city. It is also possible to use Referrer and Country as spatial information providers in the click context.

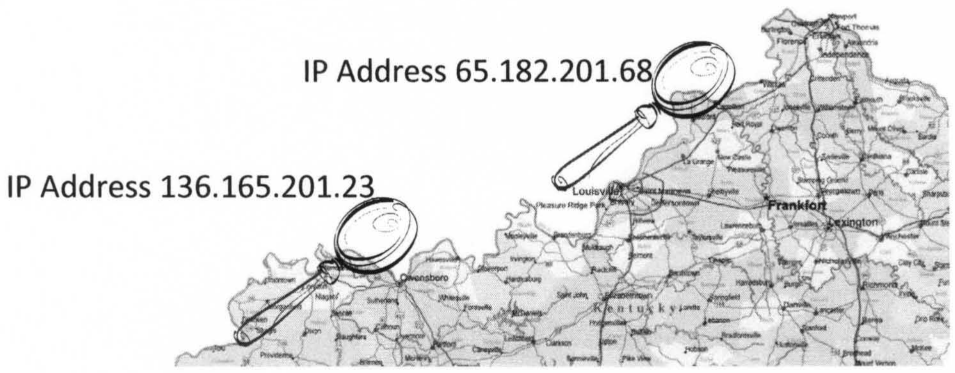
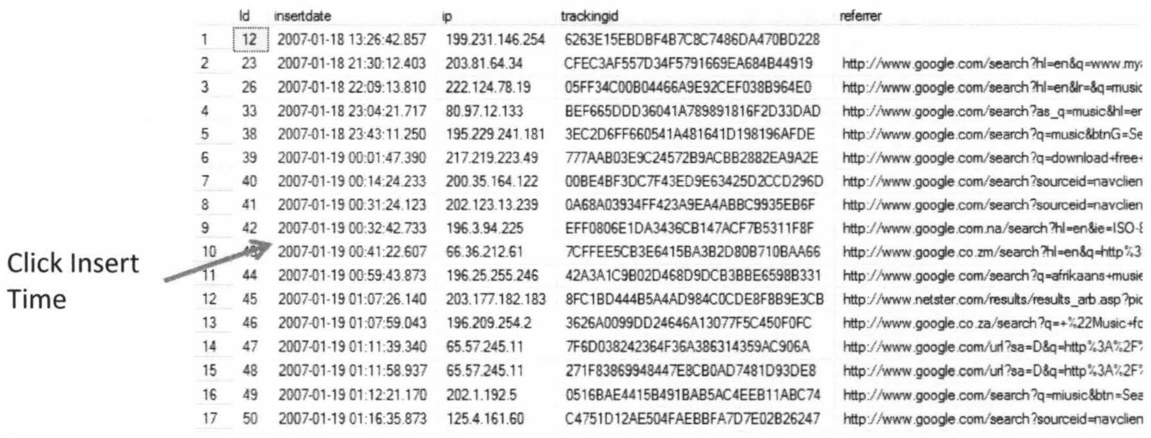


Figure 6.1: Relationship between IP address and location

On the other hand, temporal domain data encodes time aspects. More specifically the temporal aspects of data is usually associated with a valid-time. Valid time denotes the time period during which a fact is true with respect to the real world. For example "click insert time", which is the time a click is recorded in the database, provides temporal information (See Figure 6.2). A double click which measures with respect to IP or "tracking id" can also be considered as temporal information.

When a click is considered separately, its spatial and temporal data may not provide much useful information. But when the click is considered with other clicks in the neighborhood it can provide much meaningful information. For example, the concept of "double click" is not defined for a single click. Therefore, particularly for the click fraud detection, "context of the click" means the spatial and temporal characteristics of clicks that occur before or after the current click which is being observed. If we are considering the current click as the most recent click (Figure 6.3 left) the context of the current click includes all the clicks occurred before that. We will stick to this definition in all the



	Id	insertdate	ip	trackingid	referrer
1	12	2007-01-18 13:26:42.857	199.231.146.254	6263E15EBDBF4B7C8C7486DA470BD228	
2	23	2007-01-18 21:30:12.403	203.81.64.34	CFEC3AF557D34F5791669EA684B44919	http://www.google.com/search?hl=en&q=www.my
3	26	2007-01-18 22:09:13.810	222.124.78.19	05FF34C00B04466A9E92CEF038B964E0	http://www.google.com/search?hl=en&lr=&q=music
4	33	2007-01-18 23:04:21.717	80.97.12.133	BEF665DDD36041A789891816F2D33DAD	http://www.google.com/search?as_q=music&hl=er
5	38	2007-01-18 23:43:11.250	195.229.241.181	3EC2D6FF660541A481641D198196AFDE	http://www.google.com/search?q=music&btnG=Se
6	39	2007-01-19 00:01:47.390	217.219.223.49	777AAB03E9C24572B9ACBB2882EA9A2E	http://www.google.com/search?q=download+free+
7	40	2007-01-19 00:14:24.233	200.35.164.122	00BE4BF3DC7F43ED9E63425D2CCD296D	http://www.google.com/search?sourceid=navclien
8	41	2007-01-19 00:31:24.123	202.123.13.239	0A68A03934FF423A9EA4ABBC9935EB6F	http://www.google.com/search?sourceid=navclien
9	42	2007-01-19 00:32:42.733	196.3.94.225	EFF0806E1DA3436CB147ACF7B5311F8F	http://www.google.com.na/search?hl=en&ie=ISO-8
10	43	2007-01-19 00:41:22.607	66.36.212.61	7CFFEE5CB3E6415BA3B2D80B710BAA66	http://www.google.co.zm/search?hl=en&q=http%3
11	44	2007-01-19 00:59:43.873	196.25.255.246	42A3A1C9B02D468D9DCB3BBE6598B331	http://www.google.com/search?q=afrikaans+musie
12	45	2007-01-19 01:07:26.140	203.177.182.183	8FC1BD444B5A4AD984C0CDE8F8B9E3CB	http://www.netster.com/results/results_arb.asp?pic
13	46	2007-01-19 01:07:59.043	196.209.254.2	3626A0099DD24646A13077F5C450F0FC	http://www.google.co.za/search?q=+%22Music+fc
14	47	2007-01-19 01:11:39.340	65.57.245.11	7F6D038242364F36A386314359AC906A	http://www.google.com/url?sa=D&q=http%3A%2F%
15	48	2007-01-19 01:11:58.937	65.57.245.11	271F83869948447E8CB0AD7481D93DE8	http://www.google.com/url?sa=D&q=http%3A%2F%
16	49	2007-01-19 01:12:21.170	202.1.192.5	0516BAE4415B491BAB5AC4EEB11ABC74	http://www.google.com/search?q=miusic&btn=Sea
17	50	2007-01-19 01:16:35.873	125.4.161.60	C4751D12AE504FAEBBFA7D7E02B26247	http://www.google.com/search?sourceid=navclien

Figure 6.2: Records in a database

algorithms that run in realtime. But for offline processes, the current click can be any click in the click stream (Figure 6.3 right), the context of the current click will be clicks occurred before and after it.

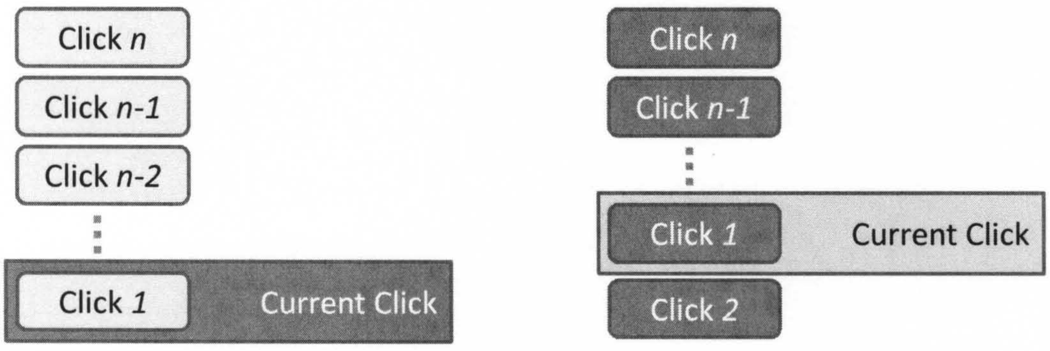


Figure 6.3: Context of the click

6.1.1 Local and Global context of a click

"The easiest way to go undetected while committing click fraud is to execute the actions in long time intervals. For example, clicking an ad once a day. Even if it is clicked once, the associated parameters such as IP, referrer and country should be changed if possible." This is advice given in a website that promotes click fraud. The vital information in this message is that click fraud may be committed within a short time interval (such as 10 clicks within a minute) or within a long interval as described above. Therefore, it is important to include mechanisms to detect both of these types of attacks.

We define local and global context of the click using the concepts of temporal and spatial data mining. Temporal data mining deals with the harvesting of useful information from temporal data, while spatial data mining is the process of discovering interesting and previously unknown, but potentially useful patterns, from large spatial datasets. Extracting interesting and useful patterns from spatial datasets is more difficult than extracting the corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types.

Local/Short term context

For data mining algorithms that work online, we define a short context. A short context can be the analysis of behavior of the 10 most recent clicks received by the system. For example if we receive 20 clicks from the same IP continuously, we can detect it by observing only the most recent clicks.

Global context

Global context of the click is defined as analyzing behavior of long term data such as clicks in the past 24 hours. Offline modules in our system use long context of data because the time restriction is minimal. For example a traffic from a certain country (India) may be significant today compare to yesterday or last week. We cannot detect this kind of variation using local context. It has to be analyzed considering all the traffic as a batch.

6.2 Mechanisms developed to detect click fraud in local context

Performance of any online solution largely depends on the amount of data it analyze or process in a given time. The amount of data that can be processed depends on how fast the results are required. For example, if the results should be available within a second, the amount of data that can be processed is lower than if the results are required within five seconds. Also, in such applications time consuming databases accesses should be eliminated or at least minimized. Therefore, complex algorithms are not appropriate for online solutions. Usually we have to depend on simple algorithms in the form of if-then-else rules.

The local context in CCFDP strict its neighborhood to most recent 10 clicks. Behavior of the clicks in this local context is done using a rule based system. Rules are designed utilizing standard industry heuristics for detecting click fraud and new concepts that we have developed. Each rule is triggered whenever a new anomaly record matches patterns given in the rule. A suspicious score is assigned for each such rule, based on the predefined threshold values. Details of the rule based system follows next.

6.2.1 Fraudulent traffic scoring using improved rule based module

In the CCFDP improved version we incorporate an improved rule base that detects suspicious clicks. Scoring the level of fraudulent activities in the given click requires:

(i) Partially scoring some characteristics of the fused click record

(ii) Combining these partial scores into final integrated score S

The current rule base consists of over 25 rules. Score values are normalized on [0, 1] interval, where $S = 0$ represents valid clicks, $S = 1$ is a fraud click, and $0 < S < 1$ values are interpreted as suspicious clicks. Sum of all scores for the clicks in the given campaign, normalized by the total number of clicks, represents a percentage of invalid click traffic. When the new clicks are coming, the scoring process includes:

(i) Collecting the values for partial scores r_i from GFD database for IP, UserAgent and Referrer (at the beginning they are not in database, so r_i values are 0)

(ii) Computing score values r_i for all other parameters (using corresponding heuristic "rules")

(iii) Combining all score values for the final score S for the given click. If partial scores are r_1, r_2,... r_n for a given click, then the final integrated score is expressed as

$$Score\ S = \frac{w_i r_i}{w_i r_i + (1 - w_i r_i)} \tag{6.1}$$

(iv) If click score in (*iii*) is $S = 0$, do not include any new records in GFD database (for IPs, UserAgents or Referrers)

(v) If click score in (3) is $S > 0$, and there is no previous records for key click attributes (IP, UserAgent or Referrer), create these new records with corresponding S values (and increase the count of the number of clicks for a given attribute)

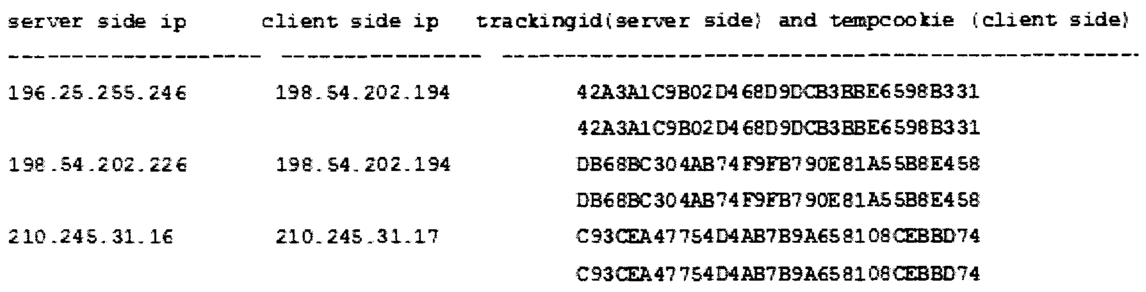
(vi) If some attributes of the click with $S > 0$ already exist in the GFD database, then adjust their historic scores by weighted averaging S for each attribute separately (IP, UserAgent, Referrer, etc.)

(vii) Based on values in fraudulent database GFD for each key attribute, dynamically are modified "blocked lists" in the blocking database BD. Specific predictor is going in the "blocked list" if its score value in GFD database is above the given threshold T (not necessarily $T = 1$, the threshold may be $T = 0.9$)

In the following section we explain in detail the rules that are used in the CCFDP system. These rules are designed utilizing standard industry heuristics for detecting click fraud and new concepts that we have developed. All the rules follow a simple IF-THEN-ELSE structure.

1. Server side IP and Client side IP mismatch

   The click fraud detection system assigns a unique identifier for each session between the web user and the web server. This identifier is known as the tracking id and it is saved in a form of a cookie. NetMosaics combines the client side record and server side record based on this tracking id value which should be the same for each session. In some situations, NetMosaics has found the server side and client side IPs do not match, indicating software-driven click fraud. An example is shown in Figure 6.4.



```
server side ip      client side ip   trackingid(server side) and tempcookie (client side)
------------------- ---------------- -------------------------------------------------
196.25.255.246      198.54.202.194         42A3A1C9B02D468D9DCB3BBE6598B331
                                           42A3A1C9B02D468D9DCB3BBE6598B331
198.54.202.226      198.54.202.194         DB68BC304AB74F9FB790E81A55B8E458
                                           DB68BC304AB74F9FB790E81A55B8E458
210.245.31.16       210.245.31.17          C93CEA47754D4AB7B9A658108CEBBD74
                                           C93CEA47754D4AB7B9A658108CEBBD74
```

Figure 6.4: Server side IP and Client side IP mismatch

The rule for this detection compares the server and client IP addresses to ensure that they are the same. The rule also checks that the client IP is not null.

```
Fraud score:
IF the server side IP and client side IP match        = 0
IF the server side IP and client side IP do not match = 1
```

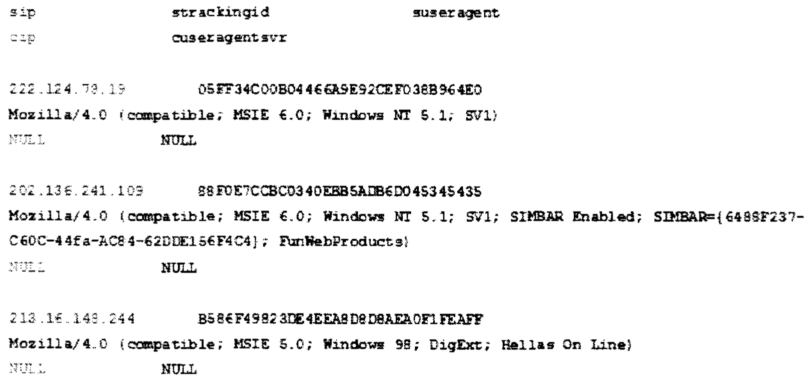
2. Empty userAgent

A user agent is a text sequence that a browser reports to a web site containing information such as the brand, version, plug-ins, and toolbars installed on a browser. Usually it looks like "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)." The basic information is this browser is Microsoft Internet Explorer version 6.0 and it is running on a Windows NT 5.1 platform. This information helps web sites to determine what capabilities a browser has, and helps the web site provide pages that cater to the browser. In addition to normal web users, robots, spiders, and bots that browse the Internet also report a user agent to web sites. Bots are automated software that browse web sites looking for information. Popular bots include Google bot, Bing bot, and Yahoo! Slurp. These bots crawl the Internet to find about content to include in their search engines. These user agents are stored and updated in the CCFDP. Fraudulent bots usually do not report their user agent. Therefore, the algorithm takes an empty user agent into consideration, making a null user agent score higher.

```
partial Fraud Score:
IF user agent is not empty = 0
IF user agent is empty = 1
```

3. Server side entries with no client side entries (Software click)

   In our system a software click is defined as a click with server side entries without corresponding client side entries. This is similar to what is shown in Figure 6.5. In this figure, SIP, strackingid, and suseragent are server side parameters and CIP, cuseragent and clientside parameters that do not exist.



```
sip             strackingid                suseragent
cip             cuseragentsvr

222.124.73.19       05FF34C00B0446A9E92CEF038B964E0
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
NULL            NULL

202.136.241.109     88F0E7CCBC0340EBB5ADB6D045345435
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; SIMBAR Enabled; SIMBAR={6488F237-
C60C-44fa-AC84-62DDE156F4C4}; FunWebProducts)
NULL            NULL

213.16.148.244      B586F49823DE4EEA8D8D8AEA0F1FEAFF
Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt; Hellas On Line)
NULL            NULL
```

Figure 6.5: Software Click

If the user is a human, for each request there will be a server side log and a client side log entry. However, if its a software, there will be no corresponding client side entries. There is one exception for search engine crawlers. Search engines use software bots to index their web pages; known as search engine crawlers that perform software clicks. In our research we have filtered out the search engine crawlers before processing of click traffic.

```
Partial Fraud Score:
Server side and matching client side entry exist = 0
Server side and matching client side entry do not exist = 1
```

4. No user activities

   A normal user will perform certain amount activities during the period that the web page is viewed. These activities could be mouse over some text, clicking a link or image, typing, scrolling up or down, etc. If there are no activities on the page, the click will likely be generated by a bot and it is click fraud. We consider the behavior of natural traffic without user activities to decide on a score for traffic. We allow the same percentage of traffic without user activities in natural traffic to exist in paid traffic.

   To calculate a score we have considered the average click fraud percentage found in pay per click traffic in search engines including Google AdSense and Yahoo Publisher Network was 28.1% in 3rd quarter of 2007. We take this value as the total fraudulent traffic to calculate other corresponding score values.

```
Partial Fraud Score:

Total fraudulent traffic = 28%

No activities in natural traffic = 4%

No activities in paid traffic = 30%

Score for paid traffic without user activities = 28%*(30% - 4%)

                                                = 0.07
```

5. Suspicious keywords in the user agent

   A user agent is a text sequence that a browser reports to a web site containing information such as the brand, version, plug-ins, and toolbars installed on a browser. Usually it looks like "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)." The basic information is: the browser is Microsoft Internet Explorer version 6.0 and it is running on a Windows NT 5.1 platform. This information helps web sites to determine what capabilities a browser has, and helps the web site provide pages that cater to the browser. In addition to normal web users, robots, spiders, and bots that browse the Internet also report a user agent to web sites. Bots are automated software that browse web sites looking for information. Popular bots include Google bot, Bing bot, and Yahoo! Slurp. These bots crawl the Internet to find about content to include in their search engines and usually they carry keywords such as "Google Bot," or "Yahoo! Slurp" in the user agent.

A lot of click fraud from adware or spyware uses suspicious keywords inside their user agent field. For example Figure 6.6 is extracted with the keyword "FunWebProducts" in the user agent field. After processing these records we have found that they are all fraudulent clicks.

```
213.55.89.84     Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1; SV1; FunWebProducts)
121.247.221.209  Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1; SV1; FunWebProducts; .NET CLR 1.1.4322)
125.99.244.44    Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.0; FunWebProducts)
24.132.32.13     Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1; ciIENL; FunWebProducts)
125.60.240.205   Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1; SV1; FunWebProducts)
```

Figure 6.6: Suspicious keywords in userAgent

In rare situations even normal traffic has such keywords. We have found out this rate in normal traffic and the same rate is allowed in the paid traffic.

To calculate a score we have considered the average click fraud percentage found in pay per click traffic in search engines including Google AdSense and Yahoo Publisher Network was 28.1% in 3rd quarter of 2007. We take this value as the total fraudulent traffic to calculate other corresponding score values.

```
Partial Fraud Score:

Total fraudulent traffic = 28%

Suspicious keywords in natural traffic = 0.94%

Suspicious keywords in paid traffic = 11%
```

```
Score for paid traffic with Suspicious keywords = 28%*(11% - 0.94%)
                                                = 0.028
```

6. Repeated visits

   Duplicate detection algorithm that we developed detect repeat visitors in two ways. The system implements a permanent cookie on the client computer for every new visitor. This permanent cookie is designed to use as a 128 bit globally unique identification number. Chances are highly unlikely that an assigned permanent cookies written to different client computers are the same. If the same user repeatedly visits the monitored web site, their permanent cookie will be recorded in the database. If the number of visits is beyond the allowed threshold value the click is suspicious. If the client browser does not accept cookies then its IP can be used as an alternative to detect multiple visits.

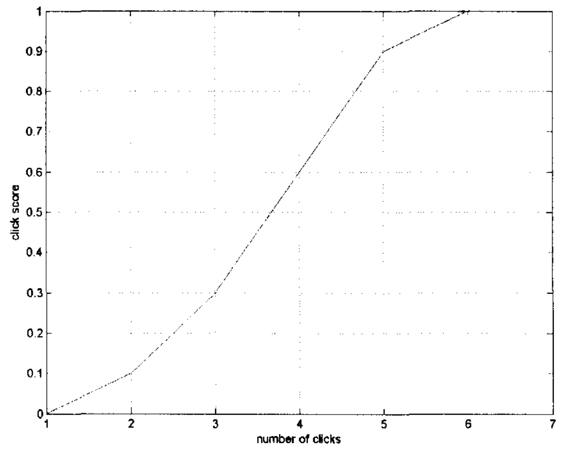


Figure 6.7: Repeated Visits scoring

At the beginning of the algorithm a table is pulled from the database that indexes the last 10 clicks to the system. If a click to a page and IP match within five seconds, they are added the match array. One match is a double click, two is a triple click, four matches is an invalid as are any matches after that. When four or more clicks occurs, the complete group of clicks is tagged as invalid. This means that the user has clicked in a session at least four times and is classified as suspicious activity. When five group Invalids take place, the IP is blacklisted. The score calculation is shown in Figure 6.7.

7. History count

   If the client computer permits Java script to be run in its web browser, our algorithm can find how many web pages the current browser visited before visiting the current website. Although Java script does not report the details of the visited web sites, this count is useful for detecting click fraud provided that the client computer uses Internet Explorer as the web browser. If Mozilla or FireFox is used, sometimes these browsers ignore the previous visits and set the history count to zero.

   If a site is referred by a search engine or other advertisement agent, the history count should not be 0. This is because the current browser must have the search engine or advertiser agent in its history. This conclusion is based on the fact that the advertisement link does not bring up a new pop up browser. The condition is easy to be satisfied for many search engines, such as Google because neither google.com nor its Adsense program use a pop up browser for their advertisements. Yahoo and the Yahoo partner network program do not use a pop up browser link either.

   We check to see if the user is using Internet Explorer using the browser field, then we also check that the history length field is not null. If there is a history length, we check that there was a referrer, to ensure that the history length is valid.

   In rare situations even normal traffic reports history count as 0, while using Internet Explorer and being referred by a website. We have found out this rate in normal traffic and the same rate is allowed in the paid traffic.

   To calculate a score we have considered the average click fraud percentage found in pay per click traffic in search engines including Google AdSense and Yahoo Publisher Network was 28.1% in 3^{rd} quarter of 2007. We take this value as the total fraudulent traffic to calculate other corresponding score values.

```
Partial Fraud Score:
Total fraudulent traffic = 28%
Zero history count in natural traffic = 0.074%
Zero history count in paid traffic = 7%
Score for zero history count = 28%*(7% - 0.074%) = 0.019
```

8. Cookie and Javascript are not allowed

   Through our research, we have found that 95% to 99% of normal users allow JavaScript and cookies in their web browsers. Most of the commercially available click fraud solutions use Java script and cookies to collect information about the client user. Traffic generated by fraudulent computers usually show as Java script and cookie disabled. We have seen in pay per click traffic the percentage that allow Java script and cookie is around 50%. Therefore, if the traffic does not allow cookies and Java script, the traffic is suspicious. Within the algorithm we check to see if the CCFDP previously set cookies are read and if the Java script generated text was recorded with the click.

   To calculate a score we have considered the average click fraud percentage found in pay per click traffic in search engines including Google AdSense and Yahoo Publisher Network was 28.1% in 3rd quarter of 2007. We take this value as the total fraudulent traffic to calculate other corresponding score values.

```
Partial Fraud Score:
Total fraudulent traffic = 28%
Percentage of natural traffic that does not allow cookie and
                                                Java script = 5%
Percentage of paid traffic that does not allow cookie and
                                                Java script = 14%
Score for cookie and Javascript not allowed =28%*(14%-5%) = 0.02
```

9. IP and permanent cookie inconsistency

   The cookie that our algorithm installs in a client computer in its first visit is a globally unique identification number for a client computer. This id is part of a the information that is sent to the web server by a client computer when a request is made. If the client user has not visited the web site previously the client's computer will be assigned a new cookie. Therefore, usually IP and permanent cookie for a client computer should be consistent. That means if a user revisits a web site, the IP address and its permanent cookie should be a fixed pair. If a cookie is deleted, the system will assign a new permanent cookie. This will create an inconsistency between the IP and permanent cookie. There is still another case for IP and permanent cookie inconsistency in which a web user connects to the monitored site with the same permanent cookie while their IP address is changing for every visit. Most likely the web user is changing a proxy server for different visits.

   In certain situations even normal traffic reports inconsistencies between cookie and IP. We have found out this rate in normal traffic and the same rate is allowed in the paid traffic.

   To calculate a score we have considered the average click fraud percentage found in pay per click traffic in search engines including Google AdSense and Yahoo Publisher Network was 28.1% in 3rd quarter of 2007. We take this value as the total fraudulent traffic to calculate other corresponding score values.

```
Partial Fraud Score:
Total fraudulent traffic = 28%
Percentage of natural traffic with IP and cookie inconsistency = 10%
Percentage of paid traffic with IP and cookie inconsistency = 40%
Score for IP and cookie inconsistency =28%*(40%-10%) = 0.08
```

10. Page activity analysis

   User or page activities that we track include mouse over text, clicks on text or image, use of browser scroll bar, and use of keyboard. Existence of at least once of these activities gives positives evidence towards a genuine click. But in some situations we have observed user activities are not reported in actual user sessions. This percentage is about 4.5%. Therefore, in paid traffic page activity analysis is considered suspicious only when its rate increases beyond 4.5%.

To calculate a score we have considered the average click fraud percentage found in pay per click traffic in search engines including Google AdSense and Yahoo Publisher Network was 28.1% in 3rd quarter of 2007. We take this value as the total fraudulent traffic to calculate other corresponding score values.

```
Partial Fraud Score:
Total fraudulent traffic = 28%
Percentage of natural traffic without user activity = 4.5%
Percentage of paid traffic without user activity = 43%
Score of percentage of users without user activity = 28%(43%-4.5%)
                                                    =0.108
```

11. IP time zone and Javascript time zone mismatch

    In our data there are three location related parameters. IP, web users local time, and time zone. The location indicated by those three parameters should be consistent by any web visitor. Our location verification algorithms check that the timezone and click time are valid.

    In rare situations even normal traffic reports inconsistencies between IP and actual time zone, which is about 3.09%. We have excluded this percentage in the paid traffic.

    To calculate a score we have considered the average click fraud percentage found in pay per click traffic in search engines including Google AdSense and Yahoo Publisher Network was 28.1% in 3rd quarter of 2007. We take this value as the total fraudulent traffic to calculate other corresponding score values.

```
Partial Fraud Score:
Total fraudulent traffic = 28%
Percentage of natural traffic with location mismatch = 3%
Percentage of paid traffic with location mismatch = 5\%
Score for IP time zone and Javascript time zone mismatch
                                                  = 28%*(5\%-3\%) = 0.006
```

12. Traffic is too dense

    If the administrator is familiar with the traffic flow to a web server, for example 100 requests per minute from a particular referrer, and all of a sudden if the server gets 100 requests within 10 seconds it is suspicious. Based on the variable think time for humans, their click flaw does not follow a pattern and there is a maximum number of clicks per given time period that they can make. For example there is no reason for a human to click 20 times on the same link within 10 second time interval. If the click count is high for a given time period it may be due to robotic activity that generates artificial clicks.

```
Partial Fraud score:
IF (page view time) is between 0 and 1 seconds
        IF (click count) $>$ 10)
                Score for dense traffic = 0.7
```

13. Mouse click location analysis

    We collect information about the location of the user clicked on the page, through a Java script. This click location information tells where user clicked in the browser. There are two ways of displaying advertisements in a web browser, either as a text link or as an image. If this click landed on either of these the click will generate a request for the target web page. But if it generates a request with a click without a mouse move, the click must have been originated by a click bot. Because, a click bot does not necessarily have to click on an advertisement to request a web page.

Our algorithm checks the running click count and mouse movement count to see if the mouse has moved and if this is the first click. A higher score is given when no movement is detected.

```
Partial Fraud score:
IF (mouse move count)$<$0
        IF (click count)$>$0
                Score for mouse click location analysis = 0.007
```

14. Blacklists

Blacklists are lists of IP addresses, domain names, email addresses or content of the headers or the body, or some combination of these different types, that can be used to help identify fraudulent traffic. A special subset of IP address and domain name lists exist which can be queried using DNS, which are called DNS Blackhole.

If we identify that the traffic is referred from certain source that has a high percentage of click fraud in the history this referrer will be added to the blacklist. For example according to the data in fraudulent database, we can easily identify that the traffic from some referrer sites has very low mouse activities and high software click rate. Those sites are added to blacklist referrer, and this list is maintained dynamically where IPs are added and removed considering the baselines of traffic they generate.

```
Partial Fraud score:
Score for blacklisted IP/referrer/country/etc. = 1
```

6.3 Mechanisms developed to detect click fraud in global context

Global context of click traffic is defined using behavior of the data in the past. Specifically we define global context for each click parameter such as IP, referrer, country, ISP, etc. We compile counts or histograms of these attribute-value distributions over time to determine "normal" activity for a particular attribute-value pair. We then compare the aggregation of previous data with an aggregation of a current data. Observed behavior is flagged as a potential fraud if it deviates significantly from expected behavior, which we referrer here onwards as "outliers".

6.3.1 Dynamic baselines

A number of definitions have been given to the term "outlier" depending upon the task at hand. For example Hawkins [Hawkins, 1980] defines outliers by the following: "an outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism." This description points out two important points for our task. First, that an outlier should be considered suspicious. The deviation from "normal" data gives rise to suspicion, suspicion that some unusual mechanism has generated this data point. Second, what is considered normal must be modeled in such a way that points that deviate from what is considered normal can be detected and separated from the rest of the data. This section will describe how we model normal data, or in our case normal click traffic.

Each click record is made up of a number of attributes such as browser, operating system, IP, referrer, etc. We model each attribute separately. We will determine normal behavior for referrer separate from all other attributes and browser separate from other attributes and so on. For each attribute we create histograms which maintain counts for the most common values in each attribute. These histograms are called baselines. An example of such a histogram can be seen in Figure 6.8 (left). In Figure 6.8 (left) one can see the number of clicks for each of four values of the referrer attribute. These histograms are then used to calculate the percentages for each value.

In addition to absolute counts being maintained in the baseline, we also measure variance. Over time certain values were found to vary greatly in the percentage of traffic with a given value. For example traffic with Google as the referrer varied through out

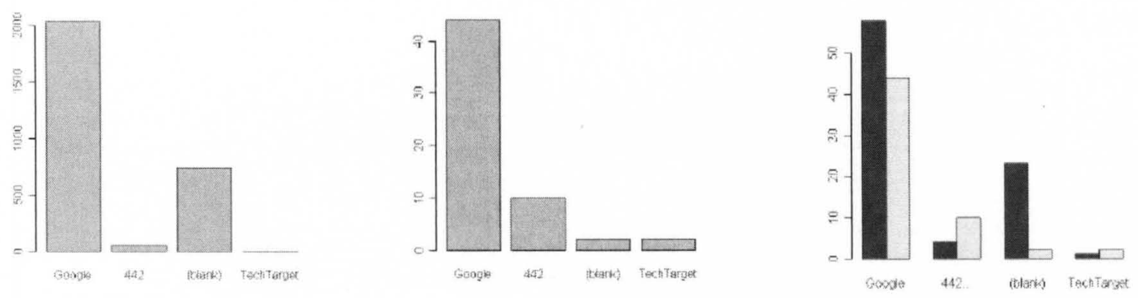


Figure 6.8: (left) Global baseline for four referrers as of 1/22/08. (center) Aggregated window for 1/22/08. (right) Counts and thresholds for referrer on 1/22/08. Thresholds are dark gray and counts in light gray.

time from nearly 100% to less than 50% of traffic in a given day. This variance made comparisons with the baseline difficult. Variance is calculated using the formula to follow.

$$\sigma = \sum_{i=1}^{n} p_i \left(\frac{x_i}{c_i} - \mu \right)^2 \tag{6.2}$$

This formulation of variance takes into consideration time periods varying in number of clicks. In the above formula each time period within the baseline is given by i. The x_i is the count within that time period for the attribute-value pair. The c_i is the total number of clicks within the time period. The value μ is the percentage of clicks for the current attribute-value pair throughout the entire baseline. Lastly, p_i gives the percentage of clicks in the baseline from the current time period.

Our baselines are used to identify outliers in incoming traffic among the attributes. Since we are identifying outliers based on context, we must necessarily accumulate clicks for a comparison of short term context with past context. These short term accumulations of clicks are called aggregated windows. An aggregated window is also treated as a histogram. An example of the comparison of an aggregated window to a baseline can be seen in Figure 6.8 (left and center).

We assume that percentages for a given attribute-value pair are distributed normally through time. An outlier is detected when a count for an attribute-value pair in an aggregated window is found in the upper 5% tail of the normal distribution, or is 1.645

standard deviations above the mean as shown in Figure 6.9. We calculate the threshold in terms of number of clicks using the following formula to ensure whole number thresholds:

$$threshold = [(\mu + 1.645 * \sigma) * windowsize] \tag{6.3}$$

where μ is the percentage of the baseline made up by the attribute-value pair and σ is the standard deviation of that percentage in the baseline. Additionally, *windowsize* refers to the number of clicks in the current aggregated window. If the number of clicks in the aggregated window for a particular attribute-value pair is greater than the threshold, then it is considered an outlier. An example of calculated thresholds can be seen in Figure 6.8 (right).

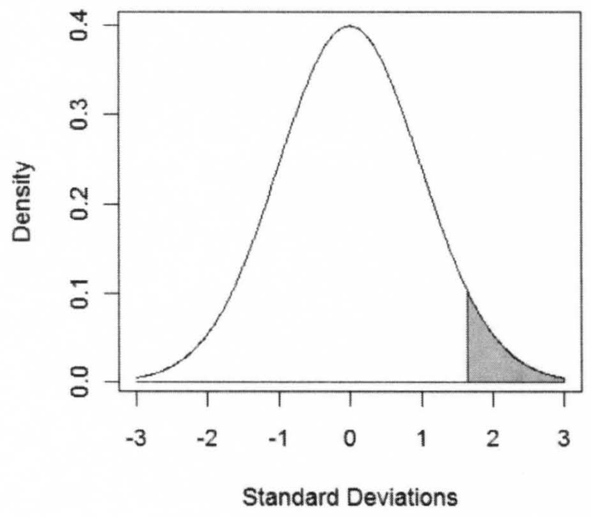


Figure 6.9: Gaussian (Normal) distribution with the top 5% highlighted

Outliers are detected for each attribute in a given aggregated window. The results are then applied to the following context fields for each record: referrer, browser, operating system, country, ISP, and IP. If a record has an outlying referrer for the current aggregated window, it is reflected in the referrer context field. An example of an extended record can be seen in Table 6.1. Baselines are then recalculated adding in the current aggregated window to keep baselines up to date and for comparison to future aggregated windows.

Table 6.1: Example record after outlier detection preprocessing. Record now contains server side data, client side data and context based on a number of attributes

Click data			Context fields										
Server side			Client side					Referrer	Browser	OS	Country	ISP	IP
2	3	4	0	1	1	2	1	0.18	0.12	0.03	0.13	0.02	0

6.3.2 Fraudulent Traffic Scoring

The CCFDP system scores each incoming click. The score will be a value in the range [0,1], where a zero represents no evidence of fraudulent behavior, and a one represents 100% confidence in the click being fraudulent. Within the CCFDP system, the outlier detection module provides relevant context to each attribute for scoring each click. Each click is provided a partial score for each attribute based on the variation of that click within the current context from normal behavior. Zero signifies that no evidence was found of suspicious activity for the given attribute-value pair in the current context.

We will discuss two approaches for scoring an attribute-value pair when the count exceeds the calculated threshold. The first approach to scoring attach a context score to a click record which conveys the extent to which a count has exceeded its threshold or the degree of suspicion for that particular count. Take for example referrer X. In the case where referrer X's calculated threshold is 2, it would seem logical to give a higher partial context score if the actual count were 38 compared to 3. A different partial score is given to counts that far exceed their threshold, compared to counts that barely exceed their threshold. This provides more information to the overall scoring algorithm. This could allow for more refined overall scoring by providing a range of values for suspicious activity.

This approach to scoring uses the difference between the attribute-value pair count in an aggregated window and the corresponding threshold. Then take that as a percentage of the total number of clicks in the aggregated window. A score in this first approach is given by the following formula:

$$\text{partial context score } A_i = \frac{count_i - threshold_i}{windowsize} \tag{6.4}$$

This type of scoring will be referred to as "variable scoring" by later sections. The closer to one the score gets, the more certainty given by the outlier detection system that something suspicious is happening. When the score is close to zero, then little evidence for suspicion is available. The second approach is to simply give a constant score to all attribute-value pairs exceeding their thresholds. Click fraud is perpetrated in a large number of ways, some approaches [Daswani and Stoppelman, 2007] expect that a large number of clicks over a short period of time raises suspicions, and attempt a "low-noise click fraud attack".

6.4 Fusion of Data

We initiated this research with an assumption that more data about each click collected from different sources will result in better estimation of the click quality.

Our data sources can be classified into three categories:

(i) Direct sources which include server computer, client computer with clicktracking information.

(ii) Indirect sources real time buffer and fraudulent database GFD (data are generated in extended real time).

(iii) History data baselines and Blocking Database (BD).

Data belonging to direct sources are collected in real-time and stored as a record of the click without any preprocessing. Indirect sources define some derived click attributes which require a significant amount of preprocessing. These transformations take place in a real-time for buffer data (recent server data about clicks), and require extended real-time for detection and recording outliers in the current time window compared with various baselines.

Data fusion from these sources includes the following activities:

(a) Server and client side data are fused based on identical tracking ID. If IDs are not the same, the equivalence of IPs is used as an alternative criterion for fusion of partial records.

(b) Fusion of clicktracking data with client data is much more complex because of their 1:n relation: for each record in client side there may be more several user activities recorded in clicktracking. Because we are performing data fusion in real time we don't have possibility to "wait" for all click tracking data, and we make "temporarily" integration based on pre-specified window of clciktracking records.

(c) Characteristics of the clicks are not only "static" based on server and client information. There are also "dynamic" data about each click describing the context which depends on clicks before and after the current one. For example, a single click for a given IP may not be suspicious, but hundred of consecutive clicks from the same IP will make this IP highly suspicious. Similar analysis can be performed with other parameters registered at server side such as referrer or country. We are using real time buffer (recent server click records) to detect these outliers online and to transform them into additional context based characteristics of clicks.

(d) Collection of data about clicks is extended real-time that gives information about clicks when comparing with standard baselines for key parameters.

(e) Each IP, referrer, etc. (key parameters) may have some history on the given site. These characteristics are included in the record about current click. For example, if the current click is based on referrer which has history of suspicious clicks, this fact will be included in the record and computation of a click score.

(f) Blocking database describes highly suspicious clicks from some IPs, referrers, or countries. Our implementation of blocking database allows online changes and therefore more efficient blocking process comparing with traditional commercial solutions.

Previous steps are only illustrative examples of a data fusion process, while the details are given in the CCFDP documentation[NetMosaics, 2009]. The integrated structure of a click record which includes all context information is shown in Figure 6.10.

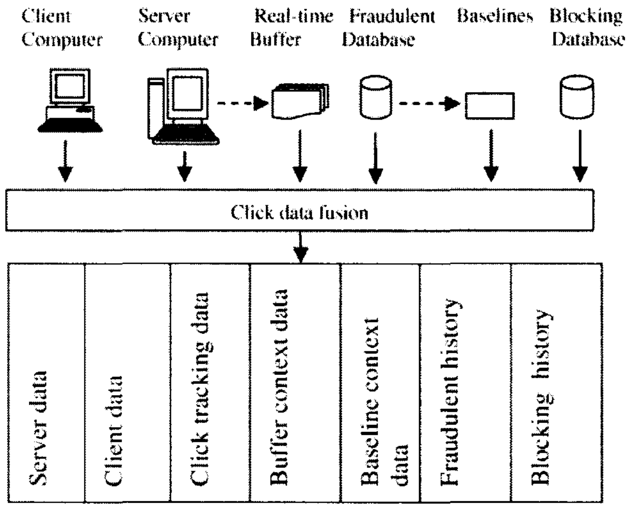


Figure 6.10: Integrated structure of the click

6.4.1 Why Data Fusion?

Data fusion is "a process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats, and their significance" [Lambert, 2009]. The resulting information is more satisfactory to the user when fusion is performed than simply delivering the raw data [Wald, 2001]. Waltz and Llinas have described important features related to the development of data fusion architecture essential also for any click fraud analysis system.

They include:

i. Robustness and reliability: The system is operational even if one or several sources are missing or malfunctioning.

ii. Extended coverage in space and time.

iii. Increased dimensionality of the data space: It increases the quality of the deduced information while reducing vulnerability of the system.

iv. Reduced ambiguity: More complete information provides better discrimination between available hypotheses.

v. Solution to information explosion [Waltz and Llinas, 1990].

Data fusion techniques are widely used for target identification and tracking, situation awareness [Fusheng and Feng, 2008], threat assessment [Janez et al., 2000], military [Tian et al., 2005] and public security applications [Zeng and Xu, 2008, Yukun et al., 2009]. In most of these applications the fusion model has been selected mainly considering the practical application since there is still not a universal fusion model available. In this paper, specific fusion process architecture has been introduced to our application based on the Joint Directors of Laboratories (JDL) model [DOD, 1991] and we differentiate between the following levels of abstraction:

i. Data/Observation level fusion: Measurements which can be univariate, multivariate, and/ or multidimensional, measurements may also exhibit temporal, spatial properties etc. are fused at this level.

ii. Variable level fusion: A variable is derived from data using a data analysis algorithm. Transformed domain variables are fused at this level.

iii. Decision Level: When the results of the high level fusion are available, variables can be interpreted for decision making. The final result is obtained at the decision module by fusing the local decisions of the system to get a more precise and comprehensive understanding to the system's situation.

Different fusion methods are used in different fusion levels, such as statistical estimation [Durrant-Whyte, 1987, Hager et al., 1993], Kalman filter [Yukun et al., 2007], fuzzy integration [Solaiman et al., 1999], neutral networks [Dai and Khorram, 1999], D-S evidence theory [Wu et al., 2002] and so on. Of these fusion methods, D-S evidence theory is widely known for better handling uncertainties. Moreover, it provides flexible information processing and can deal with asynchronous information [Ouyang et al., 2008].

6.5 Dempster-Shafer Evidence Theory

The Dempster-Shafer theory, also known as the theory of belief functions, is a generalization of the Bayesian theory of subjective probability. Whereas the Bayesian theory requires probabilities for each question of interest, belief functions allow us to base degrees of belief for one question on probabilities for a related question. These degrees of belief may or may not have the mathematical properties of probabilities; how much they differ from probabilities will depend on how closely the two questions are related.

The Dempster-Shafer theory owes its name to work by A. P. Dempster (1968) and Glenn Shafer (1976), but the kind of reasoning the theory uses can be found as far back as the seventeenth century. The theory came to the attention of AI researchers in the early 1980s, when they were trying to adapt probability theory to expert systems. Dempster-Shafer degrees of belief resemble the certainty factors in MYCIN, and this resemblance suggested that they might combine the rigor of probability theory with the flexibility of rule-based systems. Subsequent work has made clear that the management of uncertainty inherently requires more structure than is available in simple rule-based systems, but the Dempster-Shafer theory remains attractive because of its relative flexibility.

In the following section, terminology of theory of evidence [Shafer, 1976] and the notation used in this paper are defined.

i. *Frame of discernment* : If Θ denotes the set of θ_N ($\theta_N \in \Theta$) corresponding to N identifiable objects, let $\Theta = \theta_1, \theta_2, \ldots \theta_N$ be a frame of discernment. The power set of Θ is the set containing all 2^N possible subsets of Θ, represented by P(Θ):
$\mathrm{P}(\Theta)=\{\Phi,\{\theta_1\},\{\theta_2\},\ldots\{\theta_N\},\{\theta_1,\theta_2\},\{\theta_1,\theta_3\},\ldots\Theta\}$
where Φ denotes the null set.

ii. *Basic Probability Assignment function* (BPA) : The BPA is a primitive of evidence theory. The BPA, represented by m, defines a mapping of the power set to the interval between 0 and 1, where the BPA of the null set is 0 and the summation of the BPA's of all the subsets of the power set is 1. The value of the BPA for a given set A, represented as $m(A)$, expresses the proportion of all relevant and available evidence that supports the claim that a particular element of Θ belongs to the set A but to no particular subset of A. The elements of P(Θ) that have none-zero mass are called focal elements. Formally, this description of m can be represented with the following three equations:

$$m : P(\Theta) \Rightarrow [0,1] \tag{6.5}$$

$$\sum_{A \in P(\Theta)} m(A) = 1 \tag{6.6}$$

$$m(\Phi) = 0 \tag{6.7}$$

iii. *Belief function* Bel(A) : Given a BPA m, a belief function Bel is defined as:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{6.8}$$

The belief function $Bel(A)$ measures the total amount of probability that must be distributed among the elements of A.

iv. *Combination of rule of evidence* m(C) : Supposed m_1 and m_2 are two mass functions formed based on information obtained from two different information sources in the same frame of discernment; according to Dempster's orthogonal rule we define $m(C)$ $= (m_1 \oplus m_2)(C)$

if $(C{=}\Phi)$

$$(m_1 \oplus m_2)(C) = 0$$

else

$$(m_1 \oplus m_2)(C) = \frac{\sum_{A \bigcap B=C} m_1(A)m_2(B)}{1-K} \tag{6.9}$$

Where K represents basic probability mass associated with conflict defined as:

$$K = \sum_{A \bigcap B \neq \Phi} m_1(A)m_2(B) < 1 \tag{6.10}$$

In our system, evidence supports a click to either be valid or invalid. Therefore it becomes a two class problem. Accordingly we have modified the calculation of $m(C)$ for the CCFDP system [NetMosaics, 2009]. For a two class problem, we can simplify the equation for combination of evidence to:

$$S = \frac{\Pi_{i=1,n} r_i}{\Pi_{i=1,n} r_i + \Pi_{i=1,n}(1 - r_i)} \tag{6.11}$$

where r_i is the output from each model and n is the number of models.

6.6 Fusion of Evidences of Click Fraud in the CCFDP System

The collaborative click fraud detection and prevention (CCFDP) system was developed to collect data about each click, involving the data fusion between client side log and server side log [Ge and Kantardzic, 2006]. In CCFDP there are three modules that contribute to the process of finding fraudulent clicks. They are rule based module, click map module, and outlier detection module. In each of these modules, output is a probabilistic measure of evidence for the click being fraudulent. Authors have discussed the functionality of each of these modules in detail before [Kantardzic et al., 2008, Kantardzic et al., 2009]. In addition, CCFDP maintains an online fraudulent database of suspicious sources of clicks in terms of IP, referrer, country etc. When the score of an IP or a country etc. reaches a predefine threshold value the CCFDP system moves it to the online fraudulent database and inform the service providers with the instructions to block future traffic originating from these sources. Scores for each parameter are updated after a click found suspicious based on the combined evidences of the modules that we mentioned above. In the following section we differentiate between an event and an evidence.

6.6.1 Event vs. evidence

Incoming click is the event we consider in the CCFDP system. The evidence provides support information for this event's past and present activities. For example consider the following four pieces of evidence of CF.

i. Evidence A: IP associated with the click generates software clicks (Software clicks are generated by automatic agents such as click Bots).

ii. Evidence B: Search engine crawler is associated with some clicks.

iii. Evidence C: The referrer for these clicks is associated with fraudulent clicks detected in the past.

iv. Evidence D: Click does not have user activities.

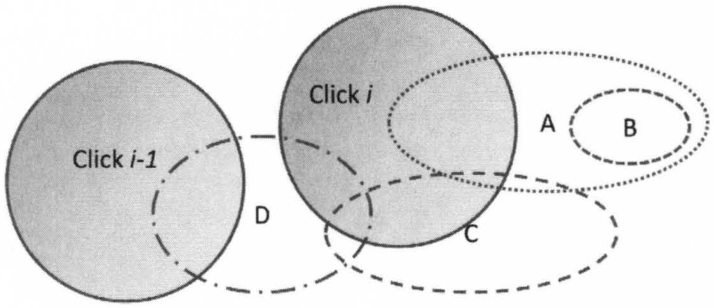


Figure 6.11: Event Vs. Evidence

A piece of evidence can be associated with multiple possible events (click i and $i-1$) unlike traditional probability theory where evidence is associated with only one event. For example the click event i in Figure 6.11 is associated with evidences A, C, and D. Click event $i-1$ is only associated with evidence D. The two events share event D. Evidences can overlap (A and C), one evidence can be a specific case of more general evidence (A and B) etc. In CCFDP we try to maximize the detection of evidence associated with an event. We achieve this by using extended context of the click, rule based module, baseline module and click map module.

i. Evidence in the rule based module

   In the rule based module evidence about CF is represented as rules. Each rule has a value between 0 and 1. This module extensively analyzes the context of the incoming click. Final score of this module is obtained by fusing these individual scores. Fusion is done in data level using the D-S evidence theory.

ii. Evidence in the outlier detection module

    The outlier detection module defines outliers based on baselines for each click parameter (IP, referrer, country etc.) considering the data collected in the past. We maintain two sets of outliers, Local and Global. Local outliers are based on one set of click parameters and Global outliers are defined using another set of click parameters. Baselines are compared with the variations in the current context of the click to detect suspicious patterns in the data. The variation (evidence) in each parameter is represented as a probabilistic score. A variable level fusion is performed to combine the individual pieces of evidence using D-S evidence theory.

iii. Evidence in the click map module

    The "click map" tracks the current activities being performed by the user while the user's session is active. "Click map" assigns a score for each click based on click's location relative to the positions of the advertisements in the webpage the user is viewing. In the current version click map module works as a classifier. It filters out clicks recorded away from the actual advertisement area.

6.6.2 Assumption of Click Orthogonality

The Dempster-Shafer theory requires evidence to be orthogonal to perform its sum of orthogonality. Therefore any method of proof should be first used to verify this assumption. In our system we will be using Pearson's correlation.

6.6.3 Model-driven fusion process

CCFDP combines the past and present evidence for each click to better understand its current and future behavior. If the click is found fraudulent (combined score greater than a threshold), the associated sources of the click will be moved to a suspicious database, which will be immediately used by modules discussed above, to score the next new incoming click.

The model-driven fusion process of CCFDP is depicted in Figure 6.12. Real-time data

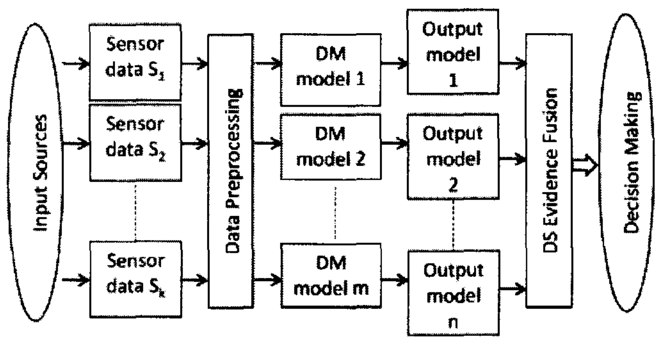


Figure 6.12: Model-driven fusion process of CCFDP

feeds from three sources *(k=3)*: server side, client side, and extended context of the click (S_1,S_2,S_3). This is represented by sensors in Figure 6.12. In the data preprocessing stage we standardize (align) the input data [Waltz, 1998]. The concept of alignment is an integral part of the fusion process, and assumes *"common language"* between the inputs and includes the standardization of measurement units. The scores from the rules based module (DM model 1), outlier detection module (DM model 2), and click map module (DM model 3) are then combined using D-S evidence theory at the decision level *(m=3)*. The combination of scores will be used to dynamically adjust advertising profiles in such a way that low quality sources of traffic will no longer be shown advertisements.

6.6.4 A case study

In this section, we demonstrate the application of D-S evidence theory to combine evidences of sources.

Evidence 1: Repeated clicks from IP during past minute detected by the rule based module.

Evidence 2: Java Script is allowed in the browser detected by the rule based module.

Evidence 3: Country Morocco is detected suspicious by outlier module.

Our task is to use these evidences to show a click to either be fraud or non-fraud. Therefore it becomes a two class problem. Fraud is represented by F, and non-Fraud is represented by N.

Let $\Theta = \{F, N\}$, We define the power set P(Θ)=$\{\Phi,\{F\},\{N\}\}$. Assuming local suspicious scores based on evidences, we define:

$m_1(\Phi)$=0, $m_1(\{\mathrm{F}\})$=0.6, $m_1\{\mathrm{N}\}$=0.4

$m_2(\Phi)$=0, $m_2(\{\mathrm{F}\})$=0.5, $m_2\{\mathrm{N}\}$=0.5

$m_3(\Phi)$=0, $m_3(\{\mathrm{F}\})$=0.7, $m_3\{\mathrm{N}\}$=0.3

Calculation of $M_1 \oplus M_2$

For the convenience we use the fusion tables, introduced by Shafer [Shafer, 1976], to show the calculations. Fusion tables are given in Table 6.2 and Table 6.3.

Table 6.2: Fusion of Evidence 1 and Evidence 2

$(M_1 \oplus M_2)$	{F}0.5	{N}0.5
{F}0.6	{F}0.3	Φ0.3
{N}0.4	Φ0.2	{N}0.2

Using equation 6: K=0.5*0.6 + 0.4*0.5 = 0.5

$m(\{F\}) = \frac{m_1\{F\}*m_2\{F\}}{1-K} = \frac{0.3}{0.5} = 0.6$

New belief function

$Bel_{m_1 \oplus m_2}(\{F\}) = \sum_{B \subseteq \{F\}} m(B)$=0 + 0.6 = 0.6

Calculation of $M_1 \oplus M_2 \oplus M_3$ using equation 6:

Table 6.3: Fusion of Evidences 1,2 and Evidence 3

$(M_1 \oplus M_2 \oplus M_3)$	{F}0.7	{N}0.3
{F}0.3	{F}0.21	Φ0.09
Φ0.3	Φ0.21	Φ0.09
Φ0.2	Φ0.14	Φ0.06
{N}0.2	Φ0.14	{N}0.06

K=0.09 + 0.21 + 0.09 + 0.14 + 0.06 + 0.14 = 0.78

$m(\{F\}) = \frac{m_1\{F\}*m_2\{F\}*m_3\{F\}}{1-K} = \frac{0.21}{0.27} = 0.78$

New belief function

$Bel_{m_1 \oplus m_2 \oplus m_3}(\{F\})=\sum_{B\subseteq\{F\}} m(B)=0 + 0.78 = 0.78$

In this example we considered the local suspicious scores of 0.6, 0.5, and 0.7. D-S evidence theory is used to find the final evidence. The belief value that the click is fraudulent is 0.78.

6.7 Experimental Results and Discussion

We developed the CCFDP improved version based on the its initial version developed by Dr. Li Ge [Ge and Kantardzic, 2006]. The real time version of CCFDP is now available online at http://www.netmosaics.com. All of our experiments use click data from Hosting.com and thebestmusicsites.org websites. The process was started on January 7th, 2007 and is still in collecting data. As of March 30th, 2011 we have collected around 1,400,000 natural and 25,000 paid click data.

Initial version of CCFDP was designed using only a rule based system. The new CCFDP has outlier module and the click map module in addition to an improved rule based system with additional click context information. Experiments are performed on both old and new versions of CCFDP. Experiments are performed under five categories. They are:

i. Verification of Click Orthogonality

ii. Comparison of results for change in score of IP, Referrer, Country etc. in two versions of CCFDP

iii. Comparison of distribution of final score

iv. Comparison of improvements in quality of traffic

v. Comparison of results with Google Adwords

6.7.1 Calculation of Click Orthogonality

We have calculated the Pearson Correlation coefficient between the results of the outlier module and the rule based module. It is 0.0071. Based on the results, safely assume the evidences are orthogonal and perform the Dempster-Shafer orthogonal summation.

6.7.2 Comparison of results for change in score of IP, Referrer, Country etc. in two versions of CCFDP

After all paid click data has been processed we have selected the top 10 IPs, countries, and referrers with the highest fraudulent scores to see if the fusion process has any effect on updating individual scores of these parameters. Tables 6.4, and 6.5 list the IPs, countries, and referrers that have the highest fraudulent scores respectively. The results are slightly modified to protect privacy of some publisher websites. For example the actual domain names and referrer names are replaced with dummy identifiers.

Table 6.4: Top IP and Country Counts

Top IP Count		Top Country Count	
IP	Count	Country	Count
71.235.26.170	122	US	19784
68.88.239.191	112	IN	1278
136.165.67.74	94	CA	856
199.231.146.254	86	GB	666
89.139.234.179	82	NULL	574
203.162.3.146	80	MX	544
170.20.96.116	80	AU	534
71.193.114.12	72	TR	518
74.133.47.66	68	BR	506
74.192.144.103	68	PH	456

In Figure 6.13 (left) the variation of scores for IPs are depicted. Except for one

Table 6.5: Top Referrer Counts

Referrer	Count
No referrer (NULL)	8800
http://www.r1.com/	4568
http://www.r2.com/	2192
http://www.r3.com/	604
http://www.r4.com/	546
http://www.r5.com/	538
http://www.r6.com/	510
http://www.r7.com/	450
http://www.r8.com/	420
http://www.r9.com/	414

IP address (136.165.67.74) all others have higher fraudulent scores after combining the evidences from all the modules. In the rule based system, evidence is collected by considering only the changes detected in a limited neighborhood [1]. For example with only the rule based system, it will be difficult to detect a Bot associated to a particular IP which sends http requests in the time intervals greater than 15 minutes. But with the outlier detection module that covers larger neighborhood[1] of the clicks, the pattern becomes observable. Once a suspicious activity is detected this evidence will contribute to increase of corresponding partial scores in the CCFDP system. IP address with higher scores have increased probability of being blacklisted sooner. Once the IP addresses are on the blacklist the search provider will be notified to eliminate future traffic from the corresponding sources. This will improve the quality of the traffic redirected to the advertiser's website.

One of the biggest advantages of using a multi-model system in CCFDP is its ability

[1]A neighborhood is a window of clicks. It can be defined as fix amount of most recent clicks or clicks arrived in a fix time interval. In the CCFDP system, rule based module uses a small (limited) neighborhood of 10 most recent clicks. The outlier detection module uses a large neighborhood which is all clicks received in the past 24 hours.

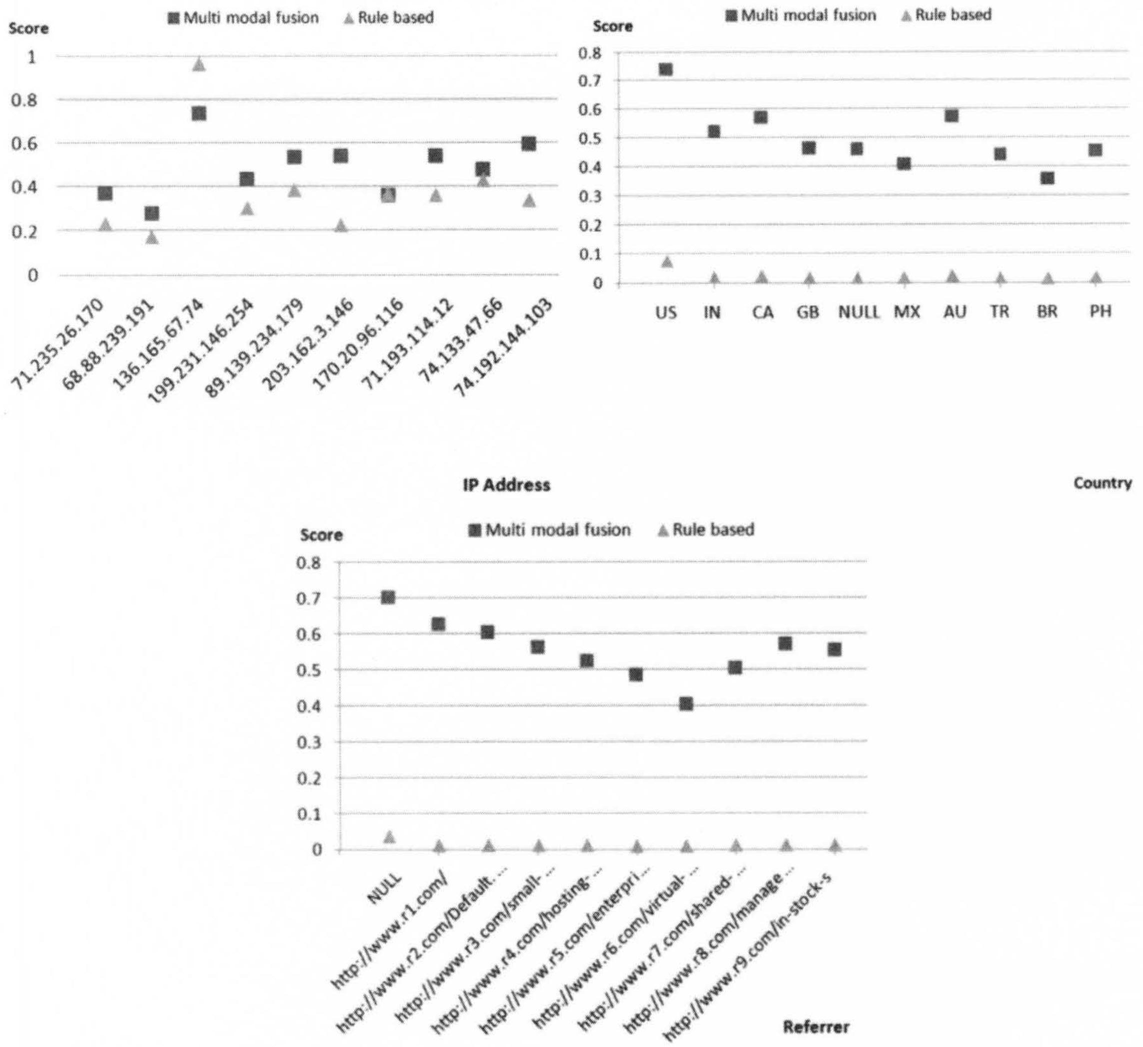


Figure 6.13: Variation of IP Score (left), Country Score (center), and Referrer Score (right)

to cover wider area in the time domain. While the rule based module deals with events within couple of minutes of each other the outlier detection module handles events in a 24 hour window. Figure 6.13 (center) shows the final scores of top 10 countries from which we have received most of the traffic. With the rule based module alone we were unable to detect patterns and variations in the time axis. Therefore almost all countries have a score less than 0.1, which implies clicks from these countries are not suspicious at all. But with the outlier module, which keeps track of traffic for extended period of time, we were able to detect abnormal traffic from most of the countries. For example some of these countries send traffic only during certain hours of the day.

A similar behavior is observed with the top referrers of traffic to hosting.com site. Figure 6.13 (right) shows the variation of scores of top 10 referrers. All these referrers appear normal when they are evaluated only with the rule based system. But when they are evaluated together with click map module and the outlier detection module referrer scores were drastically increased. Some of these referrers are from outside the US. When the countries suspicion score increases so does the scores of associated referrers. For example we mentioned in the above example that certain countries send traffic only in certain hours of the day. When we include the click context it is observed that most of these referrers are associated with those countries. This behavior will be very hard to detect if we are using only the rule based score.

In traditional system (rule based) country and referrer did not influence on the score almost at all. Inclusion of additional modules make country score and referrer score become much more sensitive. For example the new system include country parameter in 73% of clicks from US in the final score.

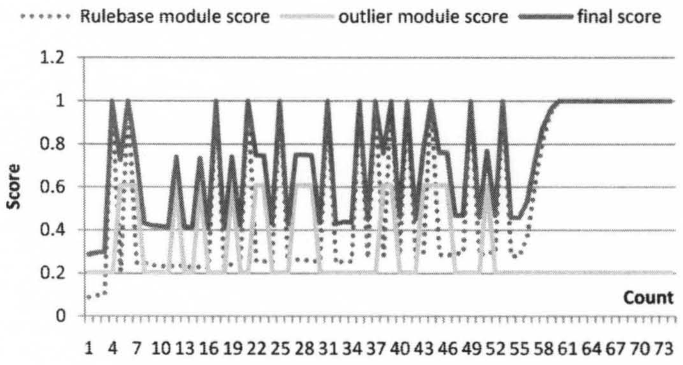


Figure 6.14: Variation of score for a blacklisted IP

We have mentioned that highly suspicious members of IP, referrer, country, etc. are blacklisted once they are detected by the CCFDP system. Figure 6.14 shows the variation of scores for such a blacklisted IP. In this graph 0 corresponds to a valid click and 1 is assigned for an invalid click. Even a valid IP can have many 1s recorded as its final score due to situations like double clicks. We usually do not penalize users for double clicking on an advertisement since the user may be accustomed to Microsoft's default way of choosing something on the screen. This can be clearly observed in Figure 6.14 where there are clicks for the same user that hit the score 1 even before it gets blacklisted. The situation changes if the clicks are repeated more than 2 times during a given time interval. Once enough suspicious activities are detected the IP is finally moved to the blacklisted database and the search providers are notified. For example In Figure 6.14 this IP is moved into the fraudulent database after 59 clicks. With the support from search providers our system will block future traffic generated from this particular IP reaching a client's website.

6.7.3 Comparison of distribution of final score

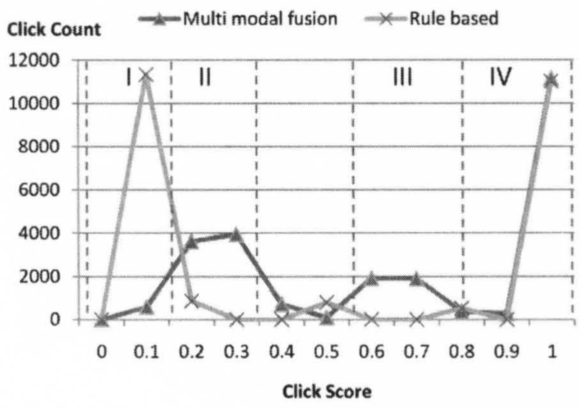


Figure 6.15: Score distribution

Figure 6.15 shows the distribution of final scores for all the clicks with two versions of CCFDP. The lighter graph (L) corresponds to the first version of CCFDP where only rule based module was used. The darker one (D) is the new version with multiple modules. Area I represents most of the valid clicks. This corresponds to the records with attributes which do not have presence in the fraudulent database and all key attributes satisfies the requirements defined in the algorithm to be a legitimate click. The percentage of traffic present in Area I with system L is much higher than that of system D. With the inclusion of multiple models the suspiciousness of clicks has increased and the graph is shifted to the Area II with the system D, which is still in the safer region. Area III shows the suspected clicks. These are records with the attributes present in the fraudulent database or attributes that exceed certain threshold values. It can be clearly seen in the graph how the scores have increased after fusing multiple pieces of evidence from different modules. Area IV includes invalid clicks. Blocked traffic is identified as clicks with highly suspicious scores usually greater than 0.9. As shown in Table 6.6 with the traditional system (rule based system) we were able to block only 520 fraudulent clicks but with the muti model system it was 643, which is about 24% additional clicks. We believe that advertisers should not be billed for any of these clicks.

Table 6.6: Distribution of clicks in each region in Figure 6.15

	I	II	III	IV
Rule based system	12198	1	3	520
Multi-model system	4197	4650	3817	643

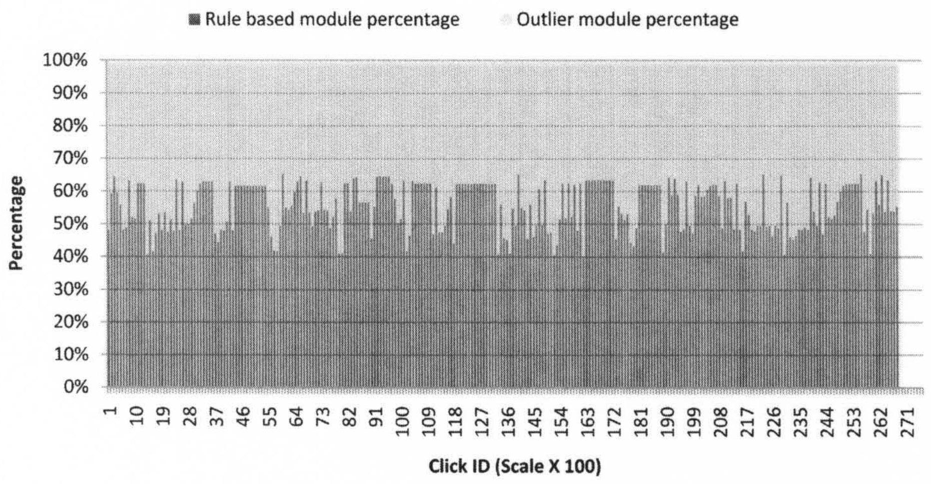


Figure 6.16: Percentage Participation

Figure 6.16 shows the percentage participation of each module in the final score calculation. Remember that the click map module is already used as a screening module to filter invalid clicks, where mouse clicks are recorded off-positioned to the advertisement. Light area of Figure 6.16 represents the rule based module participation and dark area represents the outlier detection module participation.

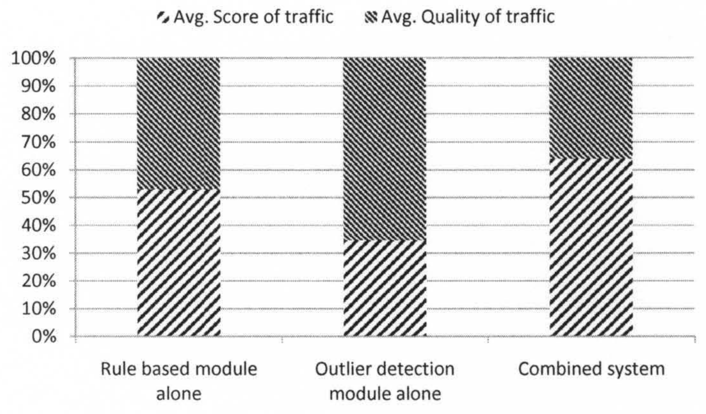
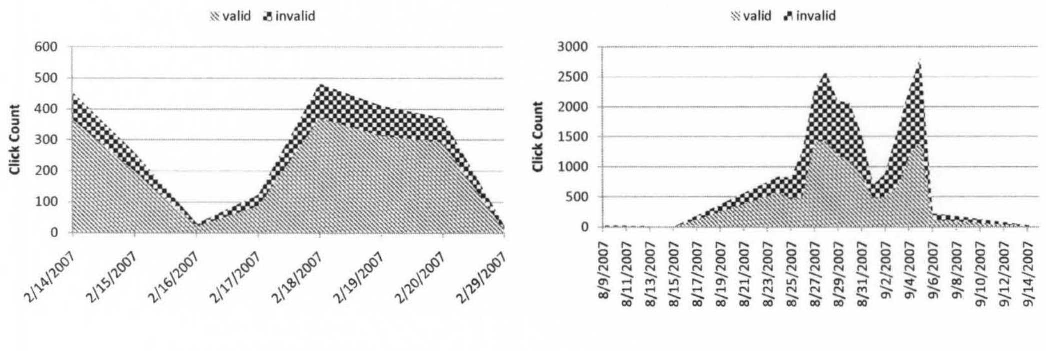


Figure 6.17: Improvement of quality of traffic

6.7.4 Comparison of improvements in quality of traffic

We looked at the changes in quality of traffic after implementing the multi-model based CCFDP system. A summarized version is depicted in Figure 6.17. The dataset was used in the rule based module alone and found that the average 53% of traffic is suspicious [Kantardzic et al., 2008]. When running the outlier detection module alone on the dataset we discovered that about 34.6% of all clicks had one or more attributes that were found to have an outlying attribute-value pair count [Kantardzic et al., 2009]. Clicks found to have an outlier will contribute evidence effecting partial scores. The CCFDP system will compute the final score measuring suspicion for each click. And with the multi-model system classified about 64% of paid traffic as fraudulent. In addition we have observed the changes in the online fraudulent database. In the traditional system, only with rule base, the fraudulent database has recorded 71 IPs as fraudulent. The multi-model system recorded 283 IPs as fraudulent with the same data set, which is nearly 4 times more than the traditional system. This is a greater improvement in terms of prevention of fraudulent traffic. As we discussed in Figure 6.13, the traditional system has very little effect on country score and referrer score when calculating the total score. But with the multi-model system scores for countries such as India, Morocco, Mexico have shown enough suspicion. Clicks came from these countries received a higher fraudulent score but the system did not have enough suspicious clicks to block any of the countries completely. A similar results are observed for referrers.

We defined the quality of traffic as $(1 - score)$. Using only the rule based module and the outlier module we have about 47% and 65% quality scores respectively. With the combined model we were able to get much better traffic with about 36% of quality. With these results we can see that the multi-model based CCFDP system is capable of improving the detection of fraudulent traffic at least by 10% compared to same models working alone.



(a) Second month (b) Eighth month

Figure 6.18: Click traffic in second and eighth months of 2007

Additionally, we looked at the number of suspicious clicks in the Hosting.com advertising traffic during two different periods, where we had considerably more clicks. Figures 6.18 (a) and 6.18 (b) show advertising traffic during these periods (second month and eighth month). In these time periods outlier detection module has found increasing number of suspicious alarms[2]. The multi-model system found 10740 clicks (out of 24528 total clicks) to be fraudulent, while the rule based system only detected 6990(not shown in the figure). In addition we analyzed the volume of total clicks from Google and its partner network during these two periods of time.

Figure 6.19 shows the total and invalid traffics from Google and Google partner networks for the second month and the eighth month. First thing to observe is there is

[2]A closer inspection of the data shows that a number of factors contributed to this higher rate of outlier detection. One such factor was that there was a major shift in the keywords used in the Google advertising campaign for the website.

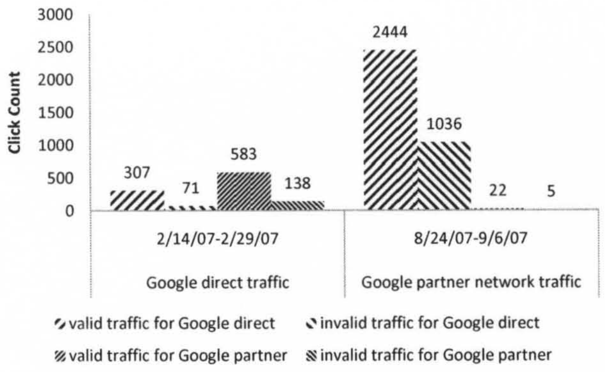


Figure 6.19: Traffic analysis for Google

much higher volume of traffic in the eighth month compared to the second month. Second thing to observe is that traffic from Google partner networks in the eighth month is almost negligible. In the second month out of 307 direct Google referrals 71 are observed invalid, while 138 of 583 Google partner network referrals are detected invalid. In the eighth month total Google only traffic is 2444 and nearly 50% (1036) of that traffic is found to be invalid.

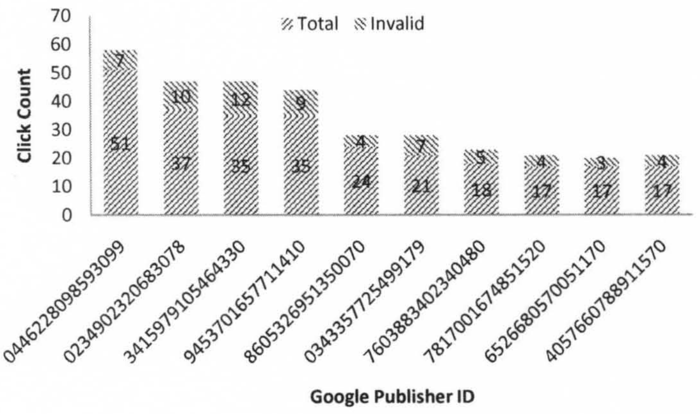


Figure 6.20: Referrer analysis with Google Publisher ID

We also looked at the top referrers in the Google partner network during the second month. Figure 6.20 shows these referrers identified by their Google publisher ID. These statistics are generated using the extended context of click data. We looked at the top five countries, other than the US, present in the invalid Google only traffic. They are depicted in Figure 6.21. Both India (IN) and Turkey (TR) remained as a common candidate in both lists. It is the outlier detection module that detected traffic from these countries becoming suspicious. These evidence is added to the overall system for the purpose of detecting fraud.

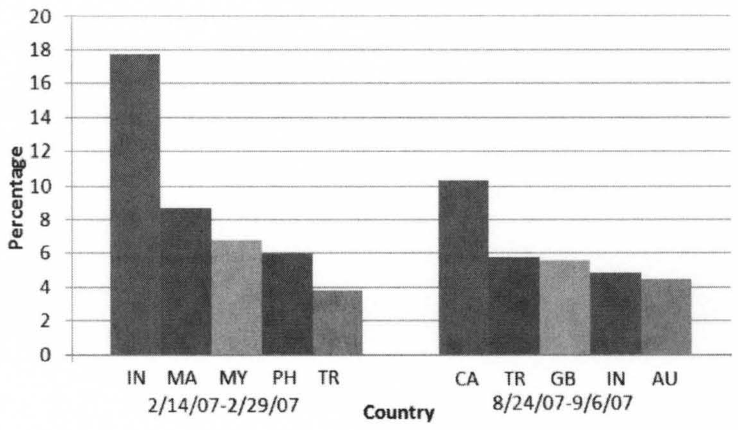
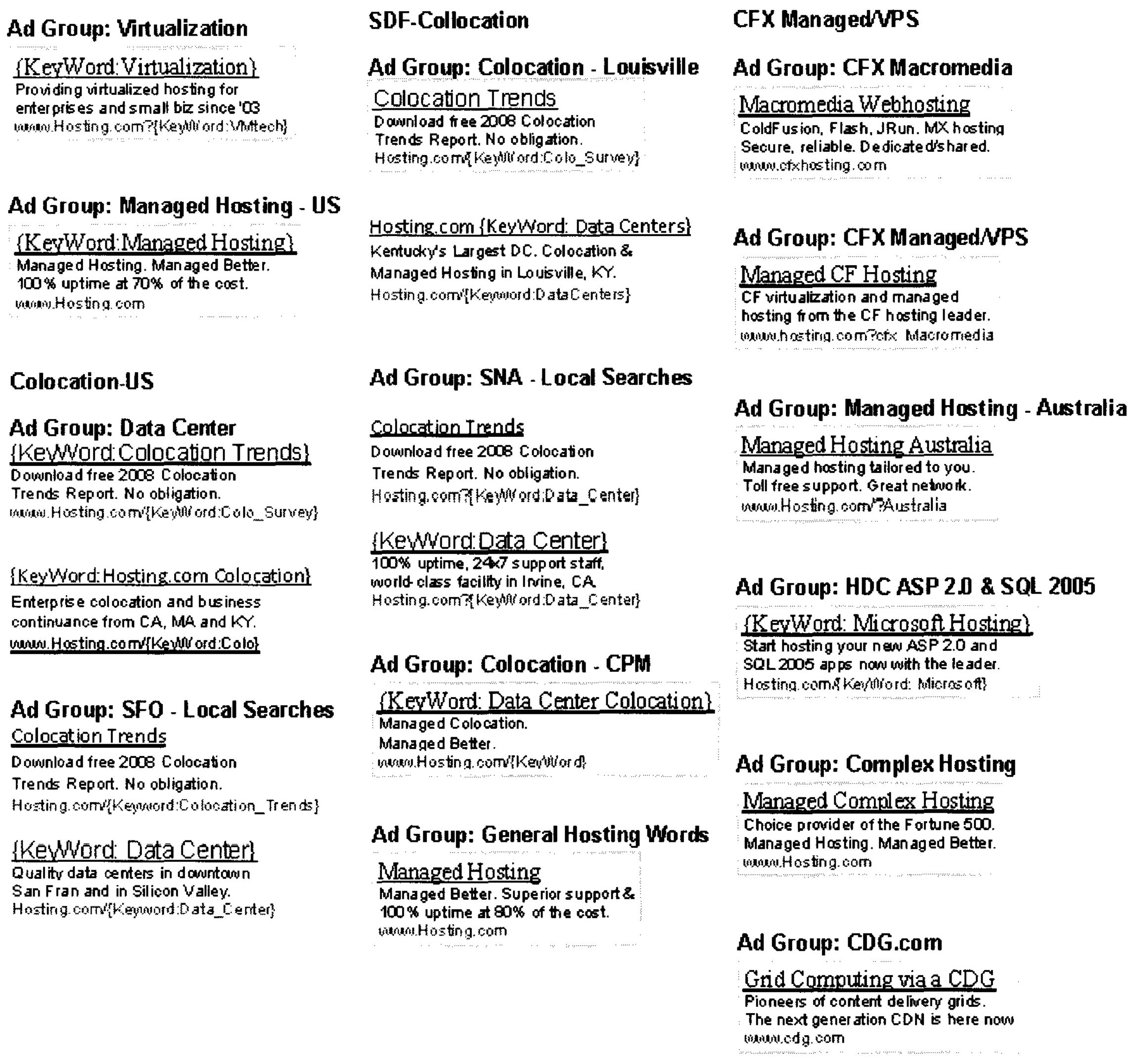


Figure 6.21: Top 5 country lists for invalid Google only traffic

6.7.5 Comparison with Google Adwords

In the previous section we have shown the comparative study between the initial and the improved version of the CCFDP system. In this section we discuss the comparative analysis that we have carried out with Google Adwords program. We have partnered with hosting.com. All of our experiments use click data from Hosting.coms website. Hosting.com is a global company which provides hosting solutions to "business critical data assets" as explained on their website. The data which will be used for the experiments to follow comes from the paid traffic of their advertising campaign. Figure 6.22 shows the list of advertisements that we have used in the Google Adwords campaign.

In the year 2007 data collected during the period from 01-18-2007 to 12-31-2007 and in the year 2008 data collected during the period from 01-01-2008 to 07-15-2008. During analysis we have found out that data are not continuous and changes are made in search advertising networks. Even though the company made available all the data to us, the official Google Adword reports are available only for 3 months of the year 2008. Therefore for the year 2007 we did not do any comparisons.



Ad Group: Virtualization

{KeyWord:Virtualization}
Providing virtualized hosting for
enterprises and small biz since '03
www.Hosting.com?{KeyWord:VMtech}

Ad Group: Managed Hosting - US

{KeyWord:Managed Hosting}
Managed Hosting. Managed Better.
100% uptime at 70% of the cost.
www.Hosting.com

Colocation-US

Ad Group: Data Center

{KeyWord:Colocation Trends}
Download free 2008 Colocation
Trends Report. No obligation.
www.Hosting.com/{KeyWord:Colo_Survey}

{KeyWord:Hosting.com Colocation}
Enterprise colocation and business
continuance from CA, MA and KY.
www.Hosting.com/{KeyWord:Colo}

Ad Group: SFO - Local Searches

Colocation Trends
Download free 2008 Colocation
Trends Report. No obligation.
Hosting.com/{Keyword:Colocation_Trends}

{KeyWord: Data Center}
Quality data centers in downtown
San Fran and in Silicon Valley.
Hosting.com/{Keyword:Data_Center}

SDF-Collocation

Ad Group: Colocation - Louisville

Colocation Trends
Download free 2008 Colocation
Trends Report. No obligation.
Hosting.com/{KeyWord:Colo_Survey}

Hosting.com {KeyWord: Data Centers}
Kentucky's Largest DC. Colocation &
Managed Hosting in Louisville, KY.
Hosting.com/{Keyword:DataCenters}

Ad Group: SNA - Local Searches

Colocation Trends
Download free 2008 Colocation
Trends Report. No obligation.
Hosting.com?{KeyWord:Data_Center}

{KeyWord:Data Center}
100% uptime, 24x7 support staff,
world-class facility in Irvine, CA.
Hosting.com?{KeyWord:Data_Center}

Ad Group: Colocation - CPM

{KeyWord: Data Center Colocation}
Managed Colocation.
Managed Better.
www.Hosting.com/{KeyWord}

Ad Group: General Hosting Words

Managed Hosting
Managed Better. Superior support &
100% uptime at 80% of the cost.
www.Hosting.com

CFX Managed/VPS

Ad Group: CFX Macromedia

Macromedia Webhosting
ColdFusion, Flash, JRun. MX hosting
Secure, reliable. Dedicated/shared.
www.cfxhosting.com

Ad Group: CFX Managed/VPS

Managed CF Hosting
CF virtualization and managed
hosting from the CF hosting leader.
www.hosting.com?cfx Macromedia

Ad Group: Managed Hosting - Australia

Managed Hosting Australia
Managed hosting tailored to you.
Toll free support. Great network.
www.Hosting.com/?Australia

Ad Group: HDC ASP 2.0 & SQL 2005

{KeyWord: Microsoft Hosting}
Start hosting your new ASP 2.0 and
SQL 2005 apps now with the leader.
Hosting.com/{KeyWord: Microsoft}

Ad Group: Complex Hosting

Managed Complex Hosting
Choice provider of the Fortune 500.
Managed Hosting. Managed Better.
www.Hosting.com

Ad Group: CDG.com

Grid Computing via a CDG
Pioneers of content delivery grids.
The next generation CDN is here now
www.cdg.com

Figure 6.22: Ad Groups used in the campaign

(a) Monthly Click Traffic Distribution

Table 6.7 and Figure 6.23 summarize the CCFDP results for each month of the year 2007 for traffic from Google and Google partner network. During this period we have analyzed total of 17586 clicks and CCFDP has determined 12258 of 17586 as valid clicks, i.e. about estimated click fraud of 30.3%. Blocked traffic are the clicks that were invalidated immediately from the CCFDP system because they were found fraudulent in previous analysis and were recorded in the blacklisted database. For example in the month of January, 528 clicks from 9 different IP addresses were immediately invalidated. Rest of the clicks are further analyzed and 1623 of them are found to be invalid.

Table 6.7: CCFDP Traffic Analysis for Hosting.com in 2007

CCFDP					
Month	Total traffic	Blocked traffic	Blacklisted IPs	Invalid clicks	Valid clicks
January	9230	528	9	1623	7080
February	4316	84	0	818	3414
August	447	2	1	183	50
September	1380	64	3	878	478
October	1434	72	2	1336	26
November	547	11	0	377	159
December	232	5	0	113	114

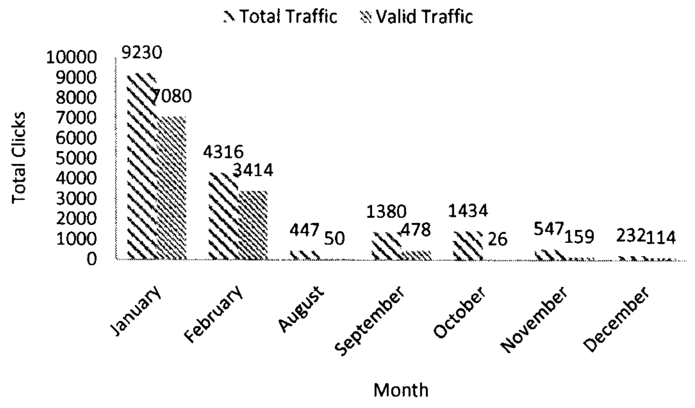


Figure 6.23: Comparison of Valid vs. Invalid Clicks

Table 6.8 and Figure 6.24 summarize the CCFDP results for each month of the year 2007 for traffic from Google and Google partner network. During this period we have analyzed total of 8167 clicks and CCFDP has determined 4042 of 8167 as valid clicks. Blocked traffic are the clicks that were invalidated immediately from the CCFDP system because they are found fraudulent in previous analysis and were recorded in the blacklisted database. For example in the month of January, 7 clicks from an IP addresses were immediately invalidated. Rest of the clicks are further analyzed and 4042 of them are found to be invalid.

Table 6.8: CCFDP Traffic Analysis for Hosting.com in 2008

CCFDP					
Month	Total traffic	Blocked traffic	Blacklisted IPs	Invalid clicks	Valid clicks
January	787	7	1	282	498
February	1617	8	0	744	865
March	1473	37	3	498	938
May	611	3	2	400	208
June	3679	135	6	2201	1343

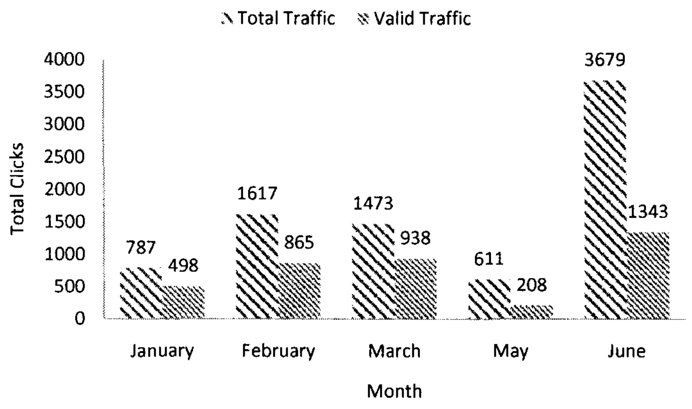


Figure 6.24: Comparison of Valid vs. Invalid Clicks

Table 6.9 summarizes the Google Adsense reports for each month of the year 2008 for traffic from Google and Google partner network. In March Google has reported 1247 valid clicks and they have charge \$2.67 on average per click. Similarly for the months of May and June total ad expenditure is \$15,562 and \$9,624 respectively. At the end of the campaign Hosting.com paid \$28,516.

Figure 6.25 shows the comparison of CCFDP and Google Adsense analysis for three months of the same campaign. In the months of March, May, and June CCFD reported to have received 1473, 611, and 3679 paid clicks. Among those it has invalidated 498, 400, and 2201 clicks as fraudulent in the corresponding months. This resulted total valid

Table 6.9: Google Adsense Traffic Analysis for Hosting.com in 2008

Google Adsense			
Month	Valid clicks	Avg.cost per click	Total cost
March	1247	$2.67	$3330
May	3304	$4.71	$15562
June	2839	$3.39	$9624

traffic of 938, 208, and 1343 for the months of March, May, and June respectively. One thing to notice here, that there was a problem in the data collection in the month of May and because of that total traffic does not represent the traffic of the entire month.

We requested the Google Adwords reports for the same duration. They have reported 1247, 3304, and 2839 of valid clicks in the months of March, May, and June. In all 3 months CCFDP has determined more invalid traffic compare to Google. If we stick to the same average cost per click, this analysis suggest that Hosting.com could have saved more than $20,480 during this period.

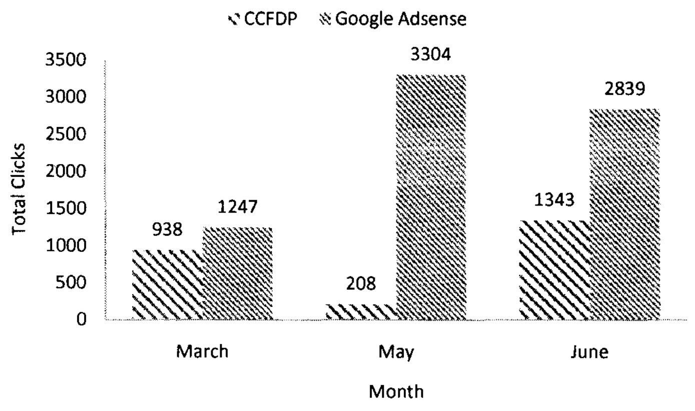


Figure 6.25: Comparison of valid traffic in CCFDP and Google Adwords in 2008

Current version of the CCFDP system provides better results compared to its first version. This is mainly due to the use of extended click context to analyze and estimate the quality of a click. There are further improvements to be done to make it much more robust and reliable to detect robot clicks, which are clicks generated by software programs known as clickbots. These clickbots click on ads and issue HTTP requests for advertiser web pages. There are many types of clickbots used on the Internet. Some are "for-sale" clickbots, while others are malware. For-sale clickbots such as the Lote Clicking Agent, I-Faker, FakeZilla, and Clickmaster can be purchased online. They typically use anonymous proxies to generate traffic with different IP addresses. An anonymous proxy server generally attempt to anonymize web surfing. However IP diversity usually is not enough to hide click fraud attacks conducted by such software, and traffic generated by them is identifiable. Malware type clickbots infect machines in order to achieve IP diversity, and their traffic may or may not be as easily identifiable as that generated by for-sale clickbots. In order to detect and remove robots, we need to have a better characterization of the distribution of click behavior. Current version of the CCFDP system detects some types of clickbots, but improvements are necessary especially with a wide variety of new clickbots occurring on Internet. In Chapter 7 we detail the new improvements to the CCFDP which we have made in its development phase 3.

CHAPTER 7

EXTENDED ANALYSIS OF CLICK BOTS IN CCFDP

One of the most significant threats to the Internet advertising today is the threat of click bots, which are networks of compromised machines under the control of an attacker. It is difficult to measure the extent of damage caused on the Internet by these bots, but it is widely accepted that the damage done is significant. In future, most of the fraudulent activities will be carried out by these bots because they are less expensive to develop or buy and easy to maintain. Therefore, future click fraud detection systems must incorporate robust detection techniques to protect their customers from these sophisticated click bot attacks.

On the other hand, users of a practical click fraud detection solutions expect these duplicate detection mechanisms to run in realtime. In order to provide real time results, solution providers should utilize data structures that can be updated in real time. In addition, If the actual volume of clicks (per unit time) is high, space requirements per click should be the lowest possible thus data structures with constant space requirements or sublinear space requirements such as $O(log(n))$ or $O(log(log(n)))$ are desirable (n is the number of elements processed).

Most of the click fraud solution providers, including search engines and third party solution providers, claim their rule-based expert system is the best among the others taking the advantage of keeping rules as a secret weapon. They do not disclose information about the set of rules due to fear of competition. This situation even led to multi-million dollar settlements in the recent years. Due to the lack of verifiability of click fraud solutions, it is inevitable that the trust between service providers and advertisers is degraded. Since real-world click fraud solutions are usually kept secret for fear of competition, it is practically impossible to study many of them in a single context.

In the "new CCFDP", that we have developed in phase 3, we have addressed the above issues. In section 7.1 we discuss the naive Bayesian classifier that we have developed to detect Smart ClickBot type clicks. In section 7.2 we elaborate on the proposed space efficient Bloom filter based data structure to process clicks. In section 7.3 we discuss the modeling of knowledge and validation (KV) model for rule based expert systems used in click fraud detection.

7.1 Extended analysis of click bots

Click bots represent one of the fastest growing threats on the Internet advertising, given that they adapt perfectly to the new malware dynamic in which threat creators are no longer searching for notoriety, but for financial returns. With this in mind, they try to ensure their creations are installed without arousing the suspicions of users or security companies.

The current situation requires the use of proactive technologies, which can detect unknown threats by examining their behavior. A click bot does not necessarily click on an ad to issue a click. It is programmed to generate "fake" clicks that mimics actual clicks. Most of the time it is an HTTP request that is artificially generated. Therefore it is important to verify whether the click is originated from an authentic browser such as Internet Explorer, Mozilla, etc.

A number of challenges make this task difficult. First, the amount of data to process is often huge, on the order of terabytes per day. Thus any method that mines the data for identifying bot traffic has to be both efficient and scalable. Secondly, most of these data are not disclosed due to privacy, security and business policy issues. Furthermore, with many bot-net hosts available, attacks are getting increasingly stealthy with each host submitting only a few clicks to evade detection. Therefore, click bot detection methods cannot just focus on aggressive patterns, such as in Bahama bot, but also need to examine the low rate patterns that are mixed with normal traffic. Third, attackers can constantly craft new attacks to make them appear different and legitimate; thus we cannot use the training-based approaches that derive patterns from historical attacks. Finally, with the lack of ground truth, evaluating detection results is non trivial and requires different methodology and metrics than the detection methods.

In the new CCFDP it was one of our goals to achieve the target of developing a robust detection mechanism against these low noise click bot. We have developed a novel classifier that for this purpose. It was successfully tested to detect the clicks from SmartClick Bot, an advanced and intelligent click bot. Before detailing the machine learning techniques that we have used, the following section gives a brief introduction to the SmartClick Bot.

7.1.1 An Overview of the Smart ClickBot

The Smart ClickBot is a software Web robot that clicks on ads (by issuing HTTP requests for advertiser web pages) to help an attacker conduct click fraud[1]. It was first detected and reported by the NetMosaics click fraud detection system in 2010 [Kantardzic et al., 2008, Kantardzic et al., 2010a, Kantardzic et al., 2010b]. Smart ClickBot is a for-sale click bot and it can be purchased[Walgampaya and Kantardzic, 2010]. Once installed and configured the Smart ClickBot is able to act by itself. It uses anonymous proxies to generate traffic with IP diversity. It also has a random user-agent generator that generates user-agents registered to well known HTTP browsers. By doing so, it can mimic a request originated from a valid browser because click fraud detection solutions usually suspect clicks without a valid user-agent field [Tan and Kumar, 2002]. To make it look more realistic it can even attach a referrer field. Referrer in pay-per-click system is a website that helps a web user to reach another website. Therefore if the referrer field carries values correspond to famous search engines or other popular websites it will be least suspicious to anybody observing the server logs.

Operator of the Smart ClickBot can set the time interval between successive clicks, known as the *Click-Through-rate*(CTR)[2] and configure to run multiple click campaigns

[1]the act of generating illegitimate clicks to make profit or deplete competitor advertisement budget.

[2]CTR is a way of measuring the success of an online advertising campaign. A CTR is obtained by

simultaneously. The Smart ClickBot has 3 distinct campaign modes. They are: single hit, list-like, and banner-like, which are especially designed to suit different webpage structures. Once the bot loads the webpage that has the advertisements the user can specify where to click.

In the next section, we discuss in detail the systematic approach taken by the NetMosaics system to detect click patterns generated by the Smart ClickBot.

7.1.2 Methodology for Smart Clickbot detection

1. Data Collection, Pre-processing and Session identification

   Interactions with a Web server, either by humans or software, are recorded in the server access logs. To characterize the behavior of bots statistically, we need to be able to isolate the behavior of robots from that of the general population of (human) Web users.

   Most of the existing bot detection systems uses only the server side data in their analysis and therefore entirely depends on these server log data to identify robot *sessions*. A session is the duration that a user (either human or software bot) maintains an active HTTP connection with the server. But, because of the stateless nature of HTTP traffic, incoming requests are considered and logged as independent events. Therefore, access logs do not contain any information that could relate together requests issued during a single "visit" of one user to the Web-pages of a Web server [Stassopoulou and Dikaiakos, 2009].

   Furthermore, in [Tan and Kumar, 2002] Tan and Kumar stated that "Without client-side tracking, cookies or embedded session identifiers, it is extremely difficult to identify the individual sessions in the Web server logs reliably". Even though there were some alternative attempts[Pirolli et al., 1996] to group server logs into sessions,

dividing the "number of users who clicked on an ad" on a web page by the "number of times the ad was delivered" (impressions). For example, if a banner ad was delivered 100 times (impressions delivered) and 1 person clicked on it (clicks recorded), then the resulting CTR would be 1 percent.

none of them showed promising results.

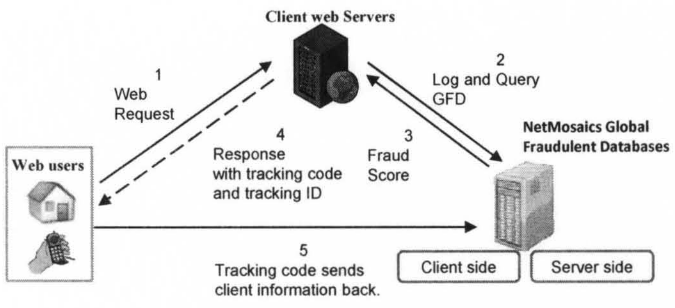


Figure 7.1: The NetMosaics data collection process.

Therefore, we developed a click fraud detection system, NetMosaics, that uses both server side data and client side data to better understand the context of the click, while providing an easy platform to generate user sessions. Figure 7.1 shows the high-level processing flow of the NetMosaics system. In this system user sessions are easily matched with a unique tracking number that is shared by both server side and client side data. Robot generated traffic usually do not have client side entries. Therefore, none of the bot traffic will be merged, and they will be left in the server side log. The matched traffic is further analyzed by the NetMosaics system for more suspicious activities to improve the quality of the incoming traffic. Only the improved matched traffic will be delivered to the NetMosaics clients. What is left in the server side is separately analyzed, which is the scope of this section, for potential bot networks.

Our system has been collecting and analyzing click data continuously since it was launched in 2004. While delivering higher quality traffic to our clients we periodically analyze what is left in server side, which are mostly bot data, collectively to identify unknown bots and their behavioral patterns. Discovery of Smart ClickBot is a result of such an attempt. One of our honeypot[3] servers was infected with the

[3]honeypot is a trap set to detect, deflect, or in some manner counteract attempts at unauthorized use of information systems. Generally it consists of a computer, data, or a network site that appears to be part of a network, but is actually isolated and monitored, and which seems to contain information or a resource of value to attackers.

Smart ClickBot. After the detection of the bot we were able to reverse engineer it to obtain a copy.

Multiple copies of the isolated click bot is then installed and used to carry out attacks in a controlled environment. Bot clicks are collected for a period of 7 days from 3/3/2011 to 3/9/2011. 1000s of different IP addresses are generated through proxy servers. Also, another 100s of referrer sites are used. Time between clicks is varied randomly between 0 - 1000 seconds. They are configured to issue HTTP clicks at www.thebestmusicsites.org, a Web site that we have designed, that has both text and banner advertisement links. A bot is set to issues clicks between 0-5 in a given session. During the experiment, bot configurations are randomly changed to maximize the diversity. This data set will be available publicly to researchers who wants to test their click fraud detection systems against this new type of click bot. Since Smart ClickBot has been developed with utmost care to not to be detected, the techniques that we have developed may help to detect even other types of click bots.

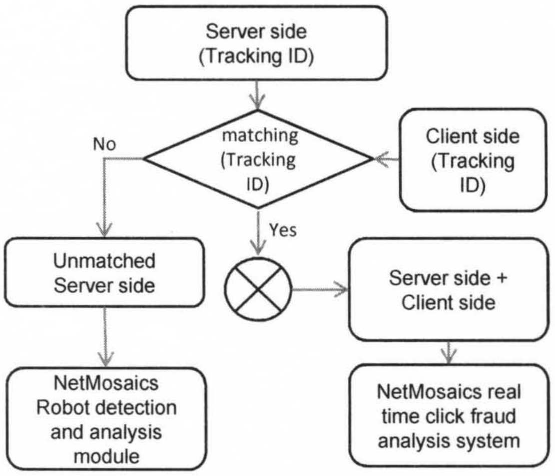

Server side (Tracking ID)
matching (Tracking ID)
No
Client side (Tracking ID)
Yes
Unmatched Server side
Server side + Client side
NetMosaics Robot detection and analysis module
NetMosaics real time click fraud analysis system

Figure 7.2: Robot data collection process.

Figure 7.2 shows the high level view of the flow diagram for bot traffic isolation. These isolated potential bot traffic is pre-processed to remove known bots. These include search engine crawlers such as Google's googlebot, Yahoo!'s Yahoo slurp, and Bing's bingbot, known click bots such as Clickbot.A and Bahamabot, link crawlers and news bots etc. For this purpose, we have used the data available at [Database, 2011]. Top 10 of those filtered bots are shown in Table 7.1 with their user agent and frequencies. Data what is left after filtering is then divided into sessions based on the techniques explained in [Tan and Kumar, 2002].

Once the bot data is grouped into sessions, our main idea in the experiment is to classify server side data into two classes: Class 1, and Class 2, where Class 1 will have "clicks" originated from Smart ClickBot and everything else will belong to the Class 2. For example, Class 2 may contain human clicks that do not accept cookies or javascript in their devices or they may be clicks that has only server side information due to an error in HTTP communication between server and client or it may be even new software bots that are not discovered yet. Therefore the next step is to experimentally derive the properties of each session that will distinguish clicks in Class 1 from that of Class 2. Table 7.2 presents a summary of attributes that can be derived from the sever sessions. Some of these features are temporal, while some are binary. Extraction process of these features is discussed in the following section.

2. Context Feature Extraction

   NetMosaics explores the distributed nature of stealthy attacks. Since click bots are pre-configured, the generated traffic by them is usually similar in nature. NetMosaics leverages this property and aims to identify groups with similar activities.

   (a) Periodicity of Smart ClickBot

      Web robots, especially click bots, usually exhibit periodic behavior because they are preset to activate after a certain time interval. At this point they randomly select a (IP, referrer, user agent) combination from the predefined lists and issue clicks in the form of HTTP requests. By observing the collected data we have seen these lists are updated daily. Therefore we can assume the

Table 7.1: Filtered UserAgents

User Agent	Requests
google.com	244
search.msn.com	149
Mozilla/5.0 (compatible YandexBot/3.0 http://yandex.com/bots)	120
yahoo.com	105
Sogou web spider/4.0(http://www.sogou.com/docs/)	79
Mozilla/5.0 (Windows U Windows NT 5.1 en rv:1.9.0.13) Gecko/2009073022 Firefox/3.5.2 (.NET CLR 3.5.30729) SurveyBot/2.3 (DomainTools)	55
Mozilla/5.0 (compatible bingbot/2.0 http://www.bing.com/bingbot.htm)	42
Netcraft	32
whois.sc	21

values in the lists to be the same for at least 24 hour period. Previous studies of Web robots also support the 24 hours threshold time[Tan and Kumar, 2002]. For each IP we extracted all requests originated within the past 24 hours. By plotting the time activity (i.e. the active and inactive periods of time) of bot processes issuing requests, we observed that several of them seem to exhibit, at least partially, a periodic pattern. We investigated further this observation and verified the periodicity for several IP addresses used by the Smart ClickBot and estimated their time cycles.

For this task, we used the Fast Fourier Transform (FFT). The FFT maps a function in the time field to a, complex in general, function in the frequency field [Dikaiakos et al., 2005]. The idea is that by observing peaks of magnitude in the frequency field we can easily conclude that time activity has periodicity. The frequency coordinate of each possible peak is inversely proportional to the

Table 7.2: Summary of attributes derived from the Server sessions.

Id	Attribute Name	Remark	Purpose
1	Periodicity	Periodicity of the attack	Feature
2	trackingIDs	Tracking IDs per IP	Feature
3	multiAgents	Unique user agents recorded per single IP	Feature
4	referrerPattern	Referrer and IP distribution	Feature
5	HEAD	Page requests made with HEAD method	Classify
6	GET	Page requests made with GET method	Feature
7	POST	Page requests made with POST method	Feature
8	clickRate	Maximum clicks per session	Feature
9	Duration	Duration of session	Feature
10	imageRequests	Percentage of image requests	Feature
11	pdf/ps	Percentage of pdf/ps requests	Feature
12	4xx	Percentage of 4xx error responses	Feature
13	Robot.txt	Whether Robot.txt file accessed during the session	Classify
14	% proxy	Percentage of proxy servers used	Feature

time cycle of the periodicity. Since we are not interested in the phase of the frequency plot, we illustrate the spectral density function, which is the square of the magnitude of the FFT.

Before implementing the FFT, time is assumed to be sliced; we used a 30 second time interval (granularity). Ideally, the granularity should be as small as possible, but we tried to keep the number of resulting points relatively small for a faster FFT computation.

We count the requests issued from an IP address of interest in each time interval. Because our focus of interest at this stage is on the presence of some periodic action, we assign the value of one to the intervals that have at least one hit and the value of zero to the ones with zero hits. Consequently, we produce an ON-OFF signal that represents the Smart ClickBot's time activity for the selected granularity. This signal is passed as input to the FFT function. The resulting diagrams reveal a periodicity in the requests issued by IP addresses belonging to the click bot; in some cases this phenomenon is rather intense.

Figure 7.3 presents the ON-OFF signal of an IP address. In Figure 7.4, we plot the power spectral density function. FFT specifies the main periods observed on that signal. For example, in Figure 7.3, we observe a periodic behavior between 05:15 - 08:15, which corresponds to the peak of around 0.15 in the Figure 7.4.

Similar results are observed for couple of other IP addresses. We have not seen similar patterns from IP addresses that do not belong to Smart ClickBot, which were left in the server logs. We can therefore conclude that periodic activity can be expected from the Smart ClickBot. Hence this feature will be binary and for IPs that shows periodicity we assign a value 1, while the rest, including some of Smart ClickBot IP addresses that does not show the same behavior, are assigned 0.

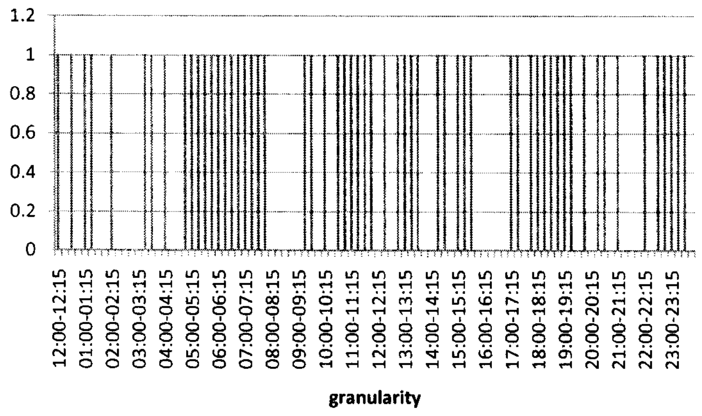


Figure 7.3: ON-OFF Signal for an IP used by Smart ClickBot.

(b) Tracking ID per IP

The NetMosaics system generates a 128bit unique tracking id (in the form of a cookie) for every server side request. This tracking id is installed in the

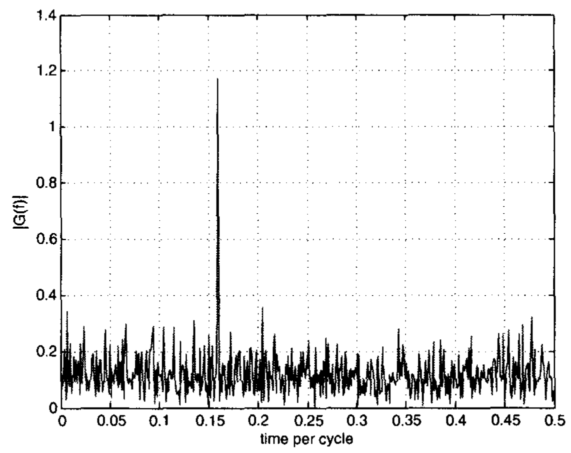


Figure 7.4: Power spectral density for an IP used by Smart ClickBot.

client's computer and will be incorporated with every revisit request to the same website by the client in the next 24 hours. Usually software bots do not accept these cookies. Therefore, they generate new tracking id for every visit to the same website. Smart ClickBot is no different. It generated multiple tracking IDs for the same IP and almost all of these IP addresses belong to free proxy servers. Table 7.3 shows the summary of average number of tracking IDs generated for a 24 hour period by the top 10 IP addresses used by the Smart ClickBot. If the IP address does not belong to a proxy server we usually treat it as a non software bot. For example it may be a human clicker, which does not allow cookies to be installed or who deletes cookies every time the session is over.

We also discovered a secondary property of these IP addresses, which has large number of tracking IDs. We have seen that almost 90% of the time the same set of IPs contain in all the lists of 24 hours. Even though we assumed they are randomly picked, it seems that the bot has some preference to certain proxy servers may be based on the level of animosity of the proxy.

We have also observed a unique pattern in the wagon wheel for visitor distribution. For example, in Table 7.3, day 1 corresponds to 24 hours while day 2 corresponds to visitors for 12 hour period. But the contribution from each IP

Table 7.3: Tracking ID per IP

IP Address	day 1	IP Address	day 2
200.29.216.146	88	115.248.202.21	49
187.4.128.12	85	208.253.158.6	40
190.203.69.69	81	187.11.201.164	39
221.7.145.42	72	200.29.216.146	36
203.153.25.218	59	222.255.28.33	34
115.78.227.155	46	190.203.69.69	33
115.78.224.215	45	203.153.25.218	31
222.255.28.33	43	187.4.128.12	28
125.162.92.233	42	221.7.145.42	26
190.128.218.90	40	89.187.142.113	21

looks alike, immaterial of the day or the number of hours observed. Figure 7.5 shows the wagon wheel distribution of visitors for those two days (day 1(left), day 2(right)).

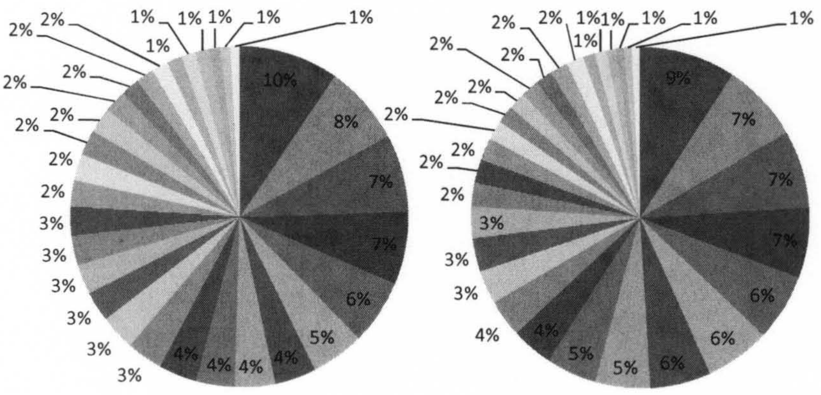


Figure 7.5: Frequency of TrackingID generation.

(c) User Agent per IP

User Agent identifies the type of browser someone is using to surf the Internet. There is a favorite browser for all of us and at least we stick with it for a while. We mark our favorites, bookmark certain websites, maintain history etc. and it will be inconvenient for us to switch browsers. Therefore for a given non-shared IP address variation of the user agent is minimal. User Agent is not defined only for browsers. Even for all the legitimate software bots there are unique user agents. They identify themselves as bots whenever visiting websites if they are cooperative. Bots that are created to carryout fraudulent activities use these user agents to mimic themselves as valid user agents. Smart ClickBot seems to pick a user agent randomly from a pre defined list and does not care about the history of user agents assigned to a particular IP. Therefore, we have seen a large number of user agent values assigned for any given IP. Sometimes the variation exceeds 50, and it is highly unlikely that somebody who is legitimately browsing Internet has over 50 user agents.

Among this list, we have also found some outdated user agents. With normal traffic these type of user agents are rarely reported but a higher percentage can be seen with the traffic generated from the Smart ClickBot. Table 7.4 lists few of these outdated user agents with its average request per 24 hours.

Table 7.4: Frequency of Outdated UserAgents

User Agent	Frequency
Mozilla/4.0 (compatible; MSIE 4.01; Windows 95)	39
Mozilla/4.0 (compatible; MSIE 5.01; Windows 95)	36
Mozilla/4.0 (compatible; MSIE 5.5; Windows 95)	33
Mozilla/4.0 (compatible; MSIE 5.0; AOL 5.0; Windows 95; DigExt)	30
Mozilla/4.0 (compatible; MSIE 5.0; Windows 95; DigExt)	27

(d) Referrer and IP distribution

The referrer field is provided by HTTP protocol to allow a Web client (particularly, a Web browser) to specify the address of the Web page that contains the link the client followed in order to reach the current requested page. Unassigned referrer field is often considered one of the most apparent characteristics of Web robots [Lu and Yu, 2006]. As typical Web robots parse a page and build up the list of pages to be visited, referrer field is frequently left blank. In interactive Web surfing environment, the field would contain the URL that led to the current request. Our analysis reveals that such observation is generally, but not always, true. Especially with the Smart ClickBot to avert the detection they have randomly assign a referrer value. But since this list is concise we have seen a pattern that pretty much every referring site is recorded the same amount of referrals. This does not happen with human clicks and the variation of referrals is usually high. Table 7.5 shows an example for a referral traffic for the www.thebestmusicsites.org website within a 24 hour period.

Table 7.5: Frequency of top referrer sites

Referrer	IP frequency
http://www.referrer1.edu/	30
http://www.referrer2.com/	29
http://www.referrer3.com/	28
http://www.referrer4.com/	27
http://www.eecs.referrer5.edu/	27
http://www.referrer6.com/	26
http://www.referrer7.com/	26
http://www.referrer8.com/	26
http://www.referrer9.org/	23
http://www.referrer10.com/	23

(e) Percentage of HEAD and GET Requests

Many suggest sessions containing a large number of HEAD requests as those generated by web robots [Tan and Kumar, 2002, Dikaiakos et al., 2005]. For example, if all the requests are made using the HEAD method, then the session is most likely created by a Web robot. Guidelines issued to web robot designers strongly recommend that only HEAD method be used to minimize performance impact on web servers. However, [Dikaiakos et al., 2005] reported that many Web bots used HEAD method in less than half (e.g., 10 to 50%) of their requests. In our data, almost all (e.g., over 99.9%) the requests made by Smart ClickBot used GET method. Therefore when the HEAD/GET request percentage is high, we use this feature as a strong feature to separate non Smart ClickBot requests from the server logs.

(f) Percentage of POST Requests

A POST request is used to send data to the server to be processed in some way, like by a CGI script. It is highly unlikely that a Web bot sends POST request to a Web server. Therefore, we can use this feature to isolate human issued requests that are left in the server logs.

(g) Maximum clicks in a session

A click is a request for an HTML file in a Web server. The "Maximum clicks in a session" is a feature corresponds to the maximum number of such HTML requests received within a certain time-window inside a session. The intuition behind this feature is two fold:

i. To isolate human clicks from bot clicks: there is an upper bound on the maximum number of clicks that a human can issue within some specific time-frame t, which is dictated by human factors. To capture this feature, we first set the time-frame value of t and then use a sliding window of time t over a given session in order to measure the maximum sustained click rate in that session. For example, if we set t to 10 seconds and find that the maximum number of clicks within some 10-second time-window inside that session is 40, we conclude that the maximum sustained click rate is 4 clicks per second. This indicates a robot-like rather than a human-like behavior. The sliding window approach starts from the first HTML request of a session and keeps a record of the maximum number of clicks within each window, sliding the window by one HTML request until we reach the last one of the given session. The maximum of all the clicks per window gives the value of this feature.

ii. To isolate Smart ClickBots from other software bots: Smart ClickBot can issue $0 - n$ number of clicks during a session. But we have seen that for every such click Smart ClickBot changes its IP address, tracking id, referrer, and the user agent. Therefore, there is no way to recognize all the clicks belong to a one session. In the server logs all the clicks generated in a session appears as several single click sessions. We can use this implementation drawback to recognize and isolate Smart ClickBot clicks from other type of software bots.

(h) Duration of session

Duration of session is the number of seconds that have elapsed between the first and the last request. Crawler-induced sessions tend to have a much longer duration than human sessions. Smart ClickBot can issue any number of clicks during a session. But we have seen that for every such click it changes its IP address, tracking id, referrer, and the user agent values. Therefore, there is no way to recognize all the clicks belong to one session. In the server logs, all clicks belong to one session, appear as multiple single click sessions. Therefore for this feature, which is binary, sessions having only a single click caries value 0, while the rest carries value 1.

(i) Percentage of image requests

This feature denotes the percentage of requests to image files (e.g. jpg, gif). An earlier study showed that crawler requests for image resources are negligible [Stassopoulou and Dikaiakos, 2009]. In contrast, human generated traffic will have access records to images in a website, because all images are loaded within the user session. Therefore, we can use this feature to differentiate human generated traffic, that for some reason did not have client side activities, from bot generated traffic. The percentage of requests seeking postscript(ps) and pdf files is also a possible feature to use. But in our experiment we did not consider this feature. Previous studies show that, in contrast to image requests, some crawlers, tend to have a higher percentage of pdf/ps requests than humans[Stassopoulou and Dikaiakos, 2009].

(j) HTTP response codes

Web bots such as link validators and email harvestors may have a higher proportion of 4xx error codes in their requests, as they are blindly traversing the web infrastructure. But, clicks from the Smart ClickBot should be very precise because these links are previously checked by humans before launching the attacks, hence we expect fewer 4xx error codes. Human clickers may stand between these two extreme ends because human users are able to recognize, memorize and avoid erroneous links, unavailable resources and servers. Table 7.6 shows the percentages of response codes received from bot and non-bot traffic for a 7 days period. We can clearly see that bot traffic has lesser 404 errors compared to non-bot traffic.

We can also expect lesser 304 response codes with Smart ClickBot traffic. HTTP 304 response code is for "not modified". With this message the web server is basically telling the browser "this file has not changed since the last time you requested it." If a client gets a 304 Not Modified message, then it is the client's responsibility to display the resource in question from its own cache. Since Smart ClickBot does not cache any information it usually gets only HTTP 200 response code. HTTP 200 is telling the browser "here is a successful response," which should be returned when it is either the first time your browser is accessing the file or the first time a modified copy is being accessed. Table 7.6 shows the differences in HTTP 200 and HTTP 304 response percentages.

(k) Robots.txt file request

Table 7.6: Percentage of Bot vs. non-Bot HTTP response codes

Response code	Percentage in Bot traffic	Percentage in non-Bot traffic
200 - OK	97.0%	10.3%
304 - Not Modified	1.7%	80%
307 - Moved Temporarily	0%	2.9%
404 - Not Found	1.2%	6.7%
500 - Internal Server Error	0.1%	0.2%

The Robot Exclusion Standard, also known as the Robots Exclusion Protocol or robots.txt protocol, is a convention to prevent cooperating web crawlers and other web robots from accessing all or part of a website which is otherwise publicly viewable. If a site owner wishes to give instructions to web robots they must place a text file called robots.txt in the root of the web site hierarchy such as www.thebestmusicsites.org/robots.txt. Robots that choose to follow the instructions try to fetch this file and read the instructions before fetching any other file from the web site. If this file does not exist web robots assume that the web owner wishes to provide no specific instructions.

We made available the robot.txt file for the experimented website. If a request to the robots.txt file was made during a session, we consider it as a strong evidence to believe that the session belongs to a bot. However because compliance to the Robot Exclusion standard is voluntary, and many robots simply do not follow the proposed standard, we can not totally rely on this criteria to detect Web robots.

(l) Distribution of countries

Table 7.7 lists the top 20 countries that Smart ClickBot uses to generate IP diversity. Even though highest number of proxy IPs are from the US, there is a large amount proxy IPs recorded from Indonesia, Brazil, China, and India.

Collectively their traffic is larger than the US alone. Since we have used an experimental website to collect data, at this time we do not have enough traffic to find out its actual(non-Robot) country distribution. Therefore, we did not include this as a feature, even though we will use it as soon as the information is available.

Table 7.7: Distribution of countries

Country	Requests	Visitors	Country	Requests	Visitors
United States	4249	824	Korea. Republic of.	423	81
Indonesia	3105	822	Spain	407	94
Brazil	2224	409	Austria	363	36
China	2148	367	South Africa	287	35
India	1121	85	Puerto Rico	271	35
Chile	912	63	Kenya	267	108
Vietnam	865	133	Croatia	262	36
Thailand	841	162	Singapore	256	48
Venezuela	618	106	Turkey	255	69
Russian Federation	429	216	Czech Republic	250	40

7.1.3 Classification of Bot traffic

After deriving the session features, classification models are built using the Bayesian Networks. We adopted the Bayesian approach due to many successful similar research that are reported in the literature [Strayer et al., 2006, Strayer et al., 2008, Kondo and Sato, 2007].

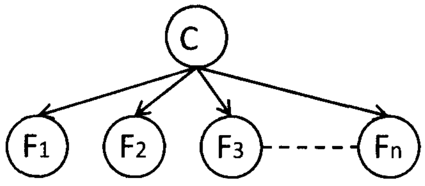


Figure 7.6: Bayesian Network as a classifier.

Bayesian Networks [Friedman et al., 1997, Pearl, 1988] are directed acyclic graphs in which the nodes represent multi-valued variables, comprising a collection of mutually exclusive and exhaustive hypotheses. The arcs signify direct dependencies between the linked variables and the direction of the arcs is from causes to effects. The strengths of these dependencies are quantified by conditional probabilities. More specifically, each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node, where $Parents(X_i)$ denotes the parent variables of X_i. This conditional probability distribution, which defines the conditional probability table of the variable, describes the probability distribution of the variable for each configuration of its parents. The graph encodes that each node is conditionally independent of its non-descendants, given its parents [Friedman et al., 1997].

Naive Bayes is a special case of a Bayesian network, where a single cause (the class) directly influences a number of effects (the features) and the cause variable has no parents. This network structure is shown in Figure 7.6. Again, the independence assumption encoded by this model is that each feature is conditionally independent given the class value.

Considering Figure 7.6, assume that $F_1, F_2, .., F_n$ are n features and f_i represents the value of feature F_i. Assume also that C is the class variable and let c represent a possible value (label) of C. Using Bayes rule and the conditional independence assumption, we can derive the posterior probability of each class label $c \in C$, i.e. the probability of the class label given the features observed, to be given by the formula:

$$P(c|f_1, f_2, ..., f_n) = \frac{P(c)\prod_{n=1}^{n} P(f_i|c)}{P(f1, f2, ..., fn)} \tag{7.1}$$

The class variable C is assigned the label that gives the maximum posterior probability given the features observed. More specifically:

$$class = argmax_{c \in C} P(c) \prod_{i=1}^{n} P(f_i|c) \tag{7.2}$$

For Smart ClickBot detection we used one, similar to Bayesian Network structure shown in Figure 7.6. Each child node corresponds to one of the features we presented earlier in section 7.1.2(2). The root node represents the *class* variable.

In the following section we present the experiments performed in order to apply our methodology and evaluate the performance of the Smart ClickBot detection system.

7.1.4 Experimental Results and Discussion

There are two main objectives in this experiment:(a) to use features described in section 7.1.2(2) to identify as many Smart ClickBot sessions as possible, (b) to find a good model for predicting Smart ClickBot sessions based upon their access features.

We have collected 22991 server side clicks during the period from 3/3/2011 to 3/9/2011. A human expert has labelled the entire data set so that it can be used for model evaluation. To build the classification model we have used the first 5000 samples. In this training data set there were 4501 Smart ClickBot (Class 1) clicks and 499 non-Smart ClickBot (Class 2) clicks. Since class representation is not balanced, training the model was a challenging task. If a model is built using an imbalanced dataset, its characteristics tend to be biased towards the majority class. Especially with the Naive Bayes classifier, the prior probability in the majority class overshadows the differences that exist in the conditional probability entries that quantify the relationship between feature and class variables [Stassopoulou and Dikaiakos, 2009].

There are a few ways to compensate the imbalanced class distribution. We can use techniques such as bagging and boosting or resampling. We used resampling as it was used successfully in a similar study discussed in [Stassopoulou and Dikaiakos, 2009]. Resampling modifies the prior probabilities of the majority and minority class by changing the records on each of the two classes. For this purpose we have used both random over sampling and random under sampling. Over sampling is used with the minority class (Class 2), while Under sampling is used with majority class (Class 1). Table 7.8 shows the resampled data that we have used to build the classifiers($C_1, C_2, C_3, C_4, and\ C_5$) along with the new prior probability distributions. C_1 is built with the original data set without resampling. Both C_2 and C_3 are built with oversampling Class 2, while C_4 and C_5 are built with under sampling Class 1.

Table 7.8: Training data set configuration

Data Set	Classifier	Class 1	Class 2	Prior Probabilities
1	C_1	4501	499	(0.90, 0.10)
2	C_2	4501	4501	(0.50, 0.50)
3	C_3	4501	899	(0.83, 0.17)
4	C_4	2436	499	(0.83, 0.17)
5	C_5	499	499	(0.50, 0.50)

We have tested the five Bayesian classifiers with the rest of the 17991 records. For the evaluation purposes "Accuracy" is a reasonable metric but it has the underline assumption that the data set remains evenly distributed i.e. between Class 1(Smart ClickBot), and Class 2(non-Smart ClickBot). When equal class distribution is not present we can use *Precision* and *Recall* to compare the models.

$$Precision(p) = \frac{no.\ of\ SmartClickBot\ sessions\ found\ correctly}{total\ no.\ of\ predicted\ SmartClickBot\ sessions} \tag{7.3}$$

$$Recall(r) = \frac{no.\ of\ SmartClickBot\ sessions\ found\ correctly}{total\ no.\ of\ actual\ SmartClickBot\ sessions} \tag{7.4}$$

A classifier that assigns the value 1 to every session will have perfect recall but poor precision. In practice, the two metrics are often summarized into a single value, called the F_1-measure [Tan and Kumar, 2002].

The F_1 score can be interpreted as a weighted average of the precision and recall. It summarizes the two metrics into a single value, in a way that both metrics are given equal importance. Recall and precision should therefore be close to each other, otherwise the F_1-measure yields a value closer to the smaller of the two. F_1 score reaches its best value

Table 7.9: Training data set configuration

Classifier	Precision	Recall	F_1measure
C_1	0.921	0.824	0.869
C_2	0.889	0.943	0.915
C_3	0.939	0.827	0.879
C_4	0.964	0.834	0.894
C_5	0.834	0.953	0.889

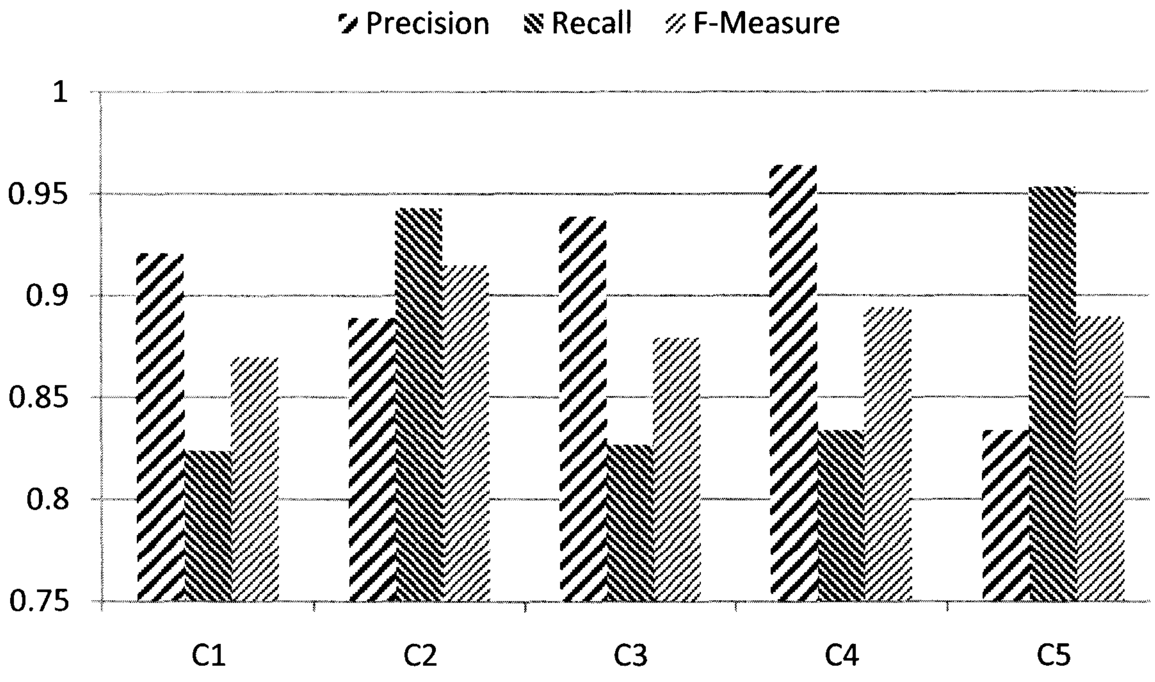


Figure 7.7: Comparison of Bayesian Classifiers.

at 1 and worst score at 0. Table 7.9 and Figure 7.7 show the Precision, Recall, and F_1 measure obtained by the five classifiers.

All the Bayesian classifiers that we have tested achieved a precision of above 83% and a recall of above 82%. Minimum F_1 measure of 87% was reported by C_1, where the training is done with the original data set without resampling. The obvious reason for a low F_1 measure is the class imbalance, where the prior probability of a session to be Smart ClickBot is as high as 0.90. A model trained with such a training data set is biased towards Class 1.

By over sampling Class 2 records for C_3, and by under sampling Class 1 records for C_4 we have achieved a higher F_1-measure than that of C_1. In both of these cases we have tried to increase the class representation of Class 2 records.

The best F_1 measure is achieved by C_2 which was trained using oversampling of Class 2 so that samples reach the number of Class 1 in the original set, hence leaving balanced representation of classes.

In this experiment we have two cases where the class representation is equal. That is with classifiers C_2 and C_5. Training data for C_2 is created by oversampling Class 2 records, while training data for C_5 is created by under sampling Class 1 records. A similar recall values are reported by both C_2 and C_5. However, there is a significant difference between the precision values. Precision of C_5 is 83%, while that of C_2 is 89%. This means we have an increase in the number of false positives in C_5, i.e. Class 1 incorrectly classified as Class 2. The significant decrease in precision of C_5, is not surprising since, with random under sampling there is no control over which examples are eliminated from the original set. Therefore significant information about the decision boundary between the two classes may be lost. This is always a risk with random oversampling where it would do over-fitting due to placing exact duplicates of minority examples from the original set and thus making the classifier biased by "remembering" examples that were seen many times.

7.2 Fast Detection of Duplicates

An important issue in defending click fraud is how to deal with duplicate clicks. If we simply consider all identical clicks as fraudulent clicks, it is unfair to advertisers in some scenarios such as that an interested client visits the same ad link several times a day. On the other hand, if the advertisers are charged for any identical clicks, then it is very easy for an attacker to make money by continuously clicking the same ad link. However, it is very difficult to identify which scenario the identical clicks belong to. A reasonable countermeasure is to prescribe that identical clicks will not count if they are within short time interval, and will count if they happen sparsely. Therefore, a feasible duplicate detecting algorithm should have a mechanism that is able to eliminate unrelated (or expired) information.

On the other hand, users of a practical click fraud detection solution expect these duplicate detection mechanisms to run in realtime. In order to provide real time results, solution providers should utilize a data structure that can be updated in real time. In addition, If the actual volume of clicks (per unit time) is high, space requirements per click should be the lowest possible thus data structures with constant space requirements or sublinear space requirements such as O(log(n)) or O(log(log(n))) are desirable (n is the number of elements processed).

In this section, we consider the problem of detecting duplicates in click data streams. Our solution uses a modified version of the Counting Bloom Filter. The Temporal Stateful Bloom Filter (TSBF) extends the standard Counting Bloom Filter by replacing the bit-vector with an array of counters of states. These counters are dynamic and decay with time.

7.2.1 Data Stream Model

We consider a data stream, including click stream, as a sequence of numbers, denoted by $C_N = x_1, x_2, x_3, ..., x_N$, where N can be infinite, which means that the stream is not bounded. In general, a stream can be a sequence of records, but it is not hard to transform each record to a number and use this stream model.

Our problem can be stated as follows: given a click data stream C_N and a certain amount of memory space, M, estimate whether each element x_i in S_N appears in $x_1, x_2, x_3, ..., x_N$ or not. Since our assumption is that M is not large enough to store all distinct elements in $x_1, x_2, x_3, ..., x_N$, there is no way to solve the problem precisely. Our goal is to approximate the answer and minimize the false positives, where false positive is a distinct element wrongly reported as a duplicate.

To address this problem we examine two techniques that have been previously used in different contexts, namely the Buffering method and Bloom filters [Song et al., 2005, Talbot and Osborne, 2007, Cuenca-Acuna and Nguyen, 2010].

7.2.2 Buffering Methods for duplicate detection

A straightforward solution, to detect duplicates, is to allocate a buffer and fill the buffer with enough elements of the stream. For each new element, the buffer can be checked, and the element may be identified as a distinct if it is not found in the buffer, and as a duplicate otherwise. Buffer can be implemented in many different ways including hash table approaches. Traditional approaches for duplicate detection using hash table based solutions are discussed in [Elmagarmid et al., 2007].

A hash table is made up of two arrays. First, the actual table where the data to be searched is stored, and second, a set of mapping functions known as hash functions is stored. The hash function is a mapping from the input space to the integer space that defines the indices of the array. In other words, the hash function provides a way for assigning numbers to the input data such that the data can then be stored at the array index corresponding to the assigned number. We will explain this further with an example related to Figure 7.8, that shows a hash table designed to map IP addresses.

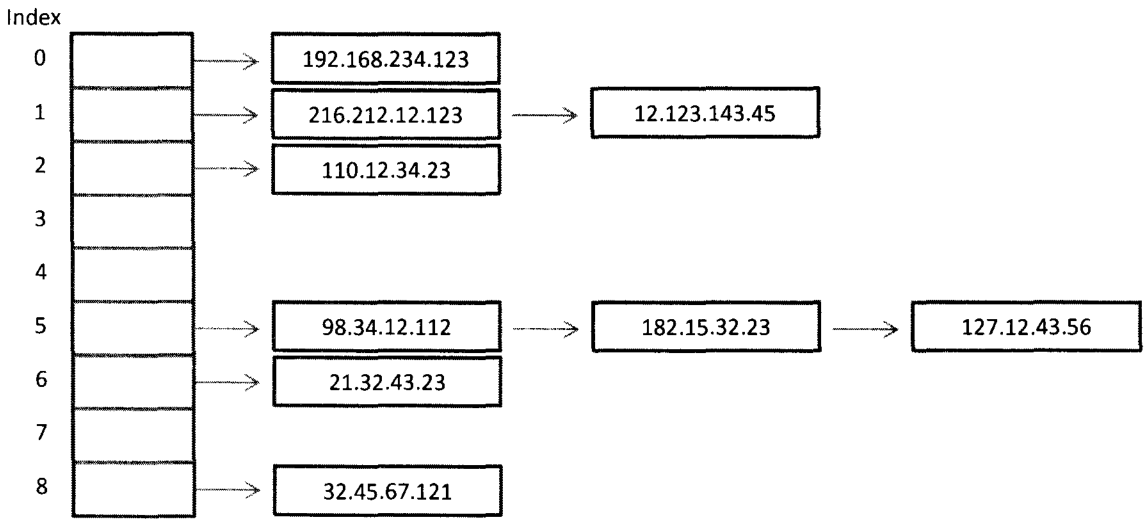


Figure 7.8: Storage of IP addresses in a Hash table.

First, we start with a hash table array of strings (we'll use strings as the data being stored and searched in this example). Let's say the hash table size is 9. In order to insert the IP address "98.34.12.112" we run "98.34.12.112" through the hash function, and find that hash("98.34.12.112", 9) yields 5. We insert IP "98.34.12.112" into the 5th index of the hash table. If the same IP address is visited again, since the hash table already contains the IP. then a signal for a duplicate will be returned.

Since the indices of the output range is a predefined number (in this case 9), different IP addresses will be mapped to the same index. In order to hold more than one item in one location the strings are stored as a *linked list*. This is often called *chaining*. Chained hash tables have the disadvantages of linked lists. It can be wasteful on memory, as many of array positions might contain empty linked lists. When storing small keys and values, the space overhead of the next pointer in each entry record can be significant. An additional disadvantage is that traversing a linked list has poor cache performance, making the processor cache ineffective. The biggest disadvantage is the cost of the linked list manipulation when strings are associated with expiring information. For example update of IP addresses which are older than two minutes.

In this method, each distinct element will be mapped to a new location and it is inevitable that size grows tremendously if the stream carries too many distinct elements. Therefore this approach can be directly applied only to applications where there are a few distinct elements. For series with more duplicates, such as IP number stream, a proper replacement mechanism should be utilized. This mechanism will evict the least important element from the buffer and replace with the new element. When the buffer is full, a newly arrived element may replace another element out of the buffer before it is stored. Broder et al. [Broder et al., 2003] discussed 5 different replacement policies which are briefly discussed here.

(a) *Clairvoyant (MIN)*: This method was introduced by Belady et al. [Belady, 2010]. They showed that if the entire sequence of requests is known in advance, then the best strategy is to evict the item, which is the one whose next request is farthest away in time.

(b) *Least Recently Used (LRU)*: The LRU algorithm evicts the item that has not been requested for the longest time. The intuition for the LRU is that an item that has not been needed for a long time in the past will likely not be needed for a long time in the future.

(c) *CLOCK*: CLOCK is a popular approximation of LRU. It was invented by Cobato et al.[Corbato, 1968]. An array of mark bits $M_0, M_1, ..., M_k$ corresponds to the items currently in the buffer of size k. The array is viewed as a circle, that is, the first location follows the last. A clock handle points to one item in the buffer. When a request X arrives, if the item X is in the buffer, then its mark bit is turned on. Otherwise, the handle moves sequentially through the array, turning the mark bits off, until an unmarked location is evicted and replaced by X.

(d) *Random Replacement*: Random replacement completely ignores the past. If the item requested is not in the buffer, then a random item from the buffer is evicted and replaced.

(e) *Static*: In this method, it is assumed that each item has a certain fixed probability of being requested. This probability is independent of the previous history of requests. An item is evicted, at any point in time to maximize the probability of an item found in the buffer.

We chose LRU mechanism as the replacement policy for Buffer implementation due to the fact that newly arrived clicks have least impact on oldest clicks. We will use this method in our experiments and compare its performance with the Bloom Filter based methods introduced in Section 7.2.4 and the proposed method which will be introduced in Section 7.2.5.

7.2.3 Bloom Filters for Duplicate Detection

Even with a replacement mechanism Buffering method requires enormous space because each element has to be stored. Therefore it is not suitable for applications with stream data. To overcome this drawback, we have researched a closely associated data structure known as *Bloom Filters*[Bloom, 1970]. Bloom filters have been used extensively in networking applications (see [Reynolds and Vahdat, 2003, Broder and Mitzenmacher, 2004, Li et al., 2000, Kumar et al., 2006, Rowstron and Druschel, 2001, Fan et al., 2000], and [Estan and Varghese, 2003]) because they enable both high speed and low cost implementation of various hardware algorithms. A Bloom filter is essentially a compact representation of a set. Standard exact representations of sets such as hash tables and binary trees requires at least L bits per element to be stored, where L is the size of the element, and often requires additional space for pointers. By contrast, a Bloom filter is an inexact representation of a set that allows a few "false positives" when queried. In return, it allows very compact storage: roughly 10 bits per element for a 1% false positive probability. It also has a constant space requirement, which is independent of the size of the element in the set or the size of the set itself [Bonomi et al., 2006].

Inspired by these properties, we have developed a solution to detect duplicates in a click stream, based on modified Bloom filters and it is now a part of our real time click fraud detection and prevention solution available at http://www.netmosaics.com. Implementation details of the system are discussed in [Kantardzic et al., 2010b], and [Walgampaya et al., 2010]. This work is primarily motivated by the tradeoff between space usage and accuracy of the existing duplicate detection systems. In section 7.2.4 we start with a brief discussion about the Bloom filter and its variants including Counting Bloom Filters and Temporal Stateful Bloom Filters. Experimental results are given in section 7.2.5, while related work are discussed in section 7.2.6.

7.2.4 Bloom Filter and Its Variants

(a) Classical Bloom Filter Bloom filter (BF) was proposed by Burton Bloom in 1970 and used for space-efficient data structures that maintain a very compact inventory of the underlying data, supporting membership queries over a given data set [Bloom, 1970]. The space requirements of Bloom Filters fall significantly below the information theoretic lower bounds for error-free data structures. This efficiency is at the cost of a small false positive rate (items not in the set have a small probability of being recognized as in the set), but have no false negative (items in the set are always recognized as being in the set). BFs are widely used in practice when storage is at a premium and an occasional false positive is tolerable [Cheng et al., 2005].

A BF is a bit array, v of size m, and all of which are initially set to 0 (See Figure 1(a)). For each element(key 1, key 2, etc.), k bits in BF are set to 1 by a set of hash functions $\{h_1(x), h_2(x), h_3(x), ..., h_k(x)\}$, all for which are assumed to be independently uniform. It is possible that one bit in BF is set multiple times, while only the first setting operation changes 0 into 1, and the rest have no effect on that bit (Figure 1(b) and 1(c)). To know whether a newly arrived element x_i has seen before, we can check the bits $\{h_1(x_i), h_2(x_i), h_3(x_i), ..., h_k(x_i)\}$. If any one of these bits is zero, with 100% confidence we know x_i is a distinct element. Otherwise, it is regarded as a duplicate with a certain probability of error. An error may occur because it is possible that the cells $\{h_1(x_i), h_2(x_i), h_3(x_i), ..., h_k(x_i)\}$ are set before by elements other than x_i [Deng and Rafiei, 2006].

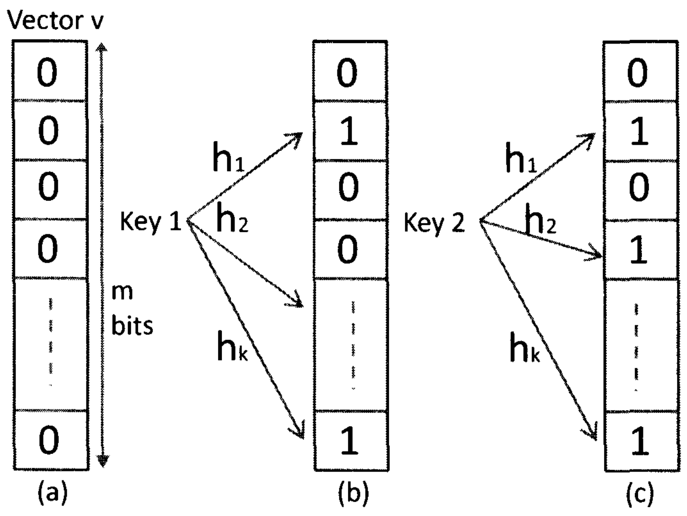


Figure 7.9: A Classical Bloom Filter

The salient feature of BFs is that there is a clear tradeoff between m and the probability of a false positive, p_{error}. Observe that after inserting n keys into a table of size m, with k number of hash functions, the probability that a particular bit is still 0 is exactly

$$(1-\frac{1}{m})^{kn} \tag{7.5}$$

Hence the probability of a false positive, p_{error}, in this situation is

$$p_{error} = (1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{\frac{-kn}{m}})^k \tag{7.6}$$

For a given m/n we can find the minimum p_{error}:

$$\begin{aligned}\frac{\partial p_{error}}{\partial k} = e^{-\frac{k^2 n}{m}} \left(e^{\frac{kn}{m}}-1\right)^k \log\left(e^{\frac{kn}{m}}-1\right) - \\ \frac{kn}{m}\left(e^{\frac{kn}{m}}-1\right)^{k-1}\left(e^{\frac{kn}{m}}-2\right)\end{aligned} \tag{7.7}$$

The right hand side is minimized for $k = \frac{m}{n}ln2$, in which case it becomes

$$(\frac{1}{2})^k = (0.6185)^{\frac{m}{n}} \tag{7.8}$$

Some example theoretical values for p_{error} for combination of m/n and k values are given in Table 7.10 and Figure 7.10 shows the graph for variation of p_{error} with m/n and k.

Obviously, p_{error} is at a minimum when $m/n \geq 20$ for any value of k. But higher m/n yields sparse Bloom filter, which indeed waste a lot of memory . Therefore, now it becomes a tradeoff between m/n, and k, when selecting the best possible configuration. What really means by m/n? It actually indicates number of vacant cells in the Bloom filter. When m/n is high the bloom filter has more slots to occupy and chance of a false positive will be low. When m/n decreases it indicates that the bloom filter is reaching its full capacity, and chance of a false positives in this case will be high. On the other hand, if the number of hash functions, k is low,

Table 7.10: Variation of p_{error}

m/n	k	p_{error}
4	1	0.2212
4	2	0.1548
6	4	0.0560
8	6	0.0216
12	8	0.0031
16	10	0.0004
16	11	0.0004

and if m/n is low then there is a high chance for a false positive. But in the same case when m/n increases then p_{error} decreases. Since n, which is the number of elements present in the Bloom filter at a given time depends on the application, if we want a bigger m/n, the only possibility is to increase the size of the Bloom filter (m). But then this will lead to unnecessary memory wastage. Therefore it is at utmost importance to define the ideal Bloom filter configuration for any practical application. In section 7.2.6, we discus in detail the procedure of selecting an ideal values for m and k for our application.

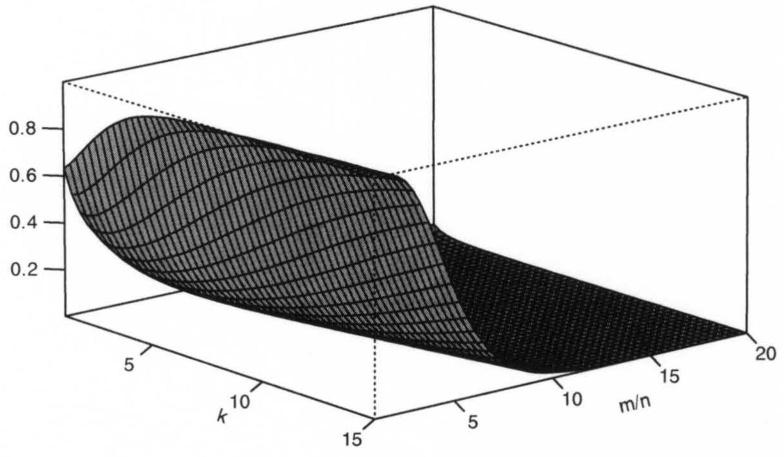


Figure 7.10: Variation of p_{error} with m/n and k.

Although BF is simple and space efficient, it does not allow deletions. Deleting elements from a BF cannot be done simply by changing them back to zeros, as a single bit may correspond to multiple elements. Therefore, in the data stream environment, if we apply BF for detecting duplicates, when more and more new elements arrive, the fractions of zeros in BF will decrease continuously, and the false positive rate will increase accordingly, and finally reach the limit one. At this time every distinct element will be reported as duplicate, indicating that BF fails completely. We call such a state of the BF as "full" [Shen and Zhang, 2008]. For the purpose of allowing deletion of stale elements, *Counting Bloom Filter* (CBF) was proposed by Fan et. al.[Fan et al., 2000]. Following section describes the functionality of CBF that enables deletion of elements.

(b) Counting Bloom Filter

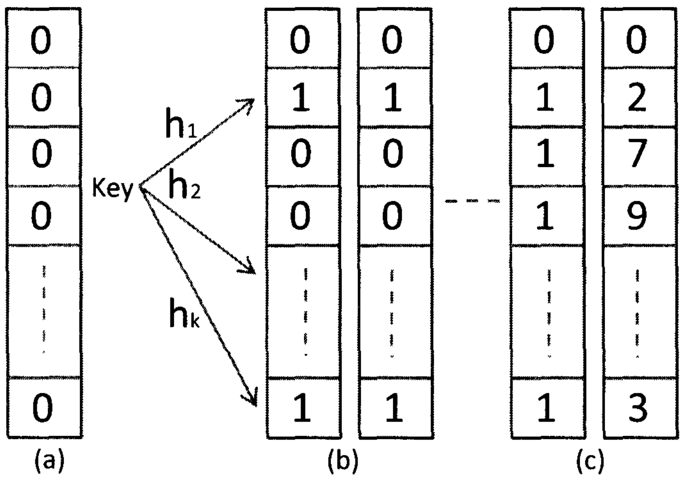


Figure 7.11: Counting Bloom Filter

A CBF uses an array of m counters, C, which replaces the bit vector v in the BF (see Figure 7.11(b)). The counters in C represent multiplicities of elements; all the counters in C are initially set to 0. When inserting an item, we increase the counters by 1 as shown in Figure 7.11(c). Also, deletion can now be safely performed with decrementing the counters by 1. A BF can be derived from a CBF by setting all non-zero counters to one. Size of the counters must be chosen large enough to avoid overflow although from the following proof we can safely limit the size of counters to 4 bits [Gonnet and Baeza-Yates, 1991].

The asymptotic expected maximum count after inserting n keys with k hash functions into a bit array of size m is:

$$\Gamma^{-1}(m)(1+\frac{ln(\frac{kn}{m})}{ln\Gamma^{-1}(m)}+O(\frac{1}{ln^2\Gamma^{-1}(m)})) \tag{7.9}$$

and the probability that any count, c, is greater or equal to i is:

$$Pr(max(c) \geq i) \leq m\binom{nk}{i}\frac{1}{m^i} \leq m(\frac{enk}{im})^i \tag{7.10}$$

From equation (7.10), the optimum value for k is $ln2\frac{m}{n}$, assuming $k < ln2\frac{m}{n}$, we have,

$$Pr(max(c) \geq i) \leq m(\frac{eln2}{i})^i \tag{7.11}$$

Table 7.11 shows some sample values for the variation of probability that a counter is greater than i bits.

Table 7.11: Variation of p_{error}

Bits Per Count	$Pr(max(c)>$i)
4	1.368 x 10^{-15} x m
3	9.468 x 10^{-06} x m
2	4.923 x 10^{-02} x m
1	1.884 x m

Therefore, for any practical value of m, with 4 bits per counter the chances of overflow will be very small [Fan et al., 2000].

Selecting the deletion mechanism for CBF sometimes depends on the practical application. Because the definition of expired data in one application may not be suitable for another application. For example, in a click duplicate detection application, two clicks are considered duplicates if they occur within a threshold time interval. If not, they are considered distinct. Therefore, at the end of the threshold interval counters should be adjusted. If x is the threshold time to decide if two clicks from the same IP are duplicates, at every x minutes counters can be decremented by 1. In this case the deletion mechanism can be represented as a step function. There are numerous other decaying functions discussed in the literature. Following section discusses few other popular decaying functions and their properties.

(a) Time Decaying Counters *Time sensitivity* of data is important in many traditional and emerging applications. For example in medical applications current data may have much weight to the decisions taken while in web tracking and personalization applications recommendations are generated considering the recent history. However, in many of these applications, older data items are less significant than more recent ones, and thus should contribute less or not at all to current decisions. This is because the characteristics or "state" of the data generator may change over time, and, for an application such as prediction of future behavior or resource allocation, the most recent behavior should be given a larger weight [Cohen and Strauss, 2006].

    Time sensitivity can be formalized in a variety of ways. We may only consider elements that fall within a *sliding window* of recent time (last one hour), and ignore (assign zero weight) any that are older; or, more generally, use some arbitrary function f that assigns a weight to each element as a function of its age [Zhang et al., 2009]. In this research we have adapted the later method where counters in the CBF are defined using a time decaying function.

    The manner of how a counter decays with time is determined by a special non-increasing, non-negative function, called the *time decaying function* (tdf). Any tdf should satisfy the following conditions:

    i. $\phi(0) = 1$

ii. $\phi(t)$ is non-increasing

iii. $0 \leq \phi(t) \leq 1$ for all $t \geq 0$

Figure 7.12 shows three different tdfs, exponential, linear, and step. Since our application investigates whether a new click is a duplicate or not (nothing in between) the counters should not keep residues over time. Therefore our counters are implemented using the step time decaying function. It decrements every non zero counter by 1 at every predefined time interval(t_0). t_0 is defined as the time threshold where two clicks from the same IP are not considered as duplicates if they occur closely.

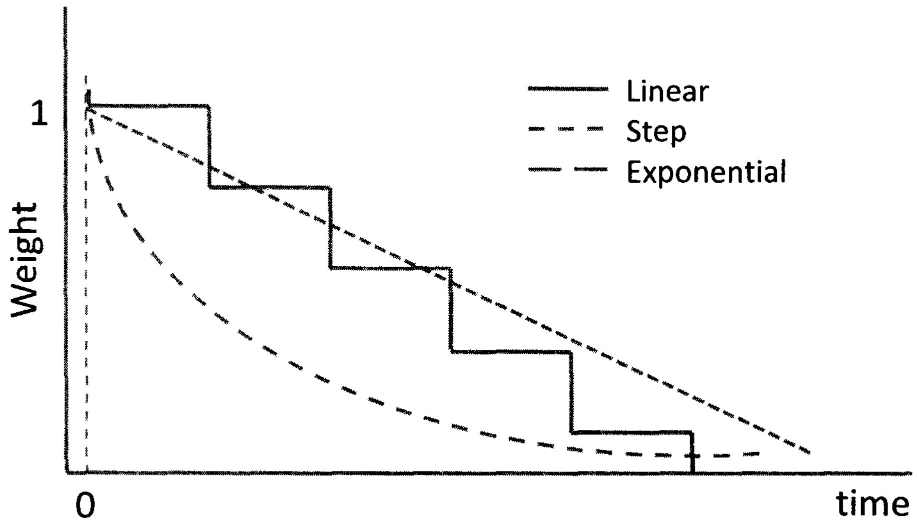


Figure 7.12: Decay functions.

Even though CBF provides a very simple approach to count the duplicates in a stream, we cannot directly use it because of the following reason. Consider the five scenarios shown in Figure 7.13. Horizontal axis corresponds to time and indicators on top of the time axis corresponds to deletions from the CBF, while indicators in the bottom of the time axis corresponds to insertions into the CBF. For simplicity we assume deletions occur at regular time intervals. This is shown with equal gaps between deletion indicators in Figure 7.13. On the other hand insertions occur randomly, which is the usual case in practical applications. If two clicks land between two deletions, we consider them to be duplicates. Therefore case 1, and case 2 are not considered duplicates but case 3, and case 5 are considered duplicates. If we strictly follow this rule case 4 will be considered as non-duplicates, while case 5 will be considered duplicates, even though both should be considered as duplicates because the time interval between clicks in case 4, and case 5 are miniscule. This scenario occurs due to small time interval δt between consecutive insertions. This situation can be expected more often in click stream data because clicks can occur very close to each other. For example the time between a double click from the same IP address may be very small. We want our system to consider both case 4 and case 5 to be duplicates. Therefore we have slightly modified the ordinary CBF, where it's counters C will contain *status* information instead of counters.

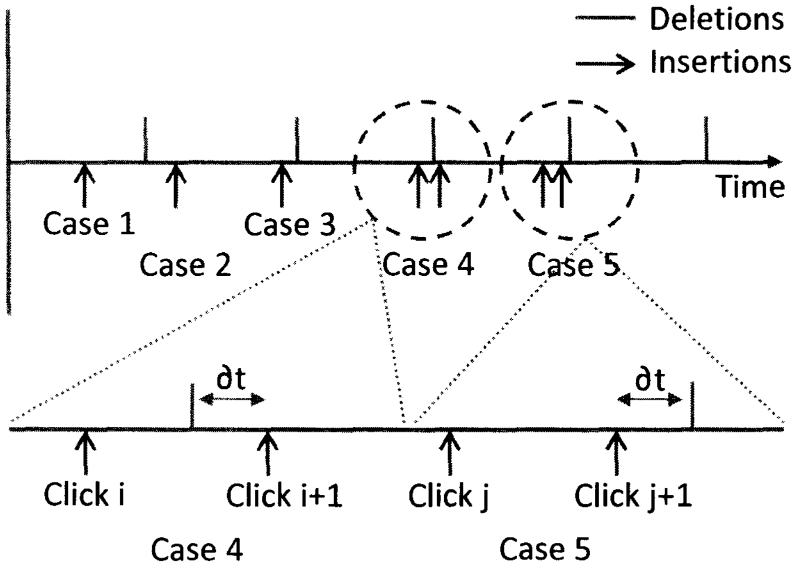


Figure 7.13: Real time Insertions and deletions in a click stream.

(b) Status information We represent the *status* information as an array with the values in the range $S : [1, V]$, where V represents the maximum number of states. This is depicted in Figure 7.14(a). The value V depends on the requirements of the application. Following is an example for the status vector

S, when $V = 3$, in a duplicate clicks detection system, and the unique situations define by each state.

i. 1: Partially deleted, indicates that it is possible the same click has seen prior to two deletions.(It is only probable because the state can be changed by a different click, if it is mapped to the same location)

ii. 2: Recently seen, indicates that the click may have occurred between the new click and the last deletion.

iii. 3: Never seen [4], indicates that either the clicks is never seen or it has expired.

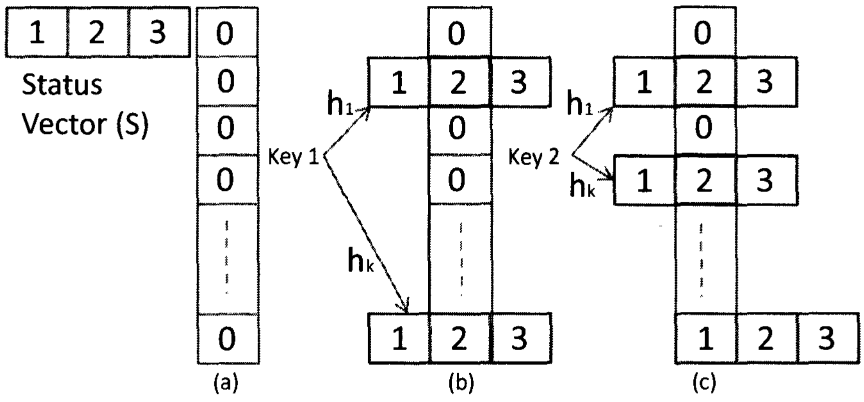


Figure 7.14: Modified Counting Bloom Filter with status information.

(c) Temporal Stateful Bloom Filter In this section, we describe our Temporal Stateful Bloom Filter (TSBF). Again the underline structure is similar to CBF, where counters are decayed with time. TSBF cells are neither bits nor counters but instead a value corresponding to the state. A TSBF consists of,

(a) An array of m counters, $\{C_1, C_2, C_3, ..., C_m\}$, where each counter C_i is replaced with the status vector, S.

[4]There is no difference between having 0 and having 3 in the bloom filter except that when it is 0, we know that cell is never touched.

(b) A set of independent k hash functions $\{h_1, h_2, h_3, ..., h_k\}$ defined for the range $[1..m]$

(c) A time decaying step function $\phi(t)$

The TSBF carries out the following *insertion*, *lookup* and *modify* tasks.

(a) *Insertion*: If the cell counter is 0, set the count to 2. If the cell value equals 1, set the count to 2 (Figure 7.14(b)).

(b) *Lookup*: Check all cells associated with the insertion. If all cell values are either 1 or 2, then it is a duplicate.

(c) *Modify*: When decaying function $\phi(t)$ decreased to the threshold level adjust the cells accordingly. If the current status value is 2, change the cell value to 1. If the current status value is 1, change the cell value to 3. This is illustrated in Figure 7.14(c).

We call the first and second operations the duplicate detection process, and the third operation the update process.

7.2.5 Experimental Results and Discussion

In order to evaluate the proposed methodology for duplicate detection in click streams, we ran a comprehensive set of experiments, especially because it is now a part of a commercial system,www.NetMosaics.com, that provides solutions for click fraud. The experiments used both synthetic and actual data. Actual click stream data are collected in the servers at the www.NetMosaics.com. Each click record contains several server side and client side parameters, as well as user activities during the each session.

The integrated structure of a click record which includes all context information is shown in Figure 7.15. There are two unique identifiers available for each click. The IP and the TrackingID. IP is the IP address of the origin of the click. TrackingID is an 128 bit long globally unique identifier generated by the NetMosaics servers. We have used IP as the identifier for each click, but the 128 bit tracking number can also be used instead.

Server Side IP	Server Side tracking ID	Other Server Side data	Client Side IP	Client Side tracking ID	Other Client Side data	Context data

Figure 7.15: Structure of the click record.

When a new click is arrived we first calculate the corresponding hash value of the IP address. The hash functions are developed by first calculating the MD5 signature[5] of the IP address [Menezes et al., 1997], which yields 128 bits, then dividing the 128 bits into m/k number of groups, and taking the modulus of each m/k bit word by the table size m.

Initially, all counters in the TSBF are set to zero. When using TSBF to detect duplicate clicks, for each newly arrived click, we execute the following three operations in order.

(a) The IP is mapped to k counters by some uniform and independent hash functions.

(b) Change the status value in the hashed locations by according to the *Insertion* function defined above. If the IP is a duplicate, it can be determined by probing whether all the k counters, hashed to are all $1s$ or $2s$.

(c) At a predefined time interval t_0 change the status of all the non-zero cells according to the *Modify* function defined above.

7.2.6 Accuracy of the Bloom Filter

Ordinary Bloom filters are discussed in literature for duplicate detection in streaming data [Shen and Zhang, 2008, Deng and Rafiei, 2006, Metwally et al., 2005b]. Since the

[5]also called as MD5 hash. It is a 128 bits long number. It is calculated from the contents of the string being read. Once the entire string is read, the bytes combined numerically via a special algorithm and the result is the MD5 hash. The algorithm for the calculation of that number is designed to be relatively quick to compute, and, perhaps more importantly, very unique.

architecture of the proposed TSBF is based on ordinary Bloom filter we have experimented BF on a streaming data series. There are three parameters that define the performance of a BF, i.e. m, n, and k.

The streaming data series is a set of unique IP addresses that are generated randomly. This will enable to find out the variation of true positives in a BF, which is the only type of error found in BF. True positives are always created by unique elements, where it is reported as a duplicate. We have selected $m = 4096$, and varied both sample size (n) and number of hash functions used (k). We have varied n from 100 to 3000. Figure 7.16 shows the variation of false positives. Legend shows the variation of the number of hash functions.

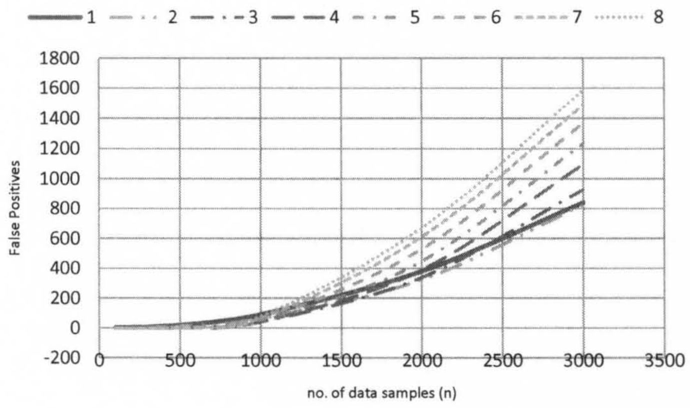


Figure 7.16: False positives in BF with m = 4096.

All the bits in the BF are initially set to zero. As new elements arrive they are mapped to proper indexes by k hash functions and the corresponding bit is turned to 1. Since BF does not have a deletion mechanism, as new elements arrived, the percentage of zeros will decrease. This will cause false positive rates to increase. In Figure 7.16, when the number of samples reached about 1000, the false positives increases drastically. For example when $n = 1000$, there are 70 false positives, with $k = 8$. i.e. an error rate of almost 7%. If we use 3 hash functions instead of 8 the error rate reduced to 3%, which is the minimum achievable at this settings. If somebody wants to operate at this level then a deletion mechanism should be executed at this level in order to maintain the percentage of $0s$ in the BF. Otherwise, for smaller error rates, a BF with different set of parameters should be chosen. Figure 7.17 shows BF configurations with lesser error rates. Legend shows the variation of the number of hash functions.

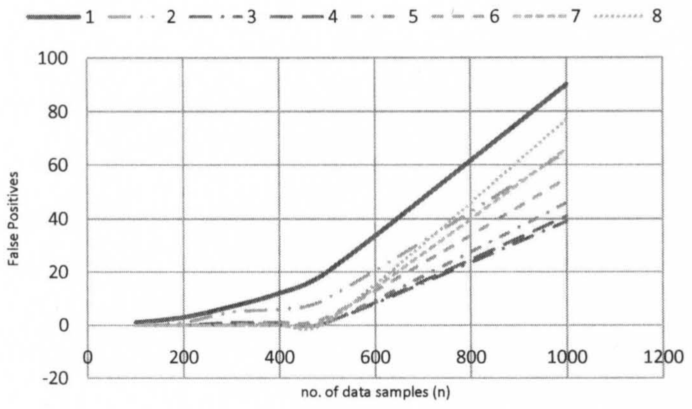


Figure 7.17: BF configurations with at most 10% False positives.

In Figure 7.17 despite the value of k, false positives increase when number of incoming samples goes beyond 500. In this region except $k = 1$, and $k = 2$ false positive rate is less than 1%. Therefore, for the rest of the experiments whenever we configure a BF k is set to 4.

If only four states(i.e. $v = 4$) are used in the TSBF, only 2 bits are required to present all the information. In this case, four states can be one in the set $00, 01, 10, 11$. Compare to the ordinary BF this requires as twice as much memory. But ordinary BF does not support deletions, and TSBF is able to support deletions with an addition of one extra bit in each cell. When compared with standard CBF, TSBF requires only half the space or less with the same false positive probability, and it appears as simple or even simpler to put into practice. (In Table 7.11, we have showed that optimum requirement of bits per cell for CBF is 4).

Next set of experiments are conducted using the proposed TSBF approach to compare the performance when detecting duplicates. First, We conducted experiments on streams of synthetic data to illustrate how the theoretical and practical error rates vary with the number of hash functions. Error rate introduced in equation 7.6 is used as an approximation for the theoretical error rate in TSBF. We used a synthetic stream of 100,000 clicks without any duplicates. Since all clicks are *distinct*, any duplicates detected by the TSBF will be erroneous. We have calculated the error rate as the percentage of duplicates detected per total clicks encountered. We have simulated the incoming clicks stream by randomly allowing $100 * n$ (where $1 \leq n \leq 1000$) records to be processed between any deletion period (i.e. before any click expired in the batch). Figure 7.18 shows the results when $n = 1000$(i.e. 1000 clicks are processed before deleting any record). We have seen similar results for all n values of the data sets.

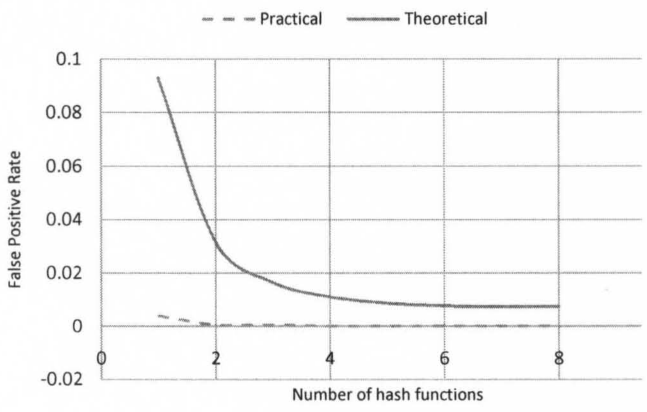


Figure 7.18: Theoretical and Practical error rates with n = 1000.

The graphs in Figure 7.18 shows how the error rate decreases as the number of hash functions increases for the same BF configuration (i.e. for same m). The theoretical error rate of BF shown in equation 7.6 is only an approximation. Therefore, we have experimentally found out whether the assumption is safe to continue the experiments. We have seen that the actual error rate is always less than the theoretical error rate, and it is almost negligeable when we use 4 or more hash functions.

From equation 7.6, we have $p_{error} \approx (1 - e^{\frac{-kn}{m}})^k$. Figure 7.19 shows the variation of p_{error} with the exponent $(\frac{-kn}{m})$. It is required that for better performance of the TSBF either $kn \leq m$ or $k \leq m/n$. This shows clearly the tradeoff between space and error rate in the TSBF. The more hash functions used in TSBF, the larger the required space and the smaller the probability of producing errors. However, since the space usage of TSBF

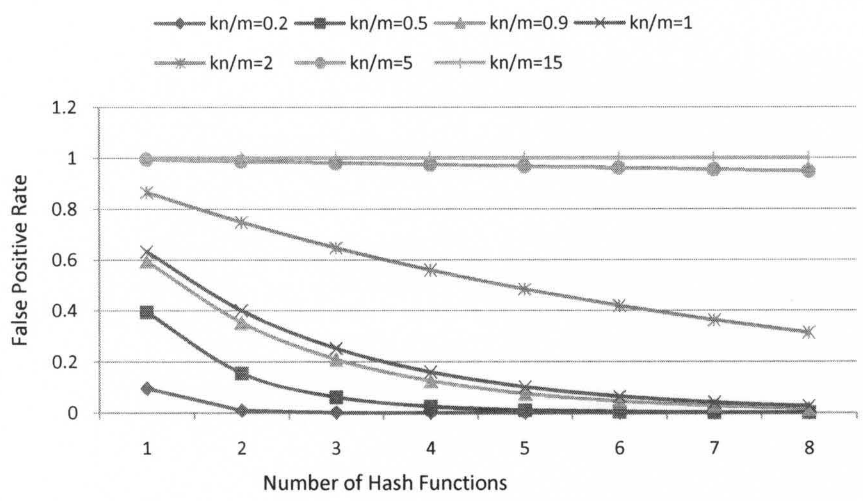


Figure 7.19: False Positive Rate Vs. kn/m.

is a constant, the number of hash functions should be chosen adequately to satisfy the above requirement.

In the next experiment we have looked at the variation of error rate with the space usage(i.e. m/n), which gives us the flexibility to select a better size for m of our TSBF.

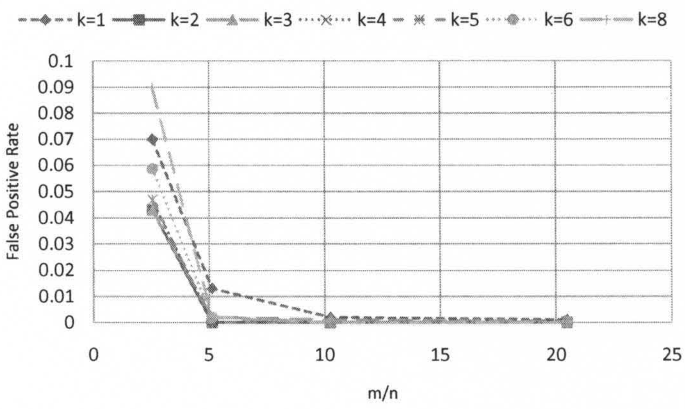


Figure 7.20: m/n vs. False Positive Rate for n=1000.

Figure 7.20 shows the variation of false positive rate with the ratio m/n for the data set size $n = 1000$. When the m/n ratio is less than 5, false positive rate decreases drastically. In this region it is important to see that higher the number of hash functions used, higher the false positive rate is. Even though we expect the error to go down with number of hash functions, due to the compactness of the array, the probability that a counter, in the BF in this area, is 1 is high. Therefore false positive rate increases. This phenomena gradually decreases after m/n is 5. It is almost negligeable when the m/n ratio is greater than 10.

When the TSBF is launched in an actual situation, the size of the BF (m) will be fixed. n which is the approximate number of clicks per unit time is also almost a constant, which varies predictably. For example in a given web server average number of clicks per unit time is usually known, which we refer to as baselines in our previous papers [Walgampaya et al., 2010, Kantardzic et al., 2010b]. Therefore the only parameter we can vary is the number of hash functions. When the value of n is found based on practical experiences, we have to carry out sample runs with different values for m, and k to find the best possible configuration for TSBF. For example Figure 7.21 shows the variation of number of false positives with the number of hash functions, for an approximate incoming click rate of 300 per unit time.

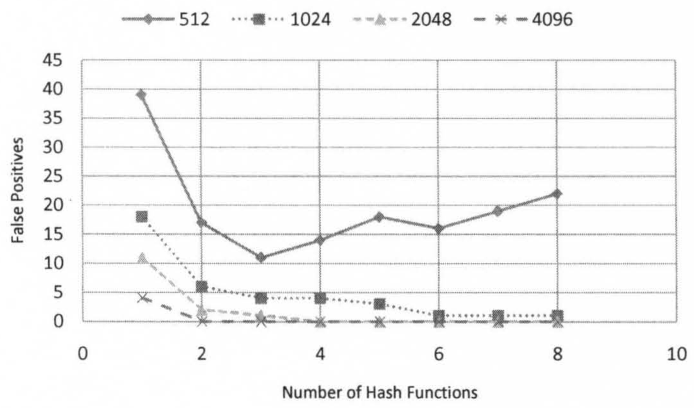


Figure 7.21: False Positives vs. Number of hash functions.

We have varied m with the values 512, 1024, 2048, and 4096 (these numbers are selected only for experimental purposes) in this experiment. In Figure 7.21, when $m = 512$, and $n = 300$, as the number of hash functions(k) increases the number of false positives also increases after $k \geq 3$. But when the BF size(m) grows, the compactness (m/n) of the BF reduces, so does the false positives with increasing number of hash functions.

7.2.7 Experiments with real world data

The data which will be used for the experiments to follow comes from the paid traffic of two very different websites. All of our experiments use click data from Hosting.coms website. Hosting.com is a global company which provides hosting solutions to "business critical data assets" as explained on their website. We also created thebestmusicsites.org, a single webpage only displaying advertisements. It was created to attract fraudulent traffic. Currently we have data for Aug. 2007 to June 2008 for Hosting.com and Jan. 2007 to Aug. 2007 for thebestmusicsites.org. Traffic was filtered before being run through the outlier detection module. First, only paid traffic is being monitored. This includes ads run on search results and network partners of Google, Yahoo, and Bing. Next, we removed all known robots from the traffic. A known robot is a robot that declares itself in the user agent field. Known robots are generally removed from paid traffic before an account is charged.

Duplicate detection is first tested with an ordinary buffering implementation as discussed in Section 7.2.4. We have used a linked list to implement the buffer because of its simplicity. In this experiment we considered two clicks are duplicates if they occurred within a two minute time interval. Accordingly the buffering approach compares the time stamps between IPs for two minute threshold, if an incoming IP is already recorded in the buffer. This threshold time interval is chosen only for experimental purposes, and duplicates detected in more finer intervals are discussed later in the experiments.

Buffering techniques are popular in applications where there are less number of distinct elements in the data. For example if we expect data from only w (for example 5) data sources, then for each incoming record there will be w comparisons in the worst case. Since the number of elements to store are small, the size of the Buffer does not grow beyond w. Therefore, we do not have to delete old elements to make space for new incoming elements. In such situations implementing the duplicate detection using Buffering mechanism may be an easy option. We also tested the Buffering without any deletion or replacement mechanism. We have done this to investigate[6] whether the requests are reaching from the same set of IPs. If the requests always come from a manageable set of IPs our work will be much easier. But we have realized that the size of the buffer grows exponentially immediately after deployment.

Table 7.12 shows the duplicates detected by the buffering mechanism along with the approximate size of the buffer. It also shows the duplicates detected by the proposed method, TSBF, and its constant space requirement over time.

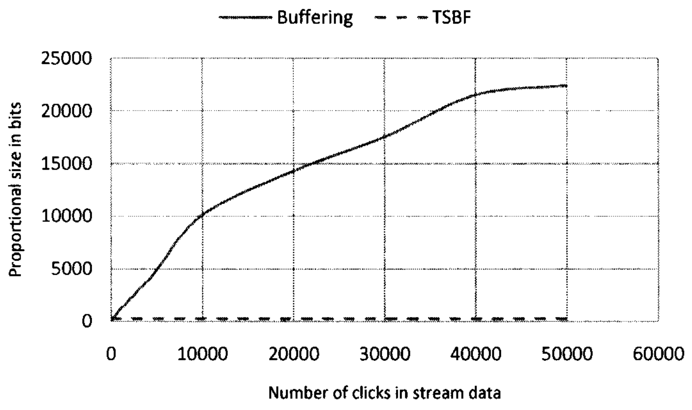


Figure 7.22: Comparison of Memory Usage.

Since we do not lose any information in the buffering mechanism (in the case without replacements) we can safely assume that there are no errors in the duplicate detection. So that we can compare the accuracy and the memory usage of TSBF with the same data set, which had more than 50,000 clicks. For the TSBF we have used $m = 256, k = 4$ with deletion cycle executing every 120 seconds.

As shown in Table 7.12, number of duplicates detected by the both Buffering and TSBF are almost the same even though there are few discrepancies in the later rows of the table. In the rows where there are mismatches, TSBF always reported more duplicate clicks. This is mostly due to the False Positives in our method and buffering do not have

[6]The reason for our hypothesis is that we had several software filters that filters obvious invalid requests. For example since the advertisements in these websites are shown only to the US customers these filters do not allow requests from other IPs except those inside the US.

Table 7.12: Comparison of different methodologies for duplicate detection in a click stream

Clicks Processed	Duplicates Detected		$\propto$ Memory Used		TSBF Accuracy (%)
	Buffering	**TSBF**	Buffering	**TSBF**	
10	5	**5**	20	**256**	100
50	27	**28**	68	**256**	96.3
75	40	**41**	108	**256**	97.5
100	53	**54**	144	**256**	98.1
200	105	**107**	292	**256**	98.1
300	168	**170**	408	**256**	98.8
400	247	**249**	464	**256**	99.2
500	303	**305**	604	**256**	99.3
750	444	**448**	904	**256**	99.1
1000	590	**594**	1164	**256**	99.3
2000	1251	**1261**	2020	**256**	99.2
3000	1881	**1890**	2996	**256**	99.5
4000	2569	**2584**	3828	**256**	99.4
5000	3115	**3134**	4872	**256**	99.4
10000	5939	**5976**	10124	**256**	99.4
20000	11168	**11222**	14320	**256**	99.5
30000	16385	**16445**	17540	**256**	99.6
40000	21449	**21523**	21524	**256**	99.6
50000	22332	**22428**	28500	**256**	99.6

false positives. False positives occur when an distinct element is wrongly reported as a duplicate. In the Bloom Filter approach this error is inherited and we can minimize it with utilizing a bigger size Bloom Filter. But, since the error rate is under 1% on average, we allow that to occur to enjoy the other benefits from using a BF.

With the use of the status vector in the TSBF approach it is possible to minimized the false negative errors. False negatives occur when duplicate element are wrongly identified as distinct. In order to identify a duplicate as a distinct, if an ordinary bloom filter is used, at least one of the mapped bit positions of the vector must have a zero. Those bit positions may have been changed by an operation associated with the another deletion, because same bit position can be shared by few other elements. In contrast, if the TSBF is used, the deletion in the shared bit position will be changed from "recently seen" to "partially seen" to "unseen". Therefore, when the new element arrives, instead of a 0, it will have "partially seen" status, which will help to correctly identify the new element as a duplicate. The ability of TSBF to keep this history information will eliminate the possibility for a false negative. If more values for the status vector are used it is possible to totally eliminate the false negatives. Therefore, in this experiment we assume that there are no false negatives present in the TSBF. The Buffering without replacement does not have either false negatives or false positives. So, the difference in the duplicates detected are solely due to the false positives in the TSBF. The average false positive rate from the experiment is less than 1% but we consider 1% false positive rate for TSBF that will be used in the experiments to follow.

After this results we have reached to two conclusions. First, for our application if one wishes to use a buffering technique then he/she must utilize a replacement method. Second, we can use a TSBF for duplicate detection in click stream, if we allow a very little error rate to occur.

Based on the first conclusion we have modified the buffering mechanism to utilize the LRU replacement technique, which now replaces the Least Recently Used item with the new item, when the buffer is full. But, with this modification, other than costly linked lists deletions we will now have False Negatives(FN) in the buffering. FN is the occasion where a duplicate element is wrongly interpreted as a distinct. FN occurs due to a gap in the data stream, which is the number of elements between a duplicate and its nearest predecessor. Since LRU get rid of least recently used item, it may be the item that is newly arrived. But since its predecessor is already deleted, buffer will treat it as a distinct element. Table 7.13 shows the false negatives which we have seen in the Buffering with LRU replacement. Compare to TSBF the error rate high. Therefore, even with the replacement mechanism Buffering mechanism is not suitable for detecting duplicates in the pay-per-click streaming data.

FN are the main reason for us to develop TSBF, comparing to its predecessor Counting Bloom Filter (CBF). In the next section we discuss effect of FN in the CBF in duplicate detection.

7.2.8 False Negatives in CBF

A false negative(FN) is an error when duplicate element is wrongly reported as distinct. It is generated only by duplicate elements, and is related to the input data distribution, especially the distribution of gaps. A gap is the number of elements between a duplicate and its nearest predecessor. Suppose a duplicate element x_i whose nearest predecessor is $x_{i-\delta_i}(x_i = x_i - \delta_i)$ is hashed into k cells is decremented to 0 within the δ_i iterations when x_i arrives, then there will be FN. If there are no duplicates in the data stream, there are no predecessors and therefore FN will be zero.

How FN are possible with CBF? In the CBF when its deletion cycle is executed the counters are decremented by 1. For example, let us assume we are using a CBF with $k = 4$ and that there are two clicks(click i, and click j) from the same IP, which are close enough to consider as duplicates. After click i is inserted the corresponding counters are $(2, 3, 2,$ and $1)$. By this time if the deletion is executed the counters become $(1, 2, 1,$ and $0)$. And now when click j comes, since one of the counters is zero, it will not be detected as a duplicate.

Table 7.13: False Negatives in Buffering with LRU Replacement

Clicks Processed	Buffering without Replacement	Buffering with LRU Replacement	Error
10	5	5	0
50	28	27	1
75	41	40	1
100	54	51	3
200	107	100	7
300	170	158	12
400	249	234	15
500	305	286	19
750	448	411	37
1000	594	547	47
2000	1261	1162	99
3000	1893	1759	134
4000	2586	2409	177
5000	3136	2924	212
10000	5980	5601	379
20000	11231	10531	700
30000	16456	15471	985
40000	21535	20222	1313
50000	22442	20998	1444

Counting Bloom Filters have both false positives and false negatives. We can conduct experiments to find out the total error, but it is difficult to separate them as false positives and false negatives experimentally. In one of the previous experiments we have shown that on average TSBF have 1% false positive error. The architecture of the TSBF and CBF are almost the same except that the TSBF uses 2 bits to represent the status and CBF uses 4 bits to represent the count. Therefore, we assume that CBF also has a 1% false positive rate, and whatever remains from the total error will be false negatives.

We have conducted several rounds of experiments to see the effect of the false positive and negative errors of the CBF. Table 7.14 shows the results. In Figure 7.23 we have plot the error with the stream size and the variation is linear. In the CBF compare to false positive rate, false negative rate is high. What does this mean in the pay-per-click model? If false negatives are high, that means duplicate clicks are wrongly detected as distinct. In this case, advertisers will be charged for the click. On the other hand, if false positives are high, distinct clicks are wrongly identified as duplicate and publishers will not get paid for unique clicks.

To reduce the FN either we have to reduce the frequency of the deletion cycle or completely change the structure of the CBF. The deletion cycle depends on the application and, in a system like pay-per-click, it is rather impossible to adjust the deletion cycle because the time period where two clicks are considered duplicate is almost fixed. Therefore the approach is to change the structure of the CBF to leave some residue when counters are decremented. For example a counter does not turn on and turn off immediately, during an insertion or deletion(i.e. to switch from 1 to 0), but it goes through several states before turns to 0, For example state 1: recently seen, 2: partially deleted, and 3: never seen. Therefore in the previous example when the first click arrives counter will be (2, 2, 2, and 2), and after the deletion cycle, they will be (1, 1, 1, and 1). When the second click arrives counters will be at state 1, which will now correct the error and identify the duplicate.

This modification in TSBF eliminates the chances for FNs, but there is a little

Table 7.14: False Negatives

Clicks Processed	Duplicates detected		Total Error	False Negative rate(%)
	HT	CBF		
10	5	5	0	0.00
50	28	27	1	2.57
75	41	40	1	1.43
100	54	51	3	4.55
200	107	100	7	5.54
300	170	158	12	6.05
400	249	234	15	5.02
500	305	286	19	5.23
750	448	411	37	7.26
1000	594	547	47	6.91
2000	1261	1162	99	6.85
3000	1893	1759	134	6.08
4000	2586	2409	177	5.84
5000	3136	2924	212	5.76
10000	5980	5601	379	5.34
20000	11231	10531	700	5.23
30000	16456	15471	985	4.98
40000	21535	20222	1313	5.09
50000	22442	20998	1444	5.43

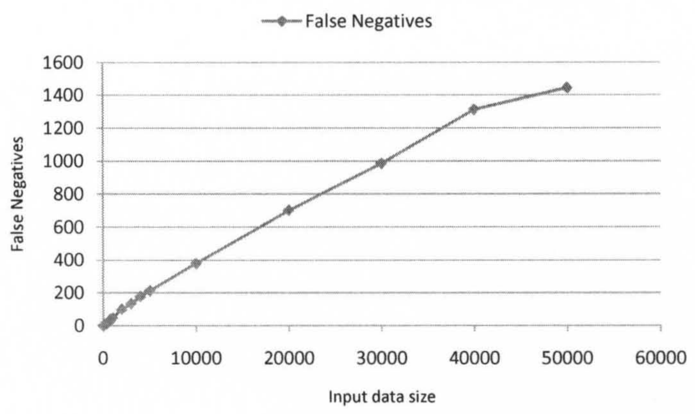


Figure 7.23: False Negatives.

chance (about 1%) for FP with TSBF. Since buffering does not have FP, to compare our results fairly and effectively, we used a variation of buffering called *FPbuffering* [Deng and Rafiei, 2006] introduced by Deng et al.

In FPbuffering, the buffer is searched when a new element arrives. If found, it is reported as a duplicate. Otherwise report it as a duplicate with probability q and as a distinct with probability $(1 - q)$. In the original buffering, if an element is not found in the buffer, it is always reported as a distinct. This variation can increase the overall error rates of buffering when there are more distincts in the stream, but can decrease the error rates when there are more duplicates in the stream. Clearly, FPbuffering has both FPs and FNs. In fact, as Deng et al. suggested q is the FP rate since a distinct element will be reported as duplicate with a probability q.

In the following experiment we compared the error rates between TSBF ($m = 256$,$k = 4$), FPbuffering with LRU replacement on the real data by varying the allowable FP rate (i.e. q) and stream sizes. For the results in Table 7.15, we calculated the Error rate as (number of FPs/number of distincts) with the assumption that with Buffering without replacement gives correct number of duplicates. As the result shows TSBF error rate is almost always less than 1% except for the first five data sets. Even though the error rate is high, the actual difference is maximum of 2 clicks. For the FPbuffering we have changed the FP rate, which is the variable q, between $0.01, 0.1, 0.2$, and 0.5. FPbuffering gives its best performance when the allowable FP rate is 1%. The results are almost similar to TSBF. When the q value is 0.1,(or allowable FP rate is 10%) the maximum error rate is slightly less than 10%, which may be acceptable in some applications but not in ours. But when the allowable FP rate higher than 10% FPbuffering has its worst performance.

Clicks Processed	Total duplicates with Buffering	Total duplicates with TSBF	Error rate	Duplicates detected							
				$q = 0.01$	Error rate(%)	$q = 0.1$	Error rate(%)	$q = 0.2$	Error rate(%)	$q = 0.5$	Error rate(%)
10	5	5	0	5	0	5	0	5	0	5	0
50	27	28	3.7	28	3.7	29	7.4	30	11.1	37	37.0
75	40	41	2.5	41	2.5	43	2.5	45	12.5	47	17.5
100	53	54	1.8	54	1.9	57	7.5	61	15.1	69	30.2
200	105	107	1.9	107	1.9	112	6.7	120	14.3	141	34.2
300	168	170	1.2	169	0.6	184	9.5	192	14.3	210	25.0
400	247	249	0.8	249	0.8	254	2.8	270	9.3	299	21.1
500	303	305	0.6	305	0.6	314	3.6	330	8.9	378	24.7
750	444	448	0.9	445	0.2	469	5.6	475	6.9	543	22.3
1000	590	594	0.7	597	1.1	620	5.1	643	8.9	744	26.1
2000	1251	1261	0.8	1255	0.3	1314	5.0	1343	7.4	1491	19.2
3000	1881	1890	0.4	1895	0.7	1948	3.6	2042	8.6	2252	19.7
4000	2569	2584	0.5	2580	0.4	2660	3.5	2745	6.9	3048	18.5

5000	3115	3134	0.6	3125	0.3	3238	3.9	3372	8.3	3733	19.8
10000	5939	5976	0.6	5968	0.5	6188	4.2	6439	8.4	7197	21.2
20000	11168	11222	0.5	11198	0.3	11511	3.1	11858	6.2	12919	15.6
30000	16385	16445	0.4	16431	0.3	16816	2.6	17216	5.1	18574	13.3
40000	21449	21523	0.3	21509	0.3	21988	2.5	22531	5.0	24172	12.6
50000	22332	22428	0.4	22394	0.3	23078	3.3	23763	6.4	25934	16.1

Table 7.15: Comparison of error rates between TSBF,FPbuffering, and Buffering

7.2.9 Double clicks Vs. Multiple clicks

Are multiple clicks on the same ad (usually known as impressions), always considered duplicates? For example, consider the case of a double click, i.e., two clicks occur on the same ad, where the second click follows the first one within a time period p. Is the second click a double click always? Unfortunately, in most of these cases it is hard or even impossible to determine the true intent of a click. Therefore in such cases we adapt a simple measure, which is the time difference between the two clicks. If p is really small, e.g. "fraction of a second", this click is considered invalid but not fraudulent because the person who made the click may be accustomed to Windows^{TM} default method of invoking options in an application. Therefore, we have to disregard such clicks when duplicate clicks are detected. It can be easily done with Buffering approach, since it is merely a comparison of two time stamps. In order to distinguish these double clicks with TSBF, we have to increase the resolution of the status vector S. This can be done by increasing the maximum number of states in V. Figure 7.24 depicts this extended status vector with $V = 5$ along with the original TSBF. With the new configuration, states in S change frequently than that of with $V = 3$. We have performed experiments to detect such double clicks with the new status vector and results are shown in Table 7.16 and Figure 7.25 show the break down of double clicks in one minute time intervals. For example, in the dataset with 20,000 records 95.6% of the duplicates are recorded within the first minute and 3.3%,0.7%, and 0.4% duplicates are recorded in the second, third, and fourth minute respectively.

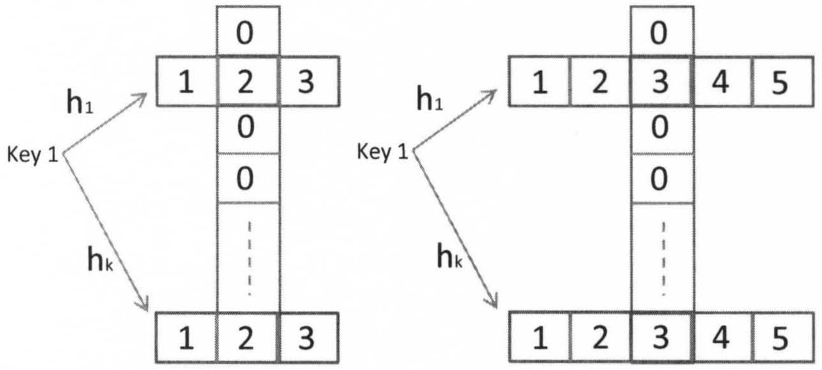


Figure 7.24: Extended status vector for duplicate detection.

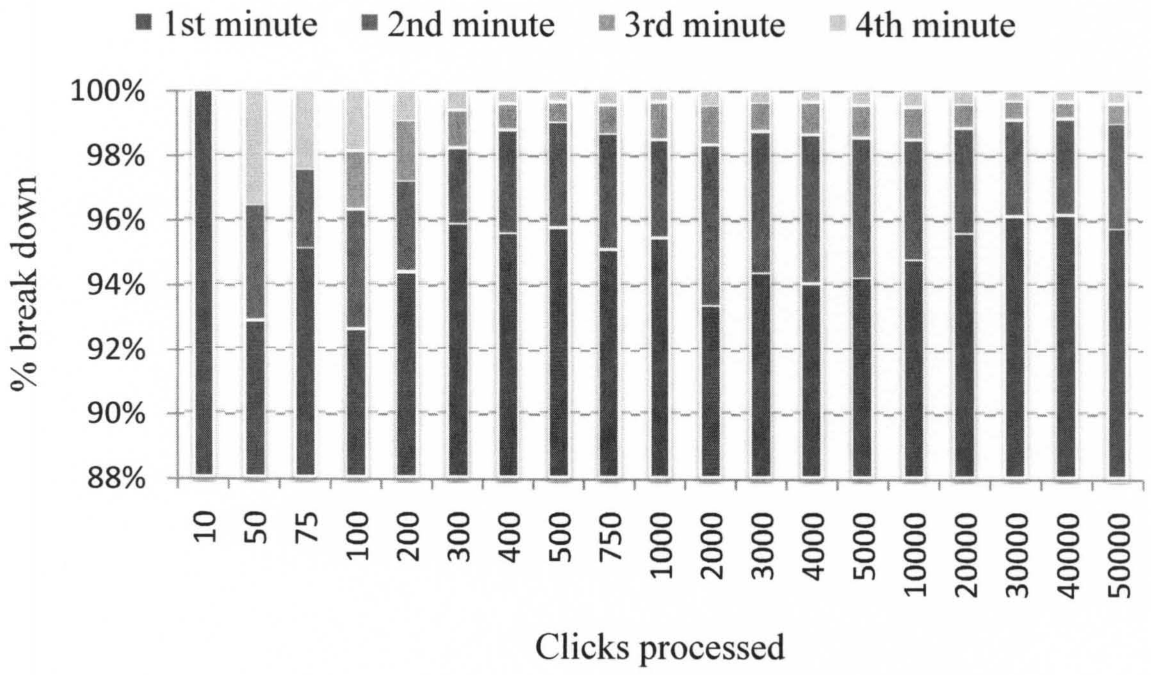


Figure 7.25: Break down of duplicate clicks.

Table 7.16: False Negatives

Clicks Processed	Total duplicates	Second click detected within			
		1st	2nd	3rd	$\geq 4^{th} minute$
10	5	5	0	0	0
50	28	26	1	0	1
75	41	39	1	0	1
100	54	50	2	1	1
200	107	101	3	2	1
300	170	163	4	2	1
400	249	238	8	2	1
500	305	292	10	2	1
750	448	426	16	4	2
1000	594	567	18	7	2
2000	1262	1178	63	15	6
3000	1894	1787	83	17	7
4000	2600	2445	120	26	9
5000	3154	2971	137	32	14
10000	6006	5692	223	62	29
20000	11266	10767	370	82	47
30000	16496	15854	494	95	53
40000	21586	20757	644	112	73
50000	22479	21518	730	143	88

7.2.10 Time Complexity Comparison for Duplicate Detection

Since our goal is to minimize the error rates given a fixed amount of space and acceptable FP rate, we focus only on time complexity. There are several parameters to be set in our method. For each newly arrived element (click) we probe m cells to detect duplicates. After that we pick k cells and update the states of them. Last we update all non-zero cells ($< m$) when the decaying function is invoked. Therefore time required for TSBF, CBF, and BF to process one element is $O(1)$, however, the exact time depends on the parameter settings.

For buffering and FPbuffering processing time is depends on two processes: element searching and element evicting. Searching can be quite expensive without an index structure. We used a hash table to accelerate the search process. The extra space that is needed for a hash table to keep the search time constant is linear in the number of elements stored. The process of maintaining the LRU replacement policy (finding the least recently used element) is also costly, and extra space is needed to make it faster. This extra space can be quite large for LRU.

7.2.11 Comparison with similar work for duplicate detection in Streaming data

In [Metwally et al., 2005b], the authors proposed to maintain a CBF for each sub-window, and a main BF which is a combination of all CBF and represents the entire jumping window. When a new sub-window is generated, the eldest window is expired and subtracted from the main Bloom filter. Combining two CBFs is performed by adding the corresponding counters, deleting an old CBF is performed by subtracting its counters from the main Bloom filter.

However, this scheme has two potential drawbacks. One is that subtracting an expired Bloom filter from the main Bloom filter needs $O(m)$ operations, and false positives increase if new elements are inserted into the main Bloom filter before subtracting operation completes. The other drawback is that this scheme may have high false positive rate, especially when the number of sub-windows is large. There are two reasons for this drawback. First, with the same limited available memory space, expanding bits in Bloom filters to counters make the size of Bloom filter smaller. In worst case, the maximum value in the counters of counting Bloom filters is N/Q, and the maximum value in the counters of the main Bloom filter is N. Therefore, each counter must ave enough bits to avoid saturation, which will generate both false negatives and false positives. Consequently, the size of the Bloom filters in their algorithm is much smaller than the size of Bloom filters in our TSBF algorithm, and the false positive rate will be much higher than that of TSBF algorithm. Second, checking the presence of an element in the main Bloom filters which is the result of combination of all CBFs will generate very high false positive rate, since it is as if all N elements are inserted into the single main Bloom filter(any entry with non-zero value in any CBF will set the corresponding entry in the main Bloom filter to non-zero).

7.3 Modeling of Knowledge and Validation

Most service providers currently approach the problem of click fraud by attempting to automatically recognize fraudulent clicks and discount them. Fraudulent clicks are recognized by machine learning algorithms, which use information regarding the navigational behavior of users to try and distinguish between human and robot generated clicks [Immorlica et al., 2005]. Such algorithms are mainly built using rule based techniques and most of them are classification systems [Metwally et al., 2005a], even though a few score based systems are also reported [Ge and Kantardzic, 2006, Kantardzic et al., 2008, Kantardzic et al., 2009]. Most of the click fraud solution providers, including search engines and third party solution providers, claim their rule-based expert system is the best among the others taking the advantage of keeping "rules" as a secret weapon. They do not disclose information about the set of rules due to fear of competition. This situation even led to multi-million dollar settlements [Hadjinian et al., 2006, Tuzhilin, 2006] in the recent years.

Due to the lack of verifiability of click fraud solutions, it is inevitable that the trust between service providers and advertisers is degraded. Since real-world click fraud solutions are usually kept secret for fear of competition, it is practically impossible to study many of them in a single context. In this research we discuss the modeling of knowledge and validation (KV) model for rule based expert systems used in click fraud detection. We have considered the case studies of one real-world click fraud detection rule-based expert system [Kantardzic et al., 2009] for validation. In future, solution providers will be able to use our tool to test their systems without revealing the implementation details of the rule base.

7.3.1 Rules in Knowledge-based Systems

Knowledge-based systems have received great attention and have become an essential tool in business, science, engineering, manufacturing, and many other fields. Because of the natural representation and the powerful inference ability of production rules, knowledge-based systems are usually implemented in the form of rule-based system (RBS). Usually a rule in an RBS describes an IF-THEN relationship of the form: "LHS $\Rightarrow$ RHS," where LHS is a collection of conditions, and RHS is a collection of actions or conclusions [Wu and Lee, 2002]. If the conditions are met a certain value is assigned and if not another threshold value is assigned. Rule-based expert systems have been in use for a couple of decades, and their usefulness has been demonstrated in many domains. However, they also draw a lot of criticism [Ben-David, 2008]. In particular, deriving the "right" rules proved a very difficult task. Another issue has been the need to adjust rule-based systems to cope with changing conditions. Although rule-based systems provide the major advantage of being interpretable, as rule-based systems grow larger, it becomes increasingly impractical to manually check each rule for consistency, redundancy etc. This topic, thus, drew the attention of the research community, and algorithmic tools began to emerge [Ayel and Laurent, 1991, Ben-David, 2008, Gupta et al., 1991].

7.3.2 Proposed Validation Framework

Despite the enormous capabilities of modern machine learning and data mining techniques in modeling complicated problems, most of the available click fraud detection systems are rule-based. This is mainly due to the fact that advertisement click streams are very hard to be labeled as fraudulent because of the infinite situations in which a person could click on an advertisement without getting through with the purchase. However, scenarios where an advertisement is clicked 100 times in 1 second is definitely fraudulent. Therefore, rules invented by experts to detect such fraudulent clicks are manageable and their output is clear. As the characteristics of a fraudulent click become more subtle, the output score of a rule-based system becomes harder to interpret. This raises two questions a) are the derived rules enough to model the fraudulent behavior? and b) what is the value of a score threshold to announce a click fraudulent. Question (a) is about the validation of the rule-based scores. In other words are we using the right rules/model.

The idea behind the proposed validation framework is basically to acquire another model of the clicks data that is not rule dependent, a model that learns the inherent statistical regularities of the data. Then the output of both models is compared. If both models are consistent, then the rules are reflective of the actual inherent knowledge in the data and therefore they provide a good interpretation of the structure of the problem in hand. On the other hand, a discrepancy between the models indicates a lack of knowledge about the system and thus it might be helpful to extract extra rules from the machine learning model so as to be added to the rule-based system. Needless to say that the choice of a proper machine learning technique is of utmost importance.

Rule-based click fraud detection systems are very reliable in extreme cases where a click is legitimate if a purchase is made while a click is fraudulent if, for example, there was no window opened to make the click (i.e. click bots). In the proposed validation framework we train a classification algorithm to classify clicks as legitimate or fraudulent. Using semi-supervised learning is natural as few samples (i.e. extreme cases clicks) have verified labels while the scores for the rest of the clicks are not definitive and again the threshold to classify them is not known. Due to the good generalization performance of Support Vector Machines (SVM), we use Semi-supervised SVM (S^3VM) [Joachims, 1999] to model the inherent regularities of the click fraud data.

The validation process consists of following related steps.

(a) Select a subset of clicks with confident labels (legitimate/fraudulent) as described earlier. The rest of the clicks are considered unlabeled samples and they are used for their internal structure.

(b) Train S^3VM on both labeled and unlabeled clicks. After training, each unlabeled click will be assigned a label (legitimate/fraudulent).

(c) As click fraud systems produce scores rather than classifications, we transform the S^3VM classifications of unlabeled clicks into posterior probabilities using [Platt, 1999].

(d) Both scoring techniques are compared.

7.3.3 Results of validation model

We used a data set of 17,558 clicks produced using our CCFDP system (`www.netmosaics.com`) [Kantardzic et al., 2008, Kantardzic et al., 2009]. These data are from an actual ad campaign conducted in 2008. The CCFDP system produced a score between 0 and 1 for each click in the data set, where 0 is legitimate and 1 is fraudulent. To generate training data set for the S^3VM some of these clicks should be relabelled for the classifier. We selected clicks with the scores in the range 0 and 0.1 as legitimate clicks. These clicks are labelled as 0 for the S^3VM data set. Similarly, we selected clicks with the scores in the range 0.9 and 1.0 as fraudulent clicks. These clicks are labelled as class 1 for the S^3VM data set. The remaining data, with the scores between 0.1 and 0.9, are assigned to be unlabeled.

Labelled data set is used to train the S^3VM, and Figure 7.26 shows the scores obtained from S^3VM modeling compared with the original scores from the CCFDP system. It is clear that the two sets of scores highly match. The Pearson correlation coefficient between two series is 0.973. This indicates that the rules used in the CCFDP system are greatly consistent with the structure of the click data. Does this give us the chance to get rid of

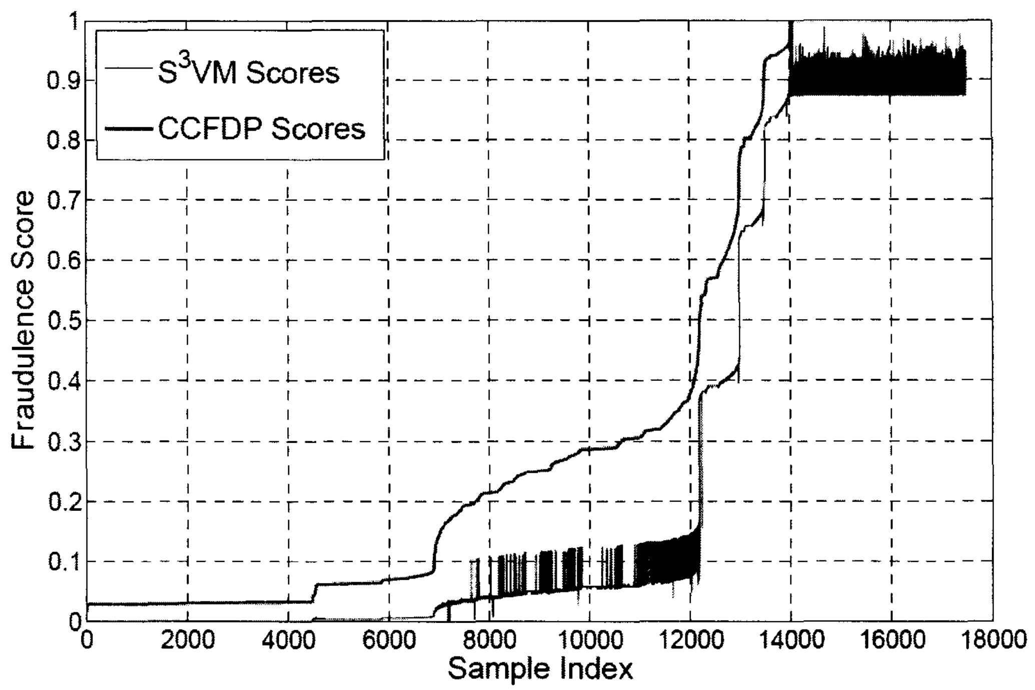


Figure 7.26: Comparison of scores produced using the CCFDP system and S^{3}VM algorithm.

the rules in the CCFDP? Yes, not only can it replace the rules but it can also eliminate the controversy due to many rule based solutions offered in the market.

We have seen that due to the lack of verifiability of click fraud solutions, it is inevitable that the trust between service providers and advertisers is degraded. Until today and many more years ahead, real-world click fraud solutions will usually kept secret for fear of competition. But, with the introduction of this validation model, click fraud solution providers can validate their rule-based solutions without revealing the actual contents.

Currently we are applying the framework on several data sets. Also we are investigating the issue of using the output of S3VM to estimate a proper threshold on the scores produced by the CCFDP system. In the near future, besides working on extracting rules to model the differences between scoring methods, we will investigate semi-supervised regression methods which might be more appropriate for continuous scoring purposes.

In the future work we will examine methods to model the differences between these scoring methods and how to extract rules that will add to our understanding of the click fraud problem in the form of new click fraud patterns and schemes.

CHAPTER 8

CONCLUSIONS

This dissertation presents new methodologies for click fraud detection and prevention in real time. The proposed solution analyzes the detailed user activities on both, the server side and client side collaboratively to better describe the intention of the click. Data fusion techniques are developed to combine evidences from several data mining models and to obtain a better estimation of the quality of the click traffic. Our ideas are experimented through the development of the Collaborative Click Fraud Detection and Prevention (CCFDP) system. Experimental results show that the CCFDP system is better than the existing commercial click fraud solution in three major aspects: 1) detecting more click fraud especially clicks generated by software; 2) providing prevention ability; 3) proposing the concept of click quality score for click quality estimation.

The existing commercial click fraud solutions cannot detect software clicks, which is one of the major forms of click fraud. Also those solutions cannot prevent click fraud beforehand. Our solution identifies click fraud by using both server side and client side data. The server side differentiates our approach from existing commercial solutions, and it allows detection of very frequent software click fraud. To improve the detection process, we added extended parameters such as mouse movement, mouse click, key stroke etc. to the client side data. Our analysis is extended with these parameters, and results show that it improves the detection capabilities.

Due to the nature of inherent weakness of Googles solution (or any other search engine, ISP based solution), which does not have enough data on post-click user activities, it is hard or even impossible to determine the true intent of a click. There are even more weaknesses in the Goolges online procedure: lack of deployment of data mining methods;

lack of use of conversion data, and lack of more advanced types of filters. Our approach collects both the server side click and post-click user activities making better infrastructure than the server side only solutions.

In the CCFDP initial version, we analyzed the performances of the click fraud detection and prediction model by using a rule base algorithm, which is similar to most of the existing systems. We have assigned a quality score for each click instead of classifying the click as fraud or genuine, because it is hard to get solid evidence of click fraud just based on the data collected, and it is difficult to determine the real intention of users who make the clicks.

The diversity of CF attack types makes it hard for a single counter measure to prevent click fraud. Therefore, it is important to be able to combine multiple measures capable of effective protection from click fraud. Therefore, in the CCFDP improved version, we provide the traffic quality score as a combination of evidence from several data mining algorithms.

We have tested the system with a data from an actual ad campaign in 2007 and 2008. We have compared the results with Google Adwords reports for the same campaign. Results show that a higher percentage of click fraud present even with the most popular search engine. The multiple model based CCFDP always estimated less valid traffic compare to Google. Sometimes the difference is as high as 53%.

Detection of duplicates, fast and efficient, is one of the most important requirement in any click fraud solution. Usually duplicate detection algorithms run in real time. In order to provide real time results, solution providers should utilize data structures that can be updated in real time. In addition, space requirement to hold data should be minimum.

In this dissertation, we also address the problem of detecting duplicate clicks in pay-per-click streams. We proposed a simple data structure, Temporal Stateful Bloom Filter (TSBF), an extension to the regular Bloom Filter and Counting Bloom Filter. The bit vector in the Bloom Filter was replaced with a status vector. Depending on the insertions and deletions these states take different values. By introducing the status vector we have eliminated the false negatives occur in the Counting Bloom Filter. Therefore, our experiments were focused on minimizing false positives that occur in TSBF. We have carried out experiments both with synthetic and real world data sets. Results of TSBF method is compared with Buffering, FPBuffering, and CBF methods. False positive rate of TSBF is less than 1% and it does not have false negatives. Space requirement of TSBF is minimal among other solutions. Even though Buffering does not have either false positives or false negatives its space requirement increases exponentially with the size of the stream data size. When the false positive rate of the FPBuffering is set to 1% its false negative rate jumps to around 5%, which will not be tolerated by most of the streaming data applications. We also compared the TSBF results with CBF. TSBF uses only half the space or less than standard CBF with the same false positive probability. CBF also suffer from false negatives, while TSBF does not have false negatives. Due to these advantages TSBF technique has replaced the Buffering based duplicate detection mechanism used in the NetMosaics click fraud detection system.

One of the biggest success with CCFDP is the discovery of new mercantile click bot, the Smart ClickBot. We presented a Bayesian approach for detecting the Smart ClickBot type clicks. The system combines evidence extracted from web server sessions to determine the final class of each click. Some of this evidence can be used alone, while some can be used in combination with other features for the click bot detection. During training and testing we also addressed the class imbalance problem. Our best classifier shows recall of 94%, and precision of 89%, with F_1 measure calculated as 92%. The high accuracy of our system proves the effectiveness of the proposed methodology. Since the Smart ClickBot is a sophisticated click bot that manipulate every possible parameters to go undetected, the techniques that we discussed here can lead to detection of other types of software bots too.

Despite the enormous capabilities of modern machine learning and data mining techniques in modeling complicated problems, most of the available click fraud detection systems are rule-based. This is mainly due to the fact that advertisement click streams are very hard to be labeled as fraudulent because of the infinite situations in which a person could click on an advertisement without getting through with the purchase. This raises two questions a) are the derived rules enough to model the fraudulent behavior? and b) what is the value of a score threshold to announce a click fraudulent. Question (a) is about the validation of the rule-based scores. In other words are we using the right rules/model.

We proposed validation framework to acquire another model of the clicks data that is not rule dependent, a model that learns the inherent statistical regularities of the data. Then the output of both models is compared. If both models are consistent, then the rules are reflective of the actual inherent knowledge in the data and therefore they provide a good interpretation of the structure of the problem in hand. On the other hand, a discrepancy between the models indicates a lack of knowledge about the system and thus it might be helpful to extract extra rules from the machine learning model so as to be added to the rule-based system. Needless to say that the choice of a proper machine learning technique is of utmost importance.

Due to the uniqueness of the CCFDP system architecture, it shows better click fraud detection than current commercial solution and search engine/ISP solution. The system will protect Pay-Per-Click advertisers from click fraud and improve their Return on Investment (ROI). The system can also provide an arbitration system for advertiser and PPC publisher whenever the click fraud argument arises. Advertisers can gain their confidence on PPC advertisement by having a channel to argue the traffic quality with big search engine publishers. General consumer will gain their confidence on internet business model by reducing fraudulent activities which are numerous in current virtual internet world.

We have been carrying out click fraud related research studies since 2004, for the benefit of the Internet community. It is worth mentioning that, we have to accept the fact that well-undercover bots will not be spotted and we have to reach a sustainable tradeoff for the detection model. Any attempt to identify deep undercover bots is likely to produce high false positive rates. A high false positive rate will mean that we are labeling many regular traverses as bot activities. Although we wish to detect as many bots as possible, it is far worse to tag regular users as bots than to skip some detection, because misclassifying regular users as bots may have unpredictable consequences. For instance, blocking a proxy IP by mistake will represent the loss of many users that may no longer visit the site.

CHAPTER 9

FUTURE WORK

The success of the CCFDP opens the door to some new areas of research in the click fraud detection and prevention. We have provided the likely characteristics of future click bots after dissecting an intelligent click bot. Since fraudsters always find ways to go undetected, while committing click fraud, no solution will provide the ultimate protection against click fraud. Therefore, continuous research should be a prime importance to detect future click fraud attacks. In this chapter we provide more ideas that can be used towards strengthening click fraud solutions.

Most of the solutions including our CCFDP depends on how well a user is identified and tracked during each of his /her visits to a website. For example, CCFDP system requires a user to allow third party cookies and run javaScript in his browser. In some situations a user may disable cookies intentionally or may not allow javaScript to run on his browser. In such cases it is difficult for CCFDP to track a user successfully. Some of the ideas explained here are proposed by Shat et.[Shah, 2005] for different context. In the following section we list few ideas that can be used towards user verification.

9.1 Browser Verification

User agent is an unique identifier of a browser. Click bots try to mimic the user agents of known browsers to go undetected. They are programmed to generate "fake" clicks that mimics actual clicks. Most of the time it is an HTTP request that is artificially generated. Therefore it is important to verify whether the click is originated from an authentic browser such as Internet Explorer, Mozilla, etc.

Two experiments are proposed to identify browser requests and they are explain below.

(a) Using extended context of the UserAgent Header

(b) Using recently visited social network websites

9.1.1 Using extended context of the UserAgent Header

9.1.1.1 UserAgent Header

When a browser requests a page from a Web server, the browser sends information about itself along with the request. These value strings are called headers. Typically, this information includes the browser type (Internet Explorer, Opera, Mozilla, etc.), the browser version, and the underlying platform (Windows XP, Linux, Mac OS X, etc.). The server then uses this information to select an appropriate page format for the browser, since different browsers (and even different versions of the same browser) have varying incompatibilities in their support for HTML and JavaScript.

For example if Internet Explorer is to fetch the URL http://www.amazon.com/index.html, this is what the browser might send to Amazon's server:

```
GET /index.html HTTP/1.1
Host: www.amazon.com
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
```

Figure 9.1: The User-Agent Header

The last four lines of this request are headers. Each header consists of a name and a value, separated by a colon. All possible headers are defined by the HTTP protocol specification. A browser identifies itself using the User-Agent header as shown in Figure 9.2. (User agent is a generic term for an application, like a browser, that is acting as an agent for the user). The header value consists of a series of product identifiers and/or comments. A product identifier is a string like "Mozilla/4.0" or "Opera/7.02" that identifies the product by name and (optionally) by version. Additional attributes about the product, referred to as comments, are enclosed in parentheses, such as "(compatible; MSIE 6.0; Windows NT 5.0)". The product identifiers and comments can come in any order, but in general the most significant values are listed first.

The Figure 9.2 shows a sample user-agent string reported by Internet Explorer that highlights its tokens.

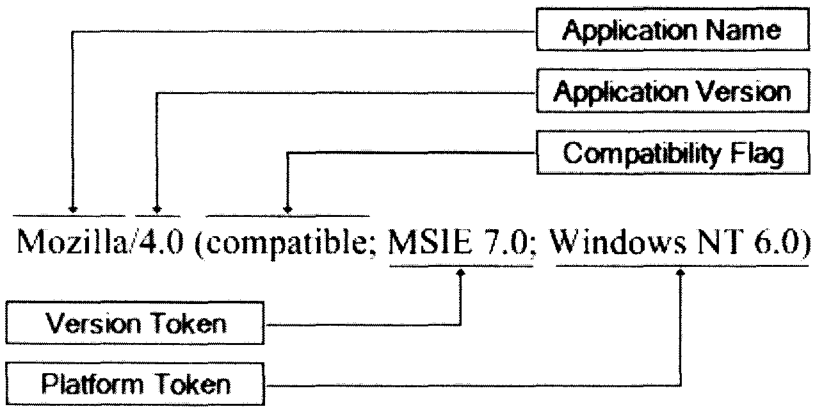


Figure 9.2: User-Agent String

For historical reasons, Internet Explorer identifies itself as a Mozilla 4.0 browser. The sample user-agent string contains three tokens.

(a) The Compatibility flag ("compatible") is used by most modern browsers. It indicates that Internet Explorer is compatible with a common set of features.

(b) The Version token identifies the browser and contains the version number. The version token in the example ("MSIE 7.0") identifies Internet Explorer 7.

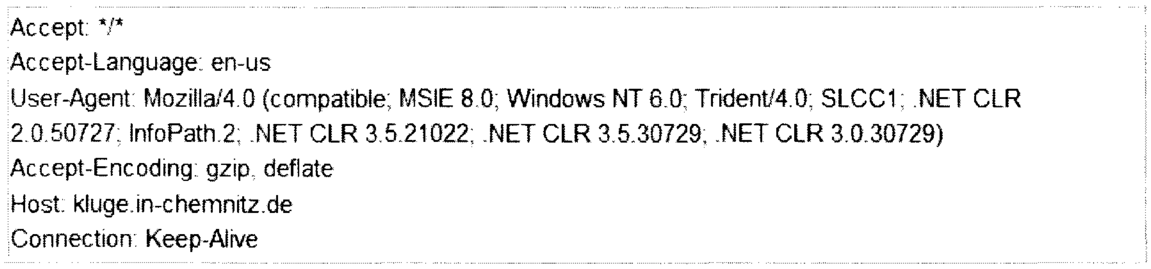
(c) The Platform token identifies your operating system and contains the version number. The platform token in the example ("Windows NT 6.0") indicates Windows Vista.

In the example, Internet Explorer is the user agent. However, other programs also provide user-agent strings when contacting servers over the Internet. For example, the Windows RSS Platform provides the following user-agent header when requesting RSS data [Microsoft, 2010].

Windows-RSS-Platform/1.0 (MSIE 7.0; Windows NT 5.1)

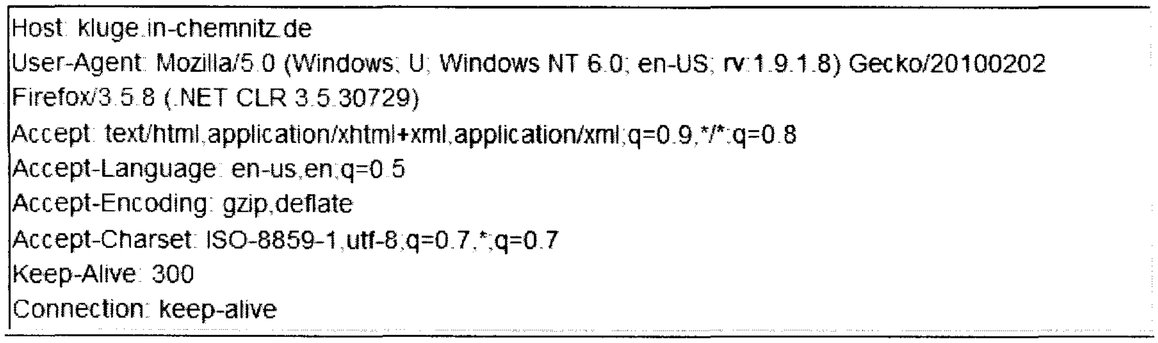
The header shown in Figure 9.1 defines a signature for each known browser. If a click bot has to generate a fake click it should identify itself as one of those legitimate browsers. But experiments have shown that it is difficult to break all signatures of browsers completely. Some of the experiments are listed below but more theoretical understanding and more complex implementations are necessary for successful detection of larger bot population.

Figures 9.3, 9.4, and 9.5 show HTTP GET requests made to a common server from different browsers.



```
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR
2.0.50727; InfoPath.2; .NET CLR 3.5.21022; .NET CLR 3.5.30729; .NET CLR 3.0.30729)
Accept-Encoding: gzip, deflate
Host: kluge.in-chemnitz.de
Connection: Keep-Alive
```

Figure 9.3: HTTP GET request generated by Internet Explorer



```
Host: kluge.in-chemnitz.de
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.1.8) Gecko/20100202
Firefox/3.5.8 (.NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Figure 9.4: HTTP GET request generated by Firefox

9.1.1.2 Observations

- In each request the order of appearance of the HTTP fields is different. For example "Host" is placed as the one before last in Internet Explorer while it appears first in Firefox.

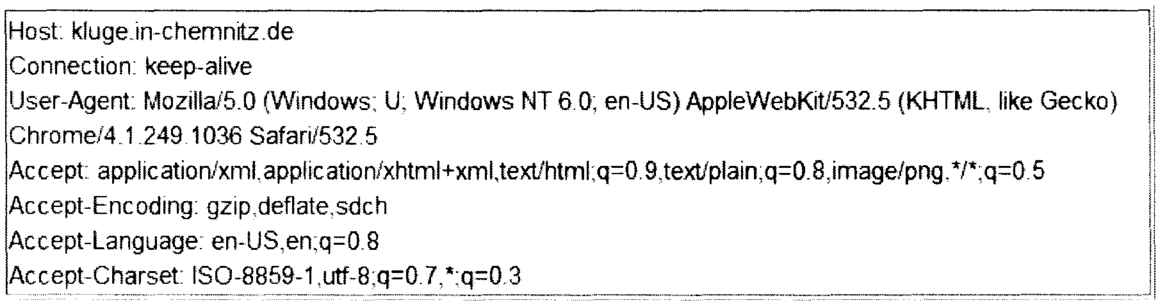

```
Host: kluge.in-chemnitz.de
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.5 (KHTML, like Gecko)
Chrome/4.1.249.1036 Safari/532.5
Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

Figure 9.5: HTTP GET request generated by Chrome

- Values in the "Accept" field are completely different.

Figure 9.7, 9.8, and 9.9 show the changes in headers when cookies are used. We use cookies in CCFDP but never pay attention to how differently they are handled by the browsers. The figures show different browsers respond to two cookies (cookl and cook2). We can identify the browser on the basis of these results.

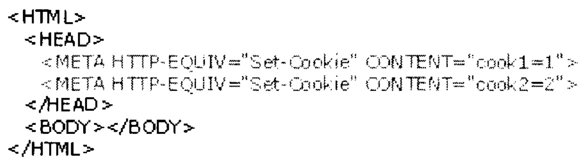

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Set-Cookie" CONTENT="cook1=1">
    <META HTTP-EQUIV="Set-Cookie" CONTENT="cook2=2">
  </HEAD>
  <BODY></BODY>
</HTML>
```

Figure 9.6: HTML code to generate the cookies

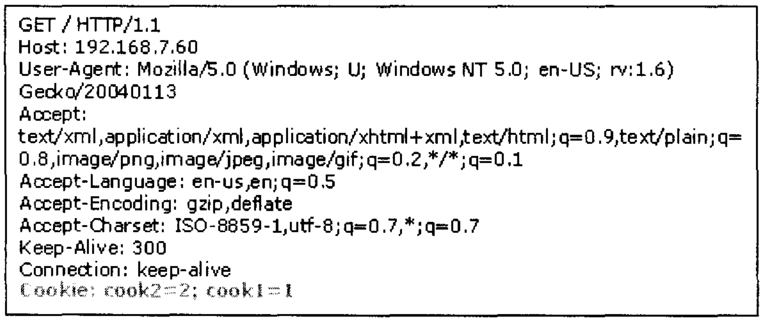

```
GET / HTTP/1.1
Host: 192.168.7.60
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.6)
Gecko/20040113
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=
0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: cook2=2; cook1=1
```

Figure 9.7: HTTP request generated by Firefox

9.1.2 User's frequently visited social websites

Most Internet users at least visits couple of social sites frequently. Examples for social networking sites are Facebook^{TM}, Twitter^{TM}, Youtube^{TM} etc. Globally, social networks

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 192.168.7.60
Connection: Keep-Alive
Cookie: cook1=1; cook2=2
```

Figure 9.8: HTTP request generated by Chrome

```
GET /test HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5; Windows 2000)
Opera 7.0 [en]
Host: 192.168.7.60
Accept: text/html, image/png, image/jpeg, image/gif, image/x-xbitmap,
*/*;q=0.1
Accept-Language: en
Accept-Charset: windows-1252, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Cookie: cook1=1; cook2=2
Cookie2: $Version="1"
Connection: Keep-Alive, TE
TE: deflate, gzip, chunked, identity, trailers
```

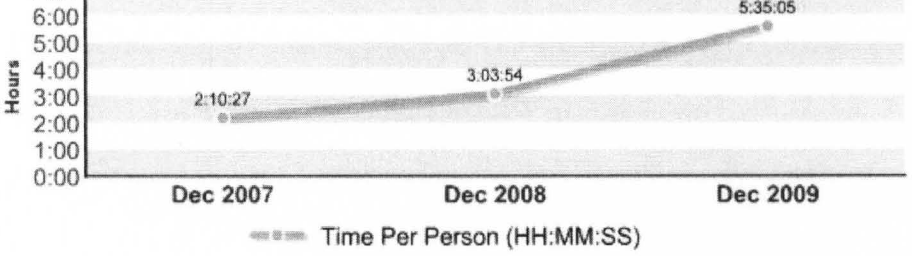
Figure 9.9: HTTP request generated by Chrome

and blogs are the most popular online categories when ranked by average time spent, followed by online games and instant messaging.

According to The Nielsen Company, global consumers spent more than five and half hours on social networking sites like Facebook^{TM} and Twitter^{TM} in the month of December 2009, an 82% increase from the same time last year when users were spending just over three hours on social networking sites (see Figure 9.10). In addition, the overall traffic to social networking sites has grown over the last three years [nielsenwire, 2010]. With 206.9 million unique visitors, Facebook^{TM} was the No. 1 global social networking destination in December 2009 and 67% of global social media users visited the site during the month. Time on site for Facebook^{TM} has also been on the rise, with global users spending nearly six hours per month on the site.

Based on this study we can safely assume that almost all the Internet users today at least visit a couple of those websites in any single day. If we know a user visits these websites, can we believe that it is not a robot in the other end?

The simplest way to access visited websites of a user's machine is to use the informa-



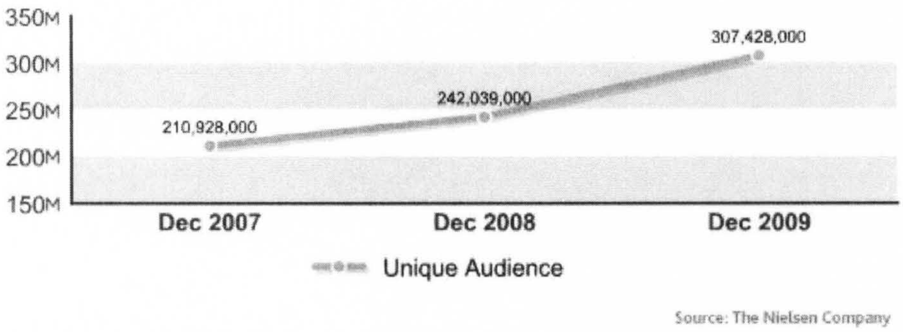


Figure 9.10: Global Web Traffic to Social Networking Sites

tion leak introduced by CSS (Cascading Style Sheets). The browser colors visited links differently than non-visited links. Our javaScript will have URLs of most popular social networking sites in an iframe. Then we can look at which of those links are blue and which are purple with the help of javaScript. We can even extend this idea to track what other pages user frequently visits. Probably to track competitors clicking on their rivals websites.

9.1.2.1 history.js

The complete source code of history.js, can be found in the Appendix B. It was originally published by Aza Raskin at `www.azarask.in`. history.js code enables anyone to detect which social bookmarking sites your visitors use. The following javaScript will be added to the webpage that is being tracked. This will force to the client computer to release websites that the user has recently visited.

```
<script src=''http://65.182.201.98/history.js''></script>
```

```
<script>
  user = history();
  var visitsDigg = user.doesVisit(''Digg'');
  var visitsSlashdot = user.doesVisit(''Slashdot'');
  var listOfVisitedSites = user.visitedSites();
</script>
```

history.js has a list of the most popular social bookmarking sites which it checks against. history.js can also check other sites. For instance, if we want to see if your visitor has visited any of the competitors website:

```
moreSites =
  {
    ''competitor'': [''http://competitor1.com/'', ''http://competitor2.com
        /'']
  };
user = history( moreSites );
alert( user.doesVisit(''competitor'') );
```

CCFDP system installs a javaScript in each web site that it tracks. Therefore the infrastructure required for this experiment is already setup. We can add the additional functionality (i.e. history.js) to the existing javaScript. There will be an additional "attribute" in the table that we install client activities, which we called "social sites visited". We can either implement it as a boolean variable with values (0/1) or as a String array that stores all the websites the user has visited.

When a user clicks on a website that has CCFDP javaSciprt, it will execute on the user's browser and tell the CCFDP if the browser has any visited social networking sites. A bot, which is trying to mimic a browser will not have any value for this variable, because its browsers are implemented to serve only a simple purpose such as generating an HTTP request.

9.2 Improve User Tracking

User tracking is a very important feature for CCFDP because some functionalities of its rule based and the outlier modules are depend on user profiles. When a user visits a client site CCFDP system wants to know how long his visit lasted, which contents he visited, what navigation path was followed and so on. Tracking of users is done with cookies. But what if the user does not allow cookies to stored in its computer?

Two experiments are proposed to track users who do not allow to install cookies in their computers.

(a) Using browser finger printing

(b) Using Super Cookies

9.2.1 Browser finger printing

Until recently, database tables, in which each row of information related to a person, were shared somewhat freely provided none of the columns included explicit identifiers, such as name, address, or Social Security number. This kind of "de-identified" data can often be linked to other tables that do include explicit identifiers ("identified data") to re-identify people by name. Fields appearing in both de-identified and identified tables link the two, thereby relating names to the subjects of the de-identified data. For example, date of birth, gender, ZIP, which commonly appeared in both de-identified and identified data, uniquely identified 87% of the U.S. population [Sweeney, 2002].

The authors are interpreting the information from each individual parameter in (Birthday, sex, and zip) as entropy, $log_2(Prx = X)$, measure in bits. Where $Pr(x = X)$ is simply the probability that the fact would be true for a random person. So, if we assume there are 337 million people in the USA, to identify someone we require around $log_2(1/337million) = 28.9$ bits.

Example

Let us consider a person John living in area 40217.

In 2009, 40217 population is 13600.

If somebody knows John's zipcode, he has $log_2(13600/337million) = 28.7$ bits

If he knows John's DOB, he has $log_2(1/365) = 8.51$ bits

If he knows John is male, he has $log_2(1/2) = 1$ bit

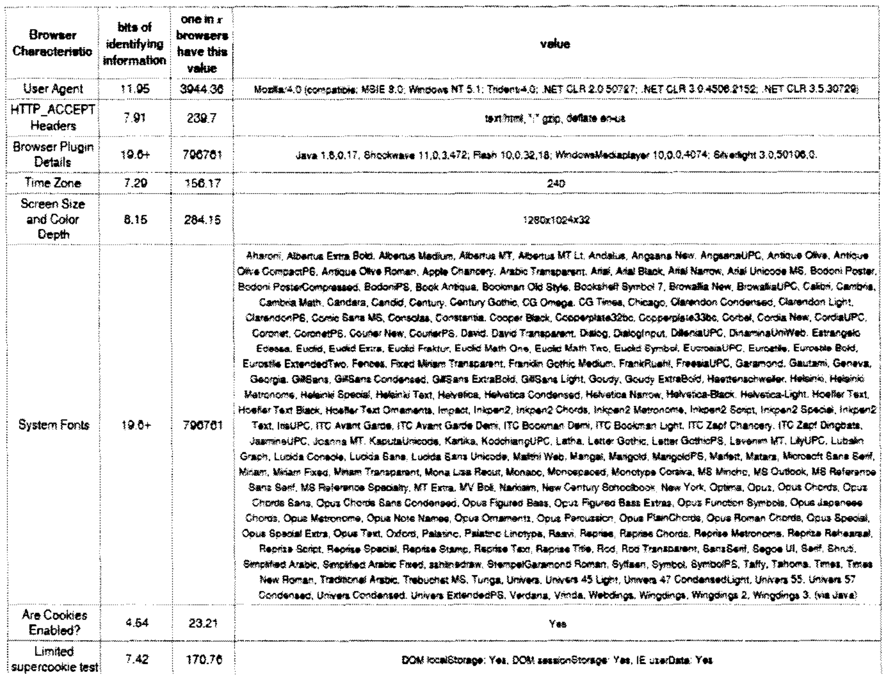
Altogether he has total $= (28.7 + 8.51 + 1) = 38.4$, More than enough information.

In the CCFDP system, the current javaScript collects nearly 60 attributes of the server client communication. But we use less than 15. If we combine, for example (UserAgent + Browser Pluggins + Screen size+ Color depth + System fonts + geo location + etc) information for each client based on the above formalization we can generate a unique identification for each client. For example see Figure 9.11 and Figure 9.12. They show data collected through the javaScript in two different computers. We can easily see that UserAgent combined with Fonts will easily make a unique identity for a computer. We call this technique as browser finger printing. [1]

9.2.2 Use of Super Cookies to track users instead of HTTP cookies

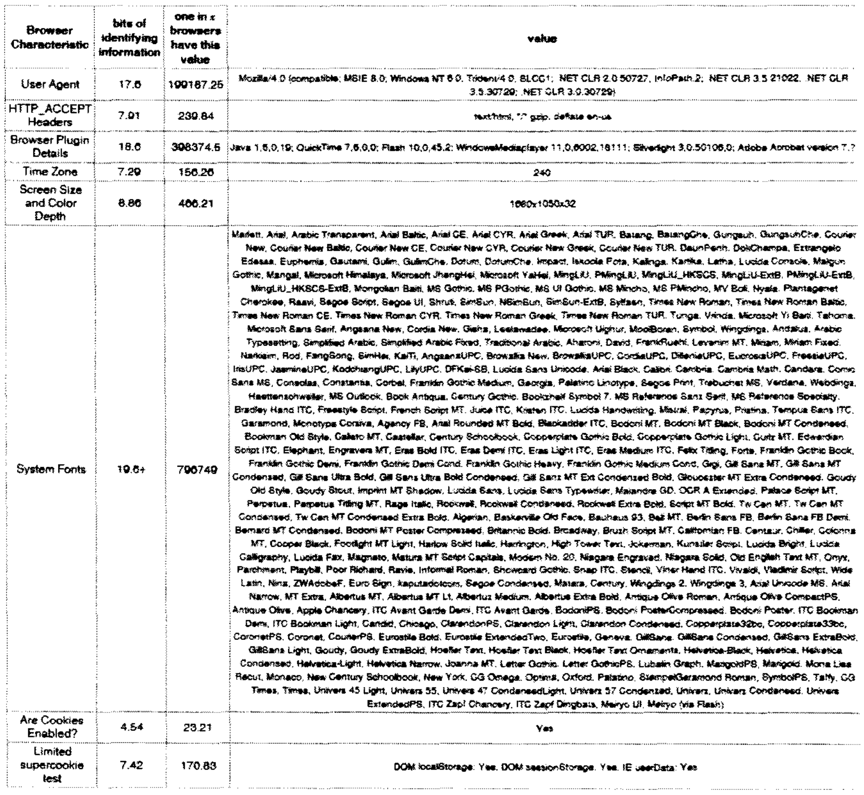
One of the disadvantages in the CCFDP system is the difficulty of tracking users if they do not allow or delete cookies. The "cookie" we used in CCFDP system is an HTTP type traditional cookie. These cookies are stored inside the browser control. For example, in Internet Explorer you can manage the cookies by going to Tools and then to Preferences. So that a user can easily either delete or block HTTP cookies. The only solution to overcome this problem is to store cookies somewhere outside the browser control. A solution to this problem is use of "super cookies". A super cookie is also known as "Flash cookie" or "Shared Objects", a kind of cookie maintained by the Adobe Flash plug-in on behalf of Flash applications embedded in web pages. Flash cookies can track users in all

[1]Panopticlick inc. has conducted a similar research and you can find it at `https://panopticlick.eff.org/`



Browser Characteristic	bits of identifying information	one in *x* browsers have this value	value
User Agent	11.95	3944.36	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
HTTP_ACCEPT Headers	7.91	239.7	text/html, */* gzip, deflate en-us
Browser Plugin Details	19.6+	796761	Java 1.6.0.17, Shockwave 11,0,3,472; Flash 10,0,32,18; WindowsMediaplayer 10,0,0,4074; Silverlight 3.0,50106,0.
Time Zone	7.29	156.17	240
Screen Size and Color Depth	8.15	284.15	1280x1024x32
System Fonts	19.6+	796761	Aharoni, Albertus Extra Bold, Albertus Medium, Albertus MT, Albertus MT Lt, Andalus, Angsana New, AngsanaUPC, Antique Olive, Antique Olive CompactPS, Antique Olive Roman, Apple Chancery, Arabic Transparent, Arial, Arial Black, Arial Narrow, Arial Unicode MS, Bodoni Poster, Bodoni PosterCompressed, BodoniPS, Book Antiqua, Bookman Old Style, Bookshelf Symbol 7, Browallia New, BrowalliaUPC, Calibri, Cambria, Cambria Math, Candara, Candid, Century, Century Gothic, CG Omega, CG Times, Chicago, Clarendon Condensed, Clarendon Light, ClarendonPS, Comic Sans MS, Consolas, Constantia, Cooper Black, Copperplate32bc, Copperplate33bc, Corbel, Cordia New, CordiaUPC, Coronet, CoronetPS, Courier New, CourierPS, David, David Transparent, Dialog, DialogInput, DilleniaUPC, DinaminaUniWeb, Estrangelo Edessa, Euclid, Euclid Extra, Euclid Fraktur, Euclid Math One, Euclid Math Two, Euclid Symbol, EucrosiaUPC, Eurostile, Eurostile Bold, Eurostile ExtendedTwo, Fences, Fixed Miriam Transparent, Franklin Gothic Medium, FrankRuehl, FreesiaUPC, Garamond, Gautami, Geneva, Georgia, GillSans, GillSans Condensed, GillSans ExtraBold, GillSans Light, Goudy, Goudy ExtraBold, Haettenschweiler, Helsinki, Helsinki Metronome, Helsinki Special, Helsinki Text, Helvetica, Helvetica Condensed, Helvetica Narrow, Helvetica-Black, Helvetica-Light, Hoefler Text, Hoefler Text Black, Hoefler Text Ornaments, Impact, Inkpen2, Inkpen2 Chords, Inkpen2 Metronome, Inkpen2 Script, Inkpen2 Special, Inkpen2 Text, IrisUPC, ITC Avant Garde, ITC Avant Garde Demi, ITC Bookman Demi, ITC Bookman Light, ITC Zapf Chancery, ITC Zapf Dingbats, JasmineUPC, Joanna MT, KaputaUnicode, Kartika, KodchiangUPC, Latha, Letter Gothic, Letter GothicPS, Levenim MT, LilyUPC, Lubalin Graph, Lucida Console, Lucida Sans, Lucida Sans Unicode, Malithi Web, Mangal, Marigold, MarigoldPS, Marlett, Mataro, Microsoft Sans Serif, Miriam, Miriam Fixed, Miriam Transparent, Mona Lisa Recut, Monaco, Monospaced, Monotype Corsiva, MS Mincho, MS Outlook, MS Reference Sans Serif, MS Reference Specialty, MT Extra, MV Boli, Narkisim, New Century Schoolbook, New York, Optima, Opus, Opus Chords, Opus Chords Sans, Opus Chords Sans Condensed, Opus Figured Bass, Opus Figured Bass Extras, Opus Function Symbols, Opus Japanese Chords, Opus Metronome, Opus Note Names, Opus Ornaments, Opus Percussion, Opus PlainChords, Opus Roman Chords, Opus Special, Opus Special Extra, Opus Text, Oxford, Palatino, Palatino Linotype, Raavi, Reprise, Reprise Chords, Reprise Metronome, Reprise Rehearsal, Reprise Script, Reprise Special, Reprise Stamp, Reprise Text, Reprise Title, Rod, Rod Transparent, SansSerif, Segoe UI, Serif, Shruti, Simplified Arabic, Simplified Arabic Fixed, sahinadraw, StempelGaramond Roman, Sylfaen, Symbol, SymbolPS, Taffy, Tahoma, Times, Times New Roman, Traditional Arabic, Trebuchet MS, Tunga, Univers, Univers 45 Light, Univers 47 CondensedLight, Univers 55, Univers 57 Condensed, Univers Condensed, Univers ExtendedPS, Verdana, Vrinda, Webdings, Wingdings, Wingdings 2, Wingdings 3, (via Java)
Are Cookies Enabled?	4.54	23.21	Yes
Limited supercookie test	7.42	170.76	DOM localStorage: Yes, DOM sessionStorage: Yes, IE userData: Yes

Figure 9.11: Browser Signature of PC 1



Browser Characteristic	bits of identifying information	one in *x* browsers have this value	value
User Agent	17.6	199187.25	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; InfoPath.2; .NET CLR 3.5.21022; .NET CLR 3.5.30729; .NET CLR 3.0.30729)
HTTP_ACCEPT Headers	7.91	239.84	text/html, */* gzip, deflate en-us
Browser Plugin Details	18.6	398374.5	Java 1,6,0,19; QuickTime 7,6,0,0; Flash 10,0,45,2; WindowsMediaplayer 11,0,6002,18111; Silverlight 3,0,50106,0; Adobe Acrobat version 7.?
Time Zone	7.29	156.26	240
Screen Size and Color Depth	8.86	466.21	1680x1050x32
System Fonts	19.6+	796749	Marlett, Arial, Arabic Transparent, Arial Baltic, Arial CE, Arial CYR, Arial Greek, Arial TUR, Batang, BatangChe, Gungsuh, GungsuhChe, Courier New, Courier New Baltic, Courier New CE, Courier New CYR, Courier New Greek, Courier New TUR, DaunPenh, DokChampa, Estrangelo Edessa, Euphemia, Gautami, Gulim, GulimChe, Dotum, DotumChe, Impact, Iskoola Pota, Kalinga, Kartika, Latha, Lucida Console, Malgun Gothic, Mangal, Microsoft Himalaya, Microsoft JhengHei, Microsoft YaHei, MingLiU, PMingLiU, MingLiU_HKSCS, MingLiU-ExtB, PMingLiU-ExtB, MingLiU_HKSCS-ExtB, Mongolian Baiti, MS Gothic, MS PGothic, MS UI Gothic, MS Mincho, MS PMincho, MV Boli, Nyala, Plantagenet Cherokee, Raavi, Segoe Script, Segoe UI, Shruti, SimSun, NSimSun, SimSun-ExtB, Sylfaen, Times New Roman, Times New Roman Baltic, Times New Roman CE, Times New Roman CYR, Times New Roman Greek, Times New Roman TUR, Tunga, Vrinda, Microsoft Yi Baiti, Tahoma, Microsoft Sans Serif, Angsana New, Cordia New, Gisha, Leelawadee, Microsoft Uighur, MoolBoran, Symbol, Wingdings, Andalus, Arabic Typesetting, Simplified Arabic, Simplified Arabic Fixed, Traditional Arabic, Aharoni, David, FrankRuehl, Levenim MT, Miriam, Miriam Fixed, Narkisim, Rod, FangSong, SimHei, KaiTi, AngsanaUPC, Browallia New, BrowalliaUPC, CordiaUPC, DilleniaUPC, EucrosiaUPC, FreesiaUPC, IrisUPC, JasmineUPC, KodchiangUPC, LilyUPC, DFKai-SB, Lucida Sans Unicode, Arial Black, Calibri, Cambria, Cambria Math, Candara, Comic Sans MS, Consolas, Constantia, Corbel, Franklin Gothic Medium, Georgia, Palatino Linotype, Segoe Print, Trebuchet MS, Verdana, Webdings, Haettenschweiler, MS Outlook, Book Antiqua, Century Gothic, Bookshelf Symbol 7, MS Reference Sans Serif, MS Reference Specialty, Bradley Hand ITC, Freestyle Script, French Script MT, Juice ITC, Kristen ITC, Lucida Handwriting, Mistral, Papyrus, Pristina, Tempus Sans ITC, Garamond, Monotype Corsiva, Agency FB, Arial Rounded MT Bold, Blackadder ITC, Bodoni MT, Bodoni MT Black, Bodoni MT Condensed, Bookman Old Style, Calisto MT, Castellar, Century Schoolbook, Copperplate Gothic Bold, Copperplate Gothic Light, Curlz MT, Edwardian Script ITC, Elephant, Engravers MT, Eras Bold ITC, Eras Demi ITC, Eras Light ITC, Eras Medium ITC, Felix Titling, Forte, Franklin Gothic Book, Franklin Gothic Demi, Franklin Gothic Demi Cond, Franklin Gothic Heavy, Franklin Gothic Medium Cond, Gigi, Gill Sans MT, Gill Sans MT Condensed, Gill Sans Ultra Bold, Gill Sans Ultra Bold Condensed, Gill Sans MT Ext Condensed Bold, Gloucester MT Extra Condensed, Goudy Old Style, Goudy Stout, Imprint MT Shadow, Lucida Sans, Lucida Sans Typewriter, Maiandra GD, OCR A Extended, Palace Script MT, Perpetua, Perpetua Titling MT, Rage Italic, Rockwell, Rockwell Condensed, Rockwell Extra Bold, Script MT Bold, Tw Cen MT, Tw Cen MT Condensed, Tw Cen MT Condensed Extra Bold, Algerian, Baskerville Old Face, Bauhaus 93, Bell MT, Berlin Sans FB, Berlin Sans FB Demi, Bernard MT Condensed, Bodoni MT Poster Compressed, Britannic Bold, Broadway, Brush Script MT, Californian FB, Centaur, Chiller, Colonna MT, Cooper Black, Footlight MT Light, Harlow Solid Italic, Harrington, High Tower Text, Jokerman, Kunstler Script, Lucida Bright, Lucida Calligraphy, Lucida Fax, Magneto, Matura MT Script Capitals, Modern No. 20, Niagara Engraved, Niagara Solid, Old English Text MT, Onyx, Parchment, Playbill, Poor Richard, Ravie, Informal Roman, Showcard Gothic, Snap ITC, Stencil, Viner Hand ITC, Vivaldi, Vladimir Script, Wide Latin, Nina, ZWAdobeF, Euro Sign, kaputadotcom, Segoe Condensed, Mataro, Century, Wingdings 2, Wingdings 3, Arial Unicode MS, Arial Narrow, MT Extra, Albertus MT, Albertus MT Lt, Albertus Medium, Albertus Extra Bold, Antique Olive Roman, Antique Olive CompactPS, Antique Olive, Apple Chancery, ITC Avant Garde Demi, ITC Avant Garde, BodoniPS, Bodoni PosterCompressed, Bodoni Poster, ITC Bookman Demi, ITC Bookman Light, Candid, Chicago, ClarendonPS, Clarendon Light, Clarendon Condensed, Copperplate32bc, Copperplate33bc, CoronetPS, Coronet, CourierPS, Eurostile Bold, Eurostile ExtendedTwo, Eurostile, Geneva, GillSans, GillSans Condensed, GillSans ExtraBold, GillSans Light, Goudy, Goudy ExtraBold, Hoefler Text, Hoefler Text Black, Hoefler Text Ornaments, Helvetica-Black, Helvetica, Helvetica Condensed, Helvetica-Light, Helvetica Narrow, Joanna MT, Letter Gothic, Letter GothicPS, Lubalin Graph, MarigoldPS, Marigold, Mona Lisa Recut, Monaco, New Century Schoolbook, New York, CG Omega, Optima, Oxford, Palatino, StempelGaramond Roman, SymbolPS, Taffy, CG Times, Times, Univers 45 Light, Univers 55, Univers 47 CondensedLight, Univers 57 Condensed, Univers, Univers Condensed, Univers ExtendedPS, ITC Zapf Chancery, ITC Zapf Dingbats, Meiryo UI, Meiryo (via Flash)
Are Cookies Enabled?	4.54	23.21	Yes
Limited supercookie test	7.42	170.83	DOM localStorage: Yes, DOM sessionStorage: Yes, IE userData: Yes

Figure 9.12: Browser Signature of PC 2

the ways traditionally HTTP cookies do.

Flash cookies offer several advantages that lead to more persistence than standard HTTP cookies. Flash cookies can contain up to 100KB of information by default (HTTP cookies only store 4KB). Flash cookies do not have expiration dates by default, whereas HTTP cookies expire at the end of a session unless programmed to live longer by the domain setting the cookie. Flash cookies are stored in a different location than HTTP cookies, thus users may not know what files to delete in order to eliminate them. Additionally, they are stored so that different browsers and stand-alone Flash widgets installed on a given computer access the same persistent Flash cookies. Web browsers do not directly allow users to view or delete the cookies stored by a flash application. Users are not notified when such cookies are set. Flash cookies are not controlled by the browser. Thus erasing HTTP cookies, clearing history, erasing the cache, or choosing a delete private data option within the browser does not affect Flash cookies. Even the "Private Browsing" mode recently added to most browsers such as Internet Explorer 8 and Firefox 3 still allows Flash cookies to operate fully and track the user. These differences make Flash cookies a more resilient technology for tracking than HTTP cookies [Soltani et al., 2009].

The following example shows how to set a super cookie and how to retrieve it.

9.2.3 Creating the Flash Cookie

(i) Create a Flash Cookie with the getLocal method of Shared Object.

The sample movie sets a variable (myLocalSO) and assigns a Shared Object with the name of "flashcookie" with the following ActionScript:

```
myLocalso = sharedobject.getLocal("flashcookie");
```

If a Shared Object with the name "flashcookie" does not already exist, then the Macromedia Flash Player will create a Shared Object with that name.

(ii) Create a Flash Cookie with the localPath option.

An optional parameter called localPath can also be specified for the Shared Object. This localPath parameter allows some control over where the Shared Object is stored on the client machine. This path match or be contained within the URL that the SWF came from. Therefore, if the sample movie that creates the Shared Object on the client machine is at

```
http://www.mydomain.com/movies/mymovie.swf
```

then the localPath parameter can be set to

```
http://www.mydomain.com/movies/mymovie.swf,/,/movies, or/movies/
    mymovie.swf.
```

The code would look like this:

```
myLocalso = sharedobject.getLocal("flashcookie","/movies/mymovie.swf")
    ;
```

This is useful when more than one Flash Cookies are used on a site.

9.2.3.1 Setting the value of the Flash Cookie

Information is stored in the Shared Object by assigning attributes to the data property of the Shared Object. In this example, the user name entered in the text field is stored in the Shared Object by assigning a name attribute to the data property of the local shared object and setting it equal to the contents of the text field as follows:

```
//set the variable ''name'' equal to the text property
//of the textfield ''userName''
myLocal_so.data.name = userName.text;

//increase the variable counter by one for each visit
myLocal_so.data.counter++;
```

The data is written to the Shared Object when the movie is removed from the Macromedia Flash Player. To write the data immediately the method flush can be used as follows:

```
myLocal_so.flush();
```

9.2.3.2 Return the value of the Flash Cookie

When a user returns to the page the Shared Object is read and its values are displayed.

```
userName.text = myLocal_so.data.name;
numVisits.text = ``You have been here'' + myLocal_so.data.counter +``times.
    ''
```

Because the Shared Object "flashcookie" has already been created on the client machine,myLocalSO = sharedobject.getLocal("flashcookie");will get the data from the Shared Object, which can be used to display the user name and number of visits. More information can be found at [Adobe, 2010].

REFERENCES

[Adobe, 2010] Adobe (2010). What is a local Shared Object? http://kb2.adobe.com/cps/161/tn16194.html accessed on 03/29/2010.

[AnupamL et al., 1999] AnupamL, V., Mayera, A., Nissimb, K., Pinkasb, B., and Reitera, M. (1999). On the security of pay-per-click and other Web advertising schemes. *Computer Networks*, 31:1091–1100.

[Ayel and Laurent, 1991] Ayel, M. and Laurent, J. (1991). *Validation, verification and test of knowledge-based systems.* John Wiley & Sons, Inc. New York, NY, USA.

[Banerjee and Ghosh, 2001] Banerjee, A. and Ghosh, J. (2001). Clickstream clustering using weighted longest common subsequences. In *Proc. of the Workshop on Web Mining, SIAM Conference on Data Mining*, pages 33–40. Citeseer.

[Belady, 2010] Belady, L. (2010). A study of replacement algorithms for a virtual-storage computer. *IBM systems journal*, 5(2):78–101.

[Ben-David, 2008] Ben-David, A. (2008). Rule effectiveness in rule-based systems: A credit scoring case study. *Expert Systems with Applications*, 34(4):2783–2788.

[Bloom, 1970] Bloom, B. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

[Bonomi et al., 2006] Bonomi, F., Mitzenmacher, M., Panigrah, R., Singh, S., and Varghese, G. (2006). Beyond bloom filters: from approximate membership checks to approximate state machines. *ACM SIGCOMM Computer Communication Review*, 36(4):315–326.

[Broder and Mitzenmacher, 2004] Broder, A. and Mitzenmacher, M. (2004). Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509.

[Broder et al., 2003] Broder, A., Najork, M., and Wiener, J. (2003). Efficient URL caching for world wide web crawling. In *Proceedings of the 12th international conference on World Wide Web*, pages 679–689. ACM.

[Buchner et al., 1999] Buchner, A., Baumgarten, M., Anand, S., Mulvenna, M., and Hughes, J. (1999). Navigation pattern discovery from internet data. In *WEBKDD99*, pages 74–91. Citeseer.

[Cheng et al., 2005] Cheng, K., Xiang, L., Iwaihara, M., Xu, H., and Mohania, M. (2005). Time-decaying Bloom filters for data streams with skewed distributions. In *Proceedings of the 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications*, page 69. IEEE Computer Society.

[Cohen and Strauss, 2006] Cohen, E. and Strauss, M. (2006). Maintaining time-decaying stream aggregates. *Journal of Algorithms*. 59(1):19–36.

[Cooley et al., 1999] Cooley, R., Mobasher, B., Srivastava, J., et al. (1999). Data preparation for mining world wide web browsing patterns. *Knowl. Inf. Syst.*, 1(1):5–32.

[Corbato, 1968] Corbato, F. (1968). A paging experiment with the multics system.

[Cuenca-Acuna and Nguyen, 2010] Cuenca-Acuna, F. and Nguyen, T. (2010). Text-based content search and retrieval in ad-hoc p2p communities. *Web Engineering and Peer-to-Peer Computing*, pages 220–234.

[Dai and Khorram, 1999] Dai, X. and Khorram, S. (1999). Data fusion using artificial neural networks: a case study on multitemporal change analysis. *Computers, Environment and Urban Systems*, 23(1):19–31.

[Daswani and Stoppelman, 2007] Daswani, N. and Stoppelman, M. (2007). The anatomy of Clickbot. A. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, page 11. USENIX Association.

[Database, 2011] Database, R. (03/21/2011). Robots database, http://www.robotstxt.org/db.html.

[Deng and Rafiei, 2006] Deng, F. and Rafiei, D. (2006). Approximately detecting duplicates for streaming data using stable bloom filters. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, page 36. ACM.

[Dikaiakos et al., 2005] Dikaiakos, M., Stassopoulou, A., and Papageorgiou, L. (2005). An investigation of web crawler behavior: characterization and metrics. *Computer Communications*, 28(8):880–897.

[DOD, 1991] DOD (1991). DOD, Data fusion lexicon, Data Fusion Subpanel of the Joint Directors of Laboratories, Technical Panel for C3, Environmental Research Inst. Of Michigan Arlington VA.

[Durrant-Whyte, 1987] Durrant-Whyte, H. (1987). Integration, coordination and control of multi-sensor robot systems. *Dissertation Abstracts International*, 47(10).

[Edelman et al., 2007] Edelman, B., Ostrovsky, M., and Schwarz, M. (2007). Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *The American Economic Review*, pages 242–259.

[Elmagarmid et al., 2007] Elmagarmid, A., Ipeirotis, P., and Verykios, V. (2007). Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, pages 1–16.

[Estan and Varghese, 2003] Estan, C. and Varghese, G. (2003). New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS)*, 21(3):270–313.

[Fan et al., 2000] Fan, L., Cao, P., Almeida, J., and Broder, A. (2000). Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking (TON)*, 8(3):293.

[Friedman et al., 1997] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, 29(2):131–163.

[Fusheng and Feng, 2008] Fusheng, Z. and Feng, D. (2008). Application of DS evidence theory in flow regime identification of two-phase horizontal pipe flow. In *27th Chinese Control Conference*, pages 758–762.

[Gandhi et al., 2006] Gandhi, M., Jakobsson, M., and Ratkiewicz, J. (2006). Badvertisements: Stealthy click-fraud with unwitting accessories. *Journal of Digital Forensic Practice*, 1(2):131–142.

[Garera et al., 2007] Garera, S., Provos, N., Chew, M., and Rubin, A. (2007). A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*, page 8. ACM.

[Ge and Kantardzic, 2006] Ge, L. and Kantardzic, M. (2006). Real-time click fraud detecting and blocking system. US Patent App. 11/413,983.

[Goldman, 2007] Goldman, E. (May 2007). Click Fraud. In *Proceedings of the 20th Annual Technology and Computer Law Conference.*

[Gonnet and Baeza-Yates, 1991] Gonnet, G. and Baeza-Yates, R. (1991). *Handbook of algorithms and data structures: in Pascal and C.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[Goodman, 2005] Goodman, J. (2005). Pay-per-percentage of impressions: an advertising method that is highly robust to fraud. In *Workshop on Sponsored Search Auctions*. Citeseer.

[Gupta et al., 1991] Gupta, U., Press, I. C. S., of Electrical, I., and Engineers, E. (1991). *Validating and verifying knowledge-based systems.* IEEE Computer Society Press.

[Haddadi, 2010] Haddadi, H. (2010). Fighting Online Click-Fraud Using Bluff Ads. *Arxiv preprint arXiv:1002.2353.*

[Hadjinian et al., 2006] Hadjinian, D. et al. (2006). Clicking away the competition: The legal ramifications of click fraud for companies that offer pay per click advertising services. *Shidler JL Com. & Tech.*, 3:5–16.

[Hager et al., 1993] Hager, G., Engelson, S., and Atiya, S. (1993). On comparing statistical and set-based methods in sensor data fusion. In *IEEE International Conference on Robot Automation.*

[Hawkins, 1980] Hawkins, D. (1980). *Identification of outliers.* Chapman & Hall.

[IAB, 2010] IAB (2010). IAB Internet Advertising Revenue Report conducted by PricewaterhouseCoopers (PWC). http://www.iab.net/insightsresearch/947883/adrevenuereport, accessed on 03/24/2001.

[(IETF), 1999] (IETF), I. E. T. F. (1999). RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. http://www.faqs.org/rfcs/rfc2616.html.

[Immorlica et al., 2005] Immorlica, N., Jain, K., Mahdian, M., and Talwar, K. (2005). Click fraud resistant methods for learning click-through rates. *Lecture Notes In Computer Science*, 3828:34–45.

[iProspect, 2004] iProspect (2004). iProspect search engine user attitude survey. http://www.iprospect.com/premiumPDFs/iProspectSurveyComplete.pdf, accessed on 03/01/2010.

[Jakobsson et al., 1999] Jakobsson, M., MacKenzie, P., and Stern, J. (1999). Secure and lightweight advertising on the Web. *Computer Networks-the International Journal of Computer and Telecommunications Networkin*, 31(11):1101–1110.

[Janez et al., 2000] Janez, F., Goretta, O., and Michel, A. (2000). Automatic map updating by fusion of multispectral images in the Dempster-Shafer framework. In *Proceedings of SPIE*, volume 4115, page 245.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In Bratko, I. and Dzeroski, S., editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL. Morgan Kaufmann Publishers, San Francisco, US.

[Juels et al., 2007] Juels, A., Laboratories, R., Stamm. S.. and Jakobsson, M. (2007). Combating Click Fraud via Premium Clicks. page 1726.

[Kantardzic et al., 2010a] Kantardzic, M., Walgampaya, C., and Emara, W. (2010a). Click fraud prevention in pay-per-click model: Learning through multi-model evidence fusion. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, pages 20–27. IEEE.

[Kantardzic et al., 2008] Kantardzic, M., Walgampaya, C., Wenerstrom, B., Lozitskiy, O., Higgins, S., and King, D. (2008). Improving Click Fraud Detection by Real Time Data Fusion. In *IEEE International Symposium on Signal Processing and Information Technology, 2008. ISSPIT 2008*, pages 69–74.

[Kantardzic et al., 2010b] Kantardzic, M., Walgampaya, C., Yampolskiy, R., and Woo, R. (2010b). Click Fraud Prevention via multimodal evidence fusion by Dempster-Shafer theory. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, pages 26–31. IEEE.

[Kantardzic et al., 2009] Kantardzic, M., Wenerstrom, B., Walgampaya, C., Lozitskiy, O., Higgins, S., and King, D. (2009). Time and Space Contextual Information Improves Click Quality Estimation. *e-Commerce 2009*, page 123.

[Kintana et al., 2009] Kintana, C., Turner, D., Pan, J., Metwally, A., Daswani, N., Chin, E., and Bortz, A. (2009). The Goals and Challenges of Click Fraud Penetration Testing Systems.

[Kondo and Sato, 2007] Kondo, S. and Sato, N. (2007). Botnet traffic detection techniques by c&c session classification using svm. *Advances in Information and Computer Security*, pages 91–104.

[Kumar et al., 2006] Kumar, A., Xu, J., and Wang, J. (2006). Space-code bloom filter for efficient per-flow traffic measurement. *Selected Areas in Communications, IEEE Journal on*, 24(12):2327–2339.

[Lambert, 2009] Lambert, D. (2009). A blueprint for higher-level fusion systems. *Information Fusion*, 10(1):6–24.

[Lee, 2009] Lee, K. (2009). *The Truth About Pay-per-click Search Advertising.* FT Press.

[Li et al., 2000] Li, J., Jannotti, J., De Couto, D., Karger, D., and Morris, R. (2000). A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 120–130. ACM.

[Lu and Yu, 2006] Lu, W. and Yu, S. (2006). Web robot detection based on hidden Markov model. In *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, volume 3, pages 1806–1810. IEEE.

[Ma et al., 2009] Ma, J., Saul, L., Savage, S., and Voelker, G. (2009). Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254. ACM.

[Mahdian, 2006] Mahdian, M. (2006). Theoretical challenges in the design of advertisement auctions. In *The Capital Area Theory Symposia*. University of Maryland.

[McGrath and Gupta, 2008] McGrath, D. and Gupta, M. (2008). Behind phishing: an examination of phisher modi operandi. In *Proc. of the USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*.

[Mehta et al., 2007] Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. (2007). Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22.

[Menezes et al., 1997] Menezes, A., Van Oorschot, P., and Vanstone, S. (1997). *Handbook of applied cryptography*. CRC.

[Metwally et al., 2005a] Metwally, A., Agrawal, D., and Abbadi, A. (2005a). Using association rules for fraud detection in web advertising networks. In *Proceedings of the 31st international conference on Very large data bases*, page 180. VLDB Endowment.

[Metwally et al., 2005b] Metwally, A., Agrawal, D., and El Abbadi, A. (2005b). Duplicate detection in click streams. In *Proceedings of the 14th international conference on World Wide Web*, pages 12–21. ACM New York, NY, USA.

[Metwally et al., 2007] Metwally, A., Agrawal, D., and El Abbadi, A. (2007). Detectives: detecting coalition hit inflation attacks in advertising networks streams. In *Proceedings of the 16th international conference on World Wide Web*. ACM.

[Metwally et al., 2006] Metwally, A., El Abbadi, D., Zheng, Q., and FastClick, I. (2006). Hide and Seek: Detecting Hit Inflation Fraud in Streams of Web Advertising Networks.

[Microsoft, 2010] Microsoft (2010). MSDN Library. http://msdn.microsoft.com/en-us/library accessed on 04/14/2010.

[NetMosaics, 2009] NetMosaics (2009). NetMosaics Inc. Internal Documentation.

[nielsenwire, 2010] nielsenwire (2010). Led by Facebook, Twitter, Global Time Spent on Social Media Sites up 82 percent Year over Year. http://blog.nielsen.com/, accessed on 03/21/2001.

[Ouyang et al., 2008] Ouyang, N., Liu, Z., and Kang, H. (2008). A method of Distributed Decision Fusion based on SVM and DS evidence theory. In *5th International Conference on Visual Information Engineering*, pages 261–264.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

[Pedersen, 2008] Pedersen, J. (2008). Introduction to the Special Issue: Click Fraud. *International Journal of Electronic Commerce*, 13(2):5–8.

[Pirolli et al., 1996] Pirolli, P., Pitkow, J., and Rao, R. (1996). Silk from a sow's ear: extracting usable structures from the Web. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, pages 118–125. ACM.

[Platt, 1999] Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

[Provos et al., 2008] Provos, N., Mavrommatis, P., Rajab, M., and Monrose, F. (2008). All your iframes point to us. In *Proceedings of the 17th conference on Security symposium*, pages 1–15. USENIX Association.

[Reynolds and Vahdat, 2003] Reynolds, P. and Vahdat, A. (2003). Efficient peer-to-peer keyword searching. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, pages 21–40. Springer-Verlag New York, Inc.

[Rowstron and Druschel, 2001] Rowstron, A. and Druschel, P. (2001). Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. *ACM SIGOPS Operating Systems Review*, 35(5):188–201.

[Shafer, 1976] Shafer, G. (1976). *A mathematical theory of evidence.* Princeton university press Princeton, NJ.

[Shah, 2005] Shah, S. (2005). Browser identification for web applications. *Net Square.*

[Shen and Zhang, 2008] Shen, H. and Zhang, Y. (2008). Improved approximate detection of duplicates for data streams over sliding windows. *Journal of Computer Science and Technology*, 23(6):973–987.

[Solaiman et al., 1999] Solaiman, B., Pierce, L., and Ulaby, F. (1999). Multisensor data fusion using fuzzy concepts: application to land-cover classification using ERS-1/JERS-1 SAR composites. *IEEE Transactions on Geoscience and Remote Sensing*, 37(3):1316–1326.

[Soltani et al., 2009] Soltani, A., Canty, S., Mayo, Q., Thomas, L., and Hoofnagle, C. (2009). Flash Cookies and Privacy.

[Song et al., 2005] Song, H., Dharmapurikar, S., Turner, J., and Lockwood, J. (2005). Fast hash table lookup using extended bloom filter: an aid to network processing. *SIGCOMM Comput. Commun. Rev.*, 35:181–192.

[Soubusta, 2008] Soubusta, S. (2008). On Click Fraud. *Information Wissenschaft Und Praxis*, 59(2):136.

[Stassopoulou and Dikaiakos, 2009] Stassopoulou, A. and Dikaiakos, M. (2009). Web robot detection: A probabilistic reasoning approach. *Computer Networks*, 53(3):265–278.

[Strayer et al., 2008] Strayer, W., Lapsely, D., Walsh, R., and Livadas, C. (2008). Botnet detection based on network behavior. *Botnet Detection*, pages 1–24.

[Strayer et al., 2006] Strayer, W., Walsh, R., Livadas, C., and Lapsley, D. (2006). Detecting botnets with tight command and control. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pages 195–202. Citeseer.

[Sweeney, 2002] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5):557–570.

[Talbot and Osborne, 2007] Talbot, D. and Osborne, M. (2007). Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 468–476.

[Tan and Kumar, 2002] Tan, P. and Kumar, V. (2002). Discovery of web robot sessions based on their navigational patterns. *Data Mining and Knowledge Discovery*, 6(1):9–35.

[Tian et al., 2005] Tian, J., Zhao, W., Du, R., and Zhang, Z. (2005). DS evidence theory and its data fusion application in intrusion detection. *Lecture notes in computer science*, 3802.

[TNS, 2010] TNS (2010). TNS Media Intelligence in the News. http://www.tns-mi.com/newsindex.htm, accessed on 03/01/2010.

[Tuzhilin, 2006] Tuzhilin, A. (2006). The lanes gifts v. google report.

[Wald, 2001] Wald, L. (2001). The present achievements of the EARSeL-SIG'Data Fusion'. In *A Decade of Trans-European Remote Sensing Cooperation: Proceedings of the 20th Earsel Symposium, Dresden, Germany, 14-16 June 2000*, page 263. Taylor & Francis.

[Walgampaya and Kantardzic, 2010] Walgampaya, C. and Kantardzic, M. (2010). Net-mosaics. In *Internal Documentation.*

[Walgampaya et al., 2010] Walgampaya, C., Kantardzic, M., and Yampolskiy, R. (2010). Real Time Click Fraud Prevention using multi-level Data Fusion. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1.

[Waltz, 1998] Waltz, E. (1998). Information understanding: integrating data fusion and data mining processes. In *IEEE International Symposium on Circuits and Systems*, pages 553–556. Institute of Electrical Engineers Inc (IEEE).

[Waltz and Llinas, 1990] Waltz, E. and Llinas, J. (1990). *Multisensor data fusion*. Artech House Boston, London.

[Wu and Lee, 2002] Wu, C. and Lee, S. (2002). Enhanced high-level Petri nets with multiple colors for knowledge verification/validation of rule-based expert systems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 27(5):760–773.

[Wu et al., 2002] Wu, H., Siegel, M., Stiefelhagen, R., and Yang, J. (2002). Sensor fusion using Dempster-Shafer theory. In *IEEE Instrumentation and Measurement Technology Conference Proceedings*, volume 1, pages 7–12. Citeseer.

[Yukun et al., 2007] Yukun, C., Xicai, S., and Zhigang, L. (2007). Research on Kalman-filter based multisensor data fusion. *Journal of Systems Engineering and Electronics*, 18(3):497–502.

[Yukun et al., 2009] Yukun, C., Xicai, S., and Zhigang, L. (2009). A Data Fusion Based Intrusion Detection Model. *First International Workshop on Education Technology and Computer Science*, 1:1017–1021.

[Zeng and Xu, 2008] Zeng, D. and Xu, J. (2008). Data Fusion for Traffic Incident Detection Using D-S Evidence Theory with Probabilistic SVMs. *Journal of Computers*, 3(10):36–43.

[Zhang et al., 2007] Zhang, Y., Hong, J., and Cranor, L. (2007). Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648. ACM New York, NY, USA.

[Zhang et al., 2009] Zhang, Y., Shen, H., Tian, H., and Zhang, X. (2009). Dynamically Maintaining Duplicate-Insensitive and Time-Decayed Sum Using Time-Decaying Bloom Filter. *Algorithms and Architectures for Parallel Processing*, pages 741–750.

APPENDIX A

ISAPI filter design

As the client computer sends the request to the web server, e. g. Internet Information Service (IIS), the server generates the requested page and sends back the to the client computer. In this step, we create an ISAPI filter to qualify the request as displayed in Figure A.1. As displayed in Figure A.1, we only concern about the successfully generated page, which is the code 200. At the same time, we only qualify pages with text/html response. Other contents, such as image, css, video clip etc are supplementary contents and will be sent back to client computer directly. If the major text/html page is blocked, there should have no following image, css, and video clip etc. request. The system logs to Global Fraudulent Database (GFD), and query for fraud score before the response is sent back to client computer. If the fraud score is higher than the threshold, a warning page will be sent to client computer instead. Otherwise, a unique 128-bit number, Globally Unique Identifier (GUID) will be added to the tracking javascript code as long as the page.

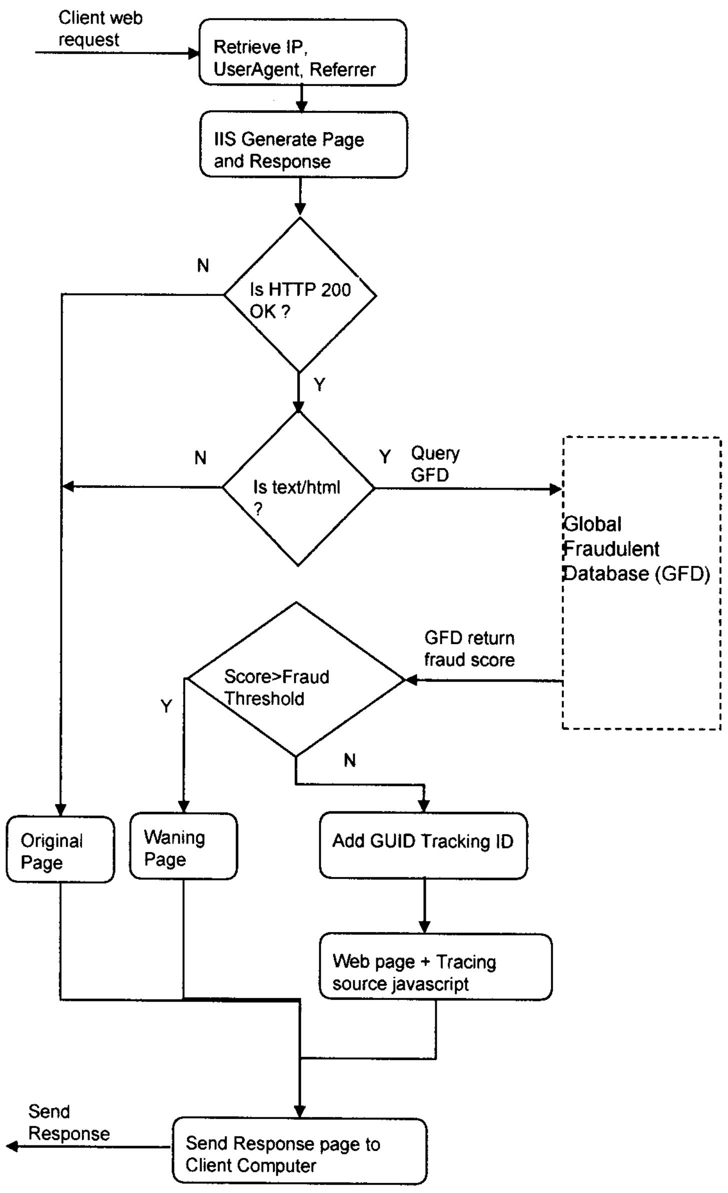

Client web request
Retrieve IP, UserAgent, Referrer
IIS Generate Page and Response
Is HTTP 200 OK ?
N
Y
Is text/html ?
N
Y
Query GFD
Global Fraudulent Database (GFD)
GFD return fraud score
Score>Fraud Threshold
Y
N
Original Page
Waning Page
Add GUID Tracking ID
Web page + Tracing source javascript
Send Response
Send Response page to Client Computer

Figure A.1: ISAPI filter design

APPENDIX B

History.js

```
/*
 * Social Limit - Only the social you care about.
 *
 * Enables your site to know which social bookmarking badges
    to display to your
 * visitors. It tells you all social sites the user has gone
    to, or you can
 * query for a specific one.
 *
 * For example:
 *
 *     var sl = History();
 *     alert( sl.doesVisit(”Digg”) ); // Returns true/false, -1
     if unknown.
 *     var listOfVisitedSites = sl.visitedSites();
 *     var checkedSites = sl.checkedSites();
 *
 * If you want to add more sites to check, you can pass that
    in as a dictionary
 * to History:
 *
 *     var more = { ”Humanized”: ”http://humanized.com”,
 *                   "Azarask.in": ["http://azarask.in", "http
   ://azarask.in/blog"]
 *                 };
 *    var sl = History(more);
 *    alert( sl.doesVisit("Humanized") );
 *
 * For a list of built-in sites, see the sites variable below.
 *
 */

var History = function( moreSites ){

  var sites = {
    "Digg": ["http://digg.com", "http://digg.com/login"],
    "Reddit": ["http://reddit.com", "http://reddit.com/new/",
       "http://reddit.com/controversial/", "http://reddit.com/
       top/", "http://reddit.com/r/reddit.com/", "http://
       reddit.com/r/programming/"],
    "StumbleUpon": ["http://stumbleupon.com"],
    "Yahoo Buzz": ["http://buzz.yahoo.com"],
    "Facebook": ["http://facebook.com/home.php", "http://
       facebook.com", "https://login.facebook.com/login.php"],
    "Del.icio.us": ["https://secure.del.icio.us/login", "http
       ://del.icio.us/"],
    "MySpace": ["http://www.myspace.com/"],
    "Technorati": ["http://www.technorati.com"],
    "Newsvine": ["https://www.newsvine.com", "https://www.
       newsvine.com/_tools/user/login"],
    "Songza": ["http://songza.com"],
    "Slashdot": ["http://slashdot.org/"],
"Ma.gnolia": ["http://ma.gnolia.com/"],
"Blinklist": ["http://www.blinklist.com"],
"Furl": ["http://furl.net", "http://furl.net/members/login
    "],
"Mister Wong": ["http://www.mister-wong.com"],
"Current": ["http://current.com", "http://current.com/
    login.html"],
"Menaeme": ["http://meneame.net", "http://meneame.net/
    login.php"],
"Oknotizie": ["http://oknotizie.alice.it", "http://
    oknotizie.alice.it/login.html.php"],
"Diigo": ["http://www.diigo.com/", "https://secure.diigo.
    com/sign-in"],
"Funp": ["http://funp.com", "http://funp.com/account/
    loginpage.php"],
"Blogmarks": ["http://blogmarks.net"],
"Yahoo Bookmarks": ["http://bookmarks.yahoo.com"],
"Xanga": ["http://xanga.com"],
"Blogger": ["http://blogger.com"],
"Last.fm": ["http://www.last.fm/", "https://www.last.fm/
    login/"],
"N4G": ["http://www.n4g.com"],
"Faves": ["http://faves.com", "http://faves.com/home", "
    https://secure.faves.com/signIn"],
"Simpy": ["http://www.simpy.com", "http://www.simpy.com/
    login"],
"Yigg": ["http://www.yigg.de"],
"Kirtsy": ["http://www.kirtsy.com", "http://www.kirtsy.com
    /login.php"],
  "Fark": ["http://www.fark.com", "http://cgi.fark.com/cgi/
      fark/users.pl?self=1"],
  "Mixx": ["https://www.mixx.com/login/dual", "http://www.
      mixx.com"],
  "Google Bookmarks": ["http://www.google.com/bookmarks", "
      http://www.google.com/ig/add?moduleurl=bookmarks.xml&hl
      =en"],
  "Subbmitt": ["http://subbmitt.com/"]
};

for( var site in moreSites ) {
  // If we don't have the site, create the URL list.
  if( typeof( sites[site] ) == "undefined" ) sites[site] =
      [];

  // If the value is string, just push that onto the URL
      list.
  if( typeof( moreSites[site] ) == "string" )
    sites[site].push( moreSites[site] );
  else
    sites[site] = sites[site].concat( moreSites[site] );
}

var visited = {};

function getStyle(el, scopeDoc,styleProp) {
  if (el.currentStyle)
    var y = el.currentStyle[styleProp];
  else if (window.getComputedStyle)
```

```
    var y = scopeDoc.defaultView.getComputedStyle(el,null).
        getPropertyValue(styleProp);
  return y;
}

function remove( el ) {
  el.parentNode.removeChild( el );
}

// Code inspired by:
// bindzus.wordpress.com/2007/12/24/adding-dynamic-contents-
   to-iframes
function createIframe() {
  var iframe = document.createElement("iframe");
  iframe.style.position = "absolute";
  iframe.style.visibility = "hidden";

  document.body.appendChild(iframe);

  // Firefox, Opera
  if(iframe.contentDocument) iframe.doc = iframe.
      contentDocument;
  // Internet Explorer
  else if(iframe.contentWindow) iframe.doc = iframe.
      contentWindow.document;

  // Magic: Force creation of the body (which is null by
      default in IE).
  // Also force the styles of visited/not-visted links.
  iframe.doc.open();
  iframe.doc.write('<style>');
  iframe.doc.write("a{color: #000000; display:none;}");
  iframe.doc.write("a:visited {color: #FF0000; display:
     inline;}");
  iframe.doc.write('</style>');
  iframe.doc.close();

  // Return the iframe: iframe.doc contains the iframe.
  return iframe;
}

var iframe = createIframe();

function embedLinkInIframe( href, text ) {
  var a = iframe.doc.createElement("a");
  a.href = href;
  a.innerHTML = site;
  iframe.doc.body.appendChild( a );
}

for( var site in sites ) {
  var urls = sites[site];
  for( var i=0; i<urls.length; i++ ) {
    // You have to create elements in the scope of the
       iframe for IE.
    embedLinkInIframe( urls[i], site );

    // Automatically try variations of the URLS with and
       without the "www"
    if( urls[i].match(/www\./) ){
      var sansWWW = urls[i].replace(/www\./, "");
      embedLinkInIframe( sansWWW, site );
    } else {
      // 2 = 1 for length of string + 1 for slice offset
      var httpLen = urls[i].indexOf("//") + 2;
      var withWWW = urls[i].substring(0, httpLen ) + "www."
        + urls[i].substring( httpLen );
      embedLinkInIframe( withWWW, site );
    }

  }
}

var links = iframe.doc.body.childNodes;
for( var i=0; i<links.length; i++) {
  // Handle both Firefox/Safari, and IE (respectively)
  var displayValue = getStyle(links[i], iframe.doc, "display
    ");
  var didVisit = displayValue != "none";

  if( didVisit ){
    visited[ links[i].innerHTML ] = true;
  }
}

remove( iframe );

return new (function(){
  var usedSites = [];
  for( var site in visited ){
      usedSites.push( site );
    }

    // Return an array of visited sites.
    this.visitedSites = function() {
      return usedSites;
    }

    // Return true/false. If we didn't check the site, return
       -1.
    this.doesVisit = function( site ) {
      if( typeof( sites[site] ) == "undefined" )
        return -1;
      return typeof( visited[site] ) != "undefined";
    }

    var checkedSites = [];
    for( var site in sites ){
      checkedSites.push( site );
    }
    // Return a list of the sites checked.
    this.checkedSites = function(){
      return checkedSites;
    }
  })();
}
```

CURRICULUM VITAE

Chamila Walgampaya

Candidate for the Degree of

Doctor of Philosophy

Dissertation: CLICK FRAUD: HOW TO SPOT IT, HOW TO STOP IT?

Data Mining Laboratory, Duthie Center for Engineering,
University of Louisville, Louisville, KY 40292, USA.
office: +1-502-852-3626, mobile: +1-502-819-8346.
e-mail: ckwalg01@louisville.edu.

Education:

Completed the requirements for the degree of Doctor of Philosophy, University of Louisville, Louisville, KY, 2007-to present.

Received the M.S. degree from the University of Louisville, Louisville, KY, 2004-2006.

Received the B.S.Eng(Hons) degree in Computer Engineering from the University of Peradeniya, Sri Lanka, 1996-2001.

Selected Publications:

1. **Chamila Walgampaya**, Mehmed Kantardzic, "Cracking The Smart ClickBot," *Submitted for review in the 13th IEEE International Symposium on Web Systems Evolution*, September 30, 2011.

2. **Chamila Walgampaya**, Mehmed Kantardzic, Brent Wenerstrom, "Duplicate Detection in Pay-Per-Click Streams using Temporal Stateful Bloom Filters," *First review received for publication the International Journal of Data Analysis Techniques and Strategies, IJDATS*, 2011.

3. Joung Woo Ryu, Mehmed Kantardzic, **Chamila Walgampaya**, "Building an Ensemble of Classifiers Using Partially Labeled Streaming Data," *First review*

received for publication in the International Journal of Information Sciences, Elsevier, 2011.

4. **Chamila Walgampaya**, Mehmed Kantardzic, Roman Yampolskiy, "Evidence Fusion For Real Time Click Fraud Detection And Prevention," *Accepted for publication in Intelligent Automation and Systems Engineering*, Springer 2011.

5. **Chamila Walgampaya**, Wael Emara, Mehmed Kantardzic, "Validation of Click Fraud Detection Models," *Accepted for publication in 7th International Conference on Machine Learning and Data Mining, MLDM 2011*, August 30-September 3, 2011.

6. Mehmed Kantardzic, **Chamila Walgampaya**, Wael Emara, "Click fraud prevention in pay-per-click model: Learning through multi-model evidence fusion," *International Conference on Machine and Web Intelligence (ICMWI)*,pages 20–27, 2010.

7. Mehmed Kantardzic, Brent Wenerstrom, **Chamila Walgampaya**, Sean Higgins, Darren King, "Click Fraud Detection Using Time and Space Contextual Information,*IADIS International Journal on WWW/Internet*, Volume: VIII,2 , Pages 101-117, 2010.

8. **Chamila Walgampaya**, Mehmed Kantardzic, and Roman Yampolskiy, "Real Time Click Fraud Prevention using multi-level Data Fusion," International Conference on Computer Science and Applications 201 (WCECS 2010), October 20 - 22, 2010, San Francisco, USA , pp514-519.

9. Mehmed Kantardzic, **Chamila Walgampaya**, Roman Yampolskiy, Joung Woo Ryu, "Click Fraud Prevention via Multimodal Evidence Fusion by Dempster-Shafer Algorithm," *2010 IEEE Conference on Multisensor Fusion and Integration (MFI2010)*, Salt Lake City, Utah, USA, September 5-7, 2010.

10. Mehmed Kantardzic, Joung Woo Ryu, **Chamila Walgampaya**, "Building a New Classifier in an Ensemble using Streaming Unlabeled Data", *The 23rd International Conference on Industrial, Engineering & Other Application of Applied Intelligent Systems (IEA-AIE 2010)*, Cordoba, Spain, June 2010.

11. Mehmed Kantardzic, **Chamila Walgampaya**, Darren King, Sean Higgins, Brent Wenerstrom, Chris Simpson, Joung Woo Ryu, "Real Time Click Fraud Detection and Prevention", *The 6th Kentucky Innovation and Entrepreneurship Conference*, Lexington, Kentucky, April 2010.

12. Joung Woo Ryu, Mehmed Kantardzic, **Chamila Walgampaya**, "Ensemble Classifier based on Misclassified Streaming Data," *The Tenth IASTED International Conference on Artificial Intelligence and Applications (AIA 2010)*, Innsbruck, Austria, Feb. 2010.

13. Mehmed Kantardzic, **Chamila Walgampaya**, Brent Wenerstrom, "Improving Click Fraud Detection by Real Time Data Fusion," *In IEEE International*

Symposium on Signal Processing and Information Technology, ISSPIT 2008, pages 6974, 2008.

14. Mehmed Kantardzic, Brent Wenerstrom, **Chamila Walgampaya**, "Time and Space Contextual Information Improves Click Quality Estimation", *e-Commerce 2009*, page 123, 2009.

15. **Chamila Walgampaya**, Mehmed Kantardzic, "Selection of Distributed Sensors in Multiple Time Series Prediction," *In proceedings of the IEEE World Congress on Computational Intelligence*, Vancouver, CA, July 2006.

16. **Chamila Walgampaya**, Mehmed Kantardzic, "Cost-Sensitive Analysis in Multiple Time Series Prediction," *In proceedings of The 2006 International Conference on Data Mining*, Las Vegas, USA, June 2006.

Honours and Awards:

1.Certificate of Merit (Student) for paper title "Real Time Click Fraud Prevention using multi-level Data Fusion," published in the International Conference on Soft Computing and Applications 2010.

2. E-EXPO Student Research Competition, Graduate overall 1st place and Best in CECS Department, 2010.

3. Co-author, Best Paper Award, IADIS International Conference on ecommerce, Portugal, 2009.

4. Grosscurth Scholarship, Speed School of Engineering, University of Louisville, 2007-2009.

5. Fulbright Scholar, University of Louisville, KY, USA, 2004-2006.

6. Ceylon Bank Employees' Union Scholarship, Sri Lanka, 1997-2001.

7. World Prize,Australian Computer Society Inc., Australia, 1997.

Professional Experience:

University of Louisville, KY, USA
    Teaching/Research Assistant, 2004 to present

University of Peradeniya, Peradeniya, Sri Lanka
    Lecturer, 2003 to present