GSWC 2011

Proceedings of

The Sixth Annual Graduate Student Workshop on Computing

October 7th, 2011
Santa Barbara, California

Department of Computer Science
University of California, Santa Barbara
http://www.cs.ucsb.edu

Organized By

Lara Deek, Chair

Steffen Gauglitz, Vice-Chair

Adam Doupé, Industry Liaison

Ludovico Cavedon, Financial Coordinator

Nichole Stockman, Proceedings Coordinator

Bryce Boe, Web Coordinator

Gianluca Stringhini, General Committee

Nan Li, General Committee

Petko Bogdanov, General Committee

Maria Zheleva, General Committee

Ceren Budak, General Committee

Shiyuan Wang, General Committee

Vineeth Kashyap, General Committee

Hassan Wassel, General Committee

Aaron Elmore, General Committee

Sudipto Das, General Committee

Arijit Khan, General Committee

Wim van Dam, Faculty Adviser

Thanks to

Platinum Supporters

Nickel Supporters

Keynote Speakers

Roberto Manduchi, Associate Professor, UCSC

Roberto Manduchi obtained his "Dottorato di Ricerca" in Electrical Engineering from the University of Padova, Italy, in 1993. After a few years roaming around Northern and Southern California in various postdoctoral positions, he landed a job at Apple in the Interactive Media Group. From 1998 to 2001 he was with the Machine Vision Group at the Jet Propulsion Laboratory. In 2001 he joined the faculty at the University of California, Santa Cruz, where he is currently an Associate Professor of Computer Engineering. Roberto's main research interest is in the application of computer vision and sensor processing to assistive technology for people with visual impairment. His research is supported by the NSF and the NIH.

Zoran Dimitrijevic, Software Engineer, Google

Zoran Dimitrijevic has been with Google as a Software Engineer since 2004. While at Google he has been working on Google File System, video streaming, search infrastructure, News Archives, and News. Prior to joining Google, he graduated with a Ph.D. degree in Computer Science from the University of California, Santa Barbara in 2004 and with a Dipl.Ing. degree from the School of Electrical Engineering, University of Belgrade, Serbia in 1999.

Discussion Panel

Andrew Mutz, Director of Software Engineering, AppFolio

Dr. Andrew Mutz is the Director of Software Engineering at AppFolio, Inc. AppFolio is the fastest growing provider of online residential property management software. Prior to joining AppFolio, he completed his Ph.D. in Computer Science from the University of California, Santa Barbara. His dissertation, "Eliciting Honest Behavior on Computational Grids," focused on the problem of maximizing the amount of useful work accomplished on a computational cluster by incentivizing accurate revelation from users of the value of the work being performed. Prior to graduate school, he worked as a Software Engineer at ExpertCity, inc. (now Citrix Online). He received his B.S. in Computer Science in the College of Creative Studies at UCSB.

Chad Sweet, Software Engineering Manager, Qualcomm

Chad Sweet is a Principal Software Engineering Manager at Qualcomm in the Corporate R&D division. He has led their augmented reality software research and development for the past three and a half years. He has more than a dozen patents pending in the wireless communications and augmented reality fields. Chad has a Bachelor's in Computer Engineering from Vanderbilt University and has been with Qualcomm for more than 13 years.

Saty Bahadur, Principle Developer Manager, Windows Phone Team, Microsoft

Saty is the Principle Developer Manager for the Windows Phone Team. As a member of the Windows Phone CX Engineering group, he works on developing software that enables more mobile operators, geographies, and variants, and adds end-user value. Previously, he was a Principal Development Manager in the Windows Devices and Networking Team, and, prior to that, he worked at Intel Corporation in various leadership roles for 12 years. Saty received his M.S. in Computer Science from Clemson University and his B.E. in (Hons) Instrumentation from BITS, Pilani, India.

Table of Contents

Security Session (led by Gianluca Stringhini)

- Fear the EAR: Automated Discovery of Execution after Redirection Vulnerabilities 1
  Adam Doupé, Bryce Boe, Christopher Kruegel, Giovanni Vigna
- BareBox: Efficient Malware Analysis on Bare-Metal 3
  Dhilung Kirat, Giovanni Vigna, Christopher Kruegel
- New Hardware Description Language for Secure Information Flow 5
  Xun Li, Mohit Tiwari, Jason Oberg, Vineeth Kashyap, Ben Hardekopf, Timothy Sherwood, Frederic Chong

Networking and Architecture Session (led by Maria Zheleva)

- The Implications of MIMO and Channel Bonding on Channel Management in 802.11n 7
  Lara Deek, Elizabeth Belding, Kevin Almeroth
- VillageCell: Cellular Connectivity in Rural Areas 9
  Abhinav Anand, Veljko Pejovic, Elizabeth Belding
- Barely Alive Memory Servers: Keeping Data Active in a Low Power State 11
  Vlasia Anagnostopoulou, Ricardo Bianchini, Tao Yang, Frederic T. Chong
- Ranking Security-Important Assets in Corporate Networks 13
  Ali Zand, Christopher Kruegel, Richard Kemmerer, and Giovanni Vigna

Multifarious Session A (led by Steffen Gauglitz)

- Complete Information Pursuit Evasion in Polygonal Environments 15
  Kyle Klein, Subhash Suri
- Temporal Cross-Sell Optimization Using Action Proxy-Driven Reinforcement Learning 17
  Nan Li, Naoki Abe
- A Flexible Open-Source Toolbox for Scalable Complex Graph Analysis 19
  Adam Lugowski, David Alber, Aydin Buluc, John R. Gilbert, Steve Reinhardt, Yun Teng, Andrew Waranis

Multifarious Session B (led by Nichole Stockman)

- Melody Matcher: A Music-Linguistic Approach to Analyzing the Intelligibility of Song Lyrics 21
  Jennifer Jenee G. Hughes

- The Composition Context in Point-and-Shoot Photography 23
  Daniel Vaquero, Matthew Turk
- iSketchVis: Integrating Sketch-based Interaction with Computer Supported Data Analysis 25
  Jeffrey Browne, Bongshin Lee, Sheelagh Carpendale, Timothy Sherwood, Nathalie Riche

Posters

- Reliable Selection of Interest Points for Keypoint Matching 27
  Victor Fragoso, Matthew Turk
- A Botmasters Perspective of Coordinating Large-Scale Spam Campaigns 29
  Brett Stone-Gross, Thorsten Holz, Gianluca Stringhini, Giovanni Vigna
- Back to the Future: Replaying Malicious Web-pages for Regression Testing 31
  Yan Shoshitaishvili, Alexandros Kapravelos, Christopher Kruegel, Giovanni Vigna
- Generating Applications for the Cloud from Formal Specifications 33
  Christopher Coakley, Peter Cappello
- EvilSeed : A Guided Approach to Finding Malicious Web Pages 35
  Luca Invernizzi, Marco Cova, Christopher Kruegel
- The Phantom Anonymity Protocol 37
  Johannes Schlumberger, Magnus Brading, Amir Houmansadr
- A Framework for Modeling Trust in Collaborative Ontologies 39
  Byungkyu Kang, John ODonovan, Tobias Hollerer
- SIGMA: A Statistical Interface for Graph Manipulation and Analysis 41
  Greg Meyer, Brynjar Gretarsson, Svetlin Bostandjiev, John ODonovan, Tobias Hollerer
- Detecting Service Outages via Social Media Analysis 43
  Eriq Augustine, Cailin Cushing, Kim Paterson, Matt Tognetti, Alex Dekhtyar
- Chronology-Sensitive Hierarchical Clustering of Pyrosequenced DNA Samples of E.Coli 45
  Aldrin Montana, Alex Dekhtyar, Emily Neal, Michael Black, Chris Kitts
- Closest Pair and the Post Office Problem for Stochastic Points 47
  Pegah Kamousi, Timothy Chan, Subhash Suri

Fear the EAR: Automated Discovery of Execution After Redirect Vulnerabilities

Adam Doupé, Bryce Boe, Christopher Kruegel, and Giovanni Vigna
University of California, Santa Barbara
{adoupe, bboe, chris, vigna}@cs.ucsb.edu

***Abstract*—The complexity of modern web applications makes it difficult for developers to fully understand the security implications of their code. The resulting security vulnerabilities are exploited by attackers to gain unauthorized access to the web application environment. Previous research into web application vulnerabilities has mostly focused on input validation flaws, such as cross-site scripting and SQL injection, while logic flaws have received comparably less attention.**

In this paper, we present a comprehensive study of a relatively unknown logic flaw in web applications, which we call Execution After Redirect, or EAR. A web application developer can introduce an EAR by calling a redirect method under the assumption that execution will halt. A vulnerability occurs when server-side execution continues after the developer's intended halting point, which can lead to broken/insufficient access controls and possibly information leakage.

We present an open-source, white-box, static analysis tool to detect EARs in Ruby on Rails web applications, which found 3,944 EAR instances in 18,127 open-source applications.

I. Introduction

An increasing number of services are being offered on-line. For example, banking, shopping, socializing, reading the news, and enjoying entertainment are all available on the web. The increasing amount of sensitive data stored by web applications has attracted the attention of cyber-criminals, who break into systems to steal valuable information.

In this paper, we present an in-depth study of a little-known web application logic flaw; one we are calling Execution After Redirect (EAR). An EAR occurs because of a developer's misunderstanding of how the web application framework operates. In the normal workflow of a web application, a user sends a request to the web application. The web application receives this request, performs some server-side processing, and returns an HTTP response. Part of the HTTP response can be a notification that the client (a web browser) should look elsewhere for the requested resource. In this case, the web application sets the HTTP response code to indicate a redirect, and adds a `Location` header. The response code instructs the browser to look for the resource originally requested at a new URL specified by the web application in the HTTP `Location` header. This process is known as redirection; the web application redirects the user to another resource.

An EAR can be introduced when a web application developer writes code that issues an HTTP redirect under the assumption that the redirect will automatically halt execution of the web application. Depending on the framework, execution can continue after the call to the redirect function, potentially violating the security of the web application.

```
1 class TopicsController < ApplicationController
2  def update
3   @topic = Topic.find(params[:id])
4   if not current_user.is_admin?
5     redirect_to("/")
6   end
7   @topic.update_attributes(params[:topic])
8  end
9 end
```

Listing 1: Example of an EAR vulnerability in Ruby on Rails

We present a study of Execution After Redirect vulnerabilities: we provide a brief overview of EARs and develop a novel static analysis algorithm to detect EARs, which we implemented in an open-source tool to analyze Ruby on Rails web applications. We then present the results of running our tool on 18,127 open-source Ruby on Rails applications, which found 3,944 EARs.

II. Overview of EARs

An Execution After Redirect is a logic flaw in web applications due to a developer's misunderstanding of the semantics of redirection. Very often this misunderstanding is caused by the web framework used by the developer. In particular, developers assume that the web application will halt after performing a redirect. Certain web frameworks, however, do not halt execution on a redirect, and instead, execute all the code that follows the redirect operation.

As an example, consider an EAR bug adapted from a real vulnerability shown in Listing 1. The code appears to redirect the current user to "/" if she is not an administrator (Line 5), and if she is an administrator, `@topic` will be updated with the parameters sent by the user (Line 7). The code does not execute in this way, because Ruby on Rails does not halt execution on a redirect. Thus, *any* user, not only the administrator, can modify the topic, violating the intended authorization and compromising the security of the web application.

Execution After Redirect logic flaws can be of two types: benign or vulnerable. A *benign* EAR is one in which no security properties of the application are violated, even though additional, unintended, code is executed after a redirect. A *vulnerable* EAR occurs when the code executed after the

Type of EAR reported	Number reported
Benign	3,089
Vulnerable	855
Total	3,944
Total Projects	18,127
Any EAR	1,173
Only Benign	830
At least one vulnerable EAR	343

TABLE I: Results of running the white-box detector against Ruby on Rails applications, 6.5% of which contained an EAR flaw. 2.9% of the projects had an EAR classified as vulnerable.

redirect violates the security properties of the web application. More specifically, in a vulnerable EAR the code executed after the redirection allows unauthorized modification to the state of the web application (typically the database), and/or causes leakage (reads and returns to the browser) of data to an unauthorized user.

We analyzed nine web frameworks to see how they differed with respect to the built-in redirect functions. Of those frameworks we examined, we found the following to be vulnerable to Execution After Redirect vulnerabilities: Ruby on Rails, Grails, PHP, J2EE, and Struts. Even though our detection is focused on Ruby on Rails, EARs are a widespread problem that can affect many different languages and frameworks.

III. EAR Detection

Our white-box tool uses static source code analysis to identify Execution After Redirect bugs in Ruby on Rails web applications. Here we will give a brief overview of our algorithm.

The goal of our EAR detector is to find a path in the application's Control Flow Graph (CFG) that contains both a call to a redirect method and code following that redirect method. The algorithm operates in five steps: (i) generate a CFG of the application; (ii) find all redirection methods; (iii) prune infeasible paths in the CFG to reduce false positives; (iv) detect EARs by finding a path in the CFG where code is executed after a redirect method is called; (v) use a heuristic to differentiate between benign and vulnerable EARs.

IV. Results

We used our EAR detection tool to find real-world EARs in open-source Ruby on Rails web applications. We identified 18,127 Ruby on Rails applications from GitHub that served as our testbed.

Table I summarizes the results. In total, we found 3,944 EAR instances in 1,173 projects. 855 of these EARs, present in 343 projects, were classified as vulnerable. This means that 6.5% of Rails applications we tested contained at least one EAR, and 29.3% of the applications containing EARs had an EAR classified as vulnerable.

A. Detection Effectiveness

To determine the effectiveness of our tool, we manually inspected all 855 vulnerable EARs. The results are shown in Table II. We verified that 485, or 59.9%, were true positives.

Classification after manual analysis	Number
True Vulnerable EARs	485
Benign EARs	325
No EARs (False Positives)	45

TABLE II: Results of manually inspecting the 855 vulnerable EARs reported by our white-box tool. 40.1% were benign, and 5.3% were not EARs.

Many of these were caused by ad-hoc authorization checks, where the developer issued a redirect when the authorization check failed, assuming that execution would halt. Some examples of security violations were allowing non-administrators access to administrator functionality, allowing modifications to items not belonging to the current user, and being able to sign up for a conference even though it was full.

For vulnerable EARs, we consider two different types of false positives: false *vulnerable* EARs, which are benign EARs mistakenly reported as vulnerable, and false EARs (false positives). As shown in Table II, our tool generated 45 false EARs, for a false positive rate of 5.3%. These were mostly due to impossible paths from the redirect methods to code that the tool was not able to statically determine. Our vulnerable EAR heuristic had a higher false detection rate of 40.1%. Once again the major reason for this high rate was that there was no feasible path from the redirect to the method that changed the state of the database. Even though this rate is high, it is well in line with the precision of previous static analysis tools [1]–[3].

We checked that the false EAR rate did not differ significantly among the benign EARs by manually inspecting 200 random benign EARs. We saw 13 false EARs in the manual inspection, for a false positive rate of 6.5%. We also did not see any vulnerable EARs among the benign EARs, thus, we did not see any false negative vulnerable EARs in our experiments.

From our results, we can conclude that we detect EARs well, however, it is difficult to distinguish between benign and vulnerable EARs. However, even though certain EARs might not be currently vulnerable, they are still programming errors that should be fixed.

V. Conclusion

We have described a new type of vulnerability, Execution After Redirect, and developed a novel static analysis tool to effectively find EARs. We showed that EARs are difficult to classify as vulnerable. This difficulty is due to vulnerable EARs violating the specific logic of the web application. Better understanding of the application's logic should help differentiate vulnerable and benign EARs and it will be the focus of future work.

References

[1] Huang, Y.-W., Yu, F., Hang, C., Tsai, C.-H., Lee, D.-T., and Kuo, S.-Y. Securing web application code by static analysis and runtime protection. In *Proceedings of the 13th international conference on World Wide Web* (New York, NY, USA, 2004), WWW '04, ACM, pp. 40–52.

[2] Jovanovic, N., Kruegel, C., and Kirda, E. Pixy: A static analysis tool for detecting web application vulnerabilities (short paper). In *IN 2006 IEEE SYMPOSIUM ON SECURITY AND PRIVACY* (2006), pp. 258–263.

[3] Livshits, V. B., and Lam, M. S. Finding security vulnerabilities in java applications with static analysis. In *Proceedings of the 14th conference on USENIX Security Symposium - Volume 14* (Berkeley, CA, USA, 2005), USENIX Association, pp. 18–18.

BareBox: Efficient Malware Analysis on Bare-Metal

Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel
Computer Security Lab
University of California, Santa Barbara
{dhilung,vigna,chris}@cs.ucsb.edu

***Abstract*—Present day malware analysis techniques use both virtualized and emulated environments which can be detected by modern malware to hide their malicious behavior. Such malware need to be analyzed in a bare-metal environment. However, with available bare-metal analysis framework, restoration of the analysis environment into the previous clean-state requires a system reboot, which largely limits the overall throughput. This paper presents a bare-metal analysis framework based on a novel technique of reboot-less system restoration, for executing and monitoring malware run in a native hardware environment. Live system restore is accomplished by restoring the entire physical memory of the analysis operating system from another, custom written operating system, that runs outside of the analysis operating system.**

I. Introduction

Present day malware analysis techniques use both virtualized and emulated environments to analyze malware. However, a class of malware called VM-aware malware are capable of detecting such environments and then hiding its malicious behavior to foil the analysis. More transparent malware analyzers such as hardware-virtualization-based Ether [1] and system-emulation-based Anubis [2] have been proposed. However, it has been demonstrated that in-guest detection of these systems are possible [3], [4].

The definitive way to observe the actual behavior of VM-aware malware is to execute them in a "bare-metal" system, a system running on real hardware. After each analysis, such system must be restored back to the previous clean state. Available state-of-the-art system restore solutions are disk-restore based and require a system reboot. Because of this significant downtime between each analysis, efficient automation of malware analysis in bare-metal systems has been a challenge.

Our primary goal is to develop a bare-metal framework for analyzing malware that does not rely on virtualization or emulation techniques, such that all of the VM detection techniques used by malware are rendered ineffective. To increase the throughput of the entire analysis framework, we also want to restore the environment as fast as possible, to the point where the next analysis can be initiated.

In this paper, we propose a bare-metal analysis framework based on a novel technique that allows reboot-less system restoring. This framework, called BareBox, executes and monitors malware run in a native environment. After the analysis run has been completed, the running OS is restored back to the previously-saved clean state on-the-fly, within few seconds. This restoration process recovers not only the underlying disk state but also the volatile state of the entire operating system (running processes, memory cache, filesystem cache, etc).

II. System Overview

The system consists of a overlay-based volatile mirror disk, a symmetric physical memory partition, and a small custom-written operating system, which we call *Meta-OS*. Live state of the running operating system is restored from this *Meta-OS* by restoring the entire physical memory of the operating system, once a malware analysis run completes. Software-based overlay mechanism selectively redirects sector read and write operations to either the main disk or the mirror disk.

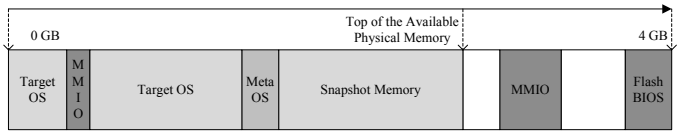


Fig. 1. Physical memory partition

As shown in Figure 1, the physical memory is partitioned in such a way that one of the partitions can be used as a snapshot area. Target OS runs in the other partition, where the actual malware is executed.

Complete restore of a live operating system involves the overwriting of the critical OS structures residing in the physical memory (e.g., the page table, interrupt handlers, etc.) which interfere with the execution of the restoring code itself. It also requires the restoring of the CPU context including system-critical registers (e.g., IDTR, GDTR and other control registers), which are heavily coupled with the content of the physical memory. Because of these circular dependencies, the restoration process can only be performed from an *out-of-OS* execution. *Meta-OS*, that resides outside of the physical memory of the target OS, provides this requirement to create a physical memory snapshot of the target OS and to later restore it. While the CPU is in *protected mode*, an OS context switch to the restored OS is particularly challenging because the CPU needs to switch into an arbitrary *hardware page table*. This is made possible by the careful relocation of the GDT before the context switch and the implementation of segmented-address-space in the *Meta-OS*.

BareBox also needs to restore the device states of the attached devices. However, the exact definition of "device state" is device-specific. Instead, we manipulated the "device

power state", which in turn forces a reset of the internal device state.

Malware monitoring component hooks System Service Descriptor Table (SSDT) to record the system calls generated by the malware. New driver loading and known SSDT hook detection techniques are prevented during the analysis, to protect the integrity of the monitoring component. However, this limits the analysis to *usermode* malware.

III. Evaluation

A. Performance

We compared the performance of BareBox with different system restore options available. As the system restore time of the BareBox was largely dominated by device-state restore, we achieved increased efficiency by implementing selective device restore.

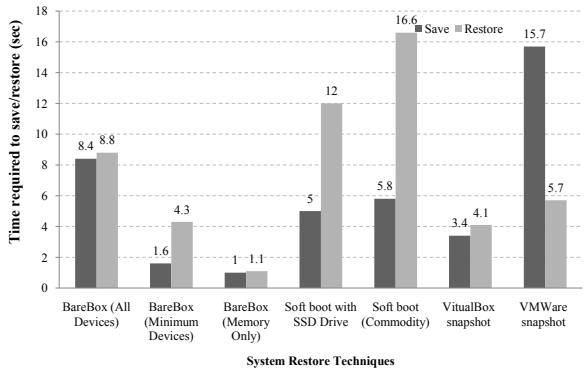


Fig. 2. Evaluation of different system restore techniques.

Looking at the results, we can see that the performance of BareBox is comparable to fast, virtual-machine-based solutions.

B. Dynamic Malware Analysis

We collected six malware samples per family for seven different families from the Anubis [2] database that are known to detect virtualization and emulation environments. We executed them inside a virtualized environment (VMware), an emulated environment (QEMU), and in the BareBox and compared their network related system call traces.

The result in Table I clearly shows the increased number of network activities in our BareBox environment. In addition, we also monitored the number of new processes created by the malware samples in each environment. BareBox was able to elicit more process-related activity across the board.

To evaluate the overall efficiency of the system, we allowed each sample to load and execute for 15 seconds and then the system was automatically restored back for the next analysis run. With only about four seconds overhead for the system restore, we could analyze three malware samples per minute.

TABLE I
Interactions with the Network

Malware Family	BareBox	VMware	QEMU
Rebhip	346	10	55
Telock	205	107	34
Confickr	24	0	0
Zlob/Zeus	408	406	176
Sdbot	152	45	30
Agobot	50	48	3
Armadillo-4	172	82	58

IV. Conclusion

In this paper, we presented BareBox, a framework for dynamic malware analysis in a bare-metal environment. To facilitate efficient analysis, we introduced a novel technique for reboot-less system restore. Since the system executes malware on real hardware, it is not vulnerable to any type of VM/emulation-based detection attacks. We evaluated the effectiveness of the system by successfully monitoring the true malicious behavior of VM/emulation-aware malware samples that did not show malicious behavior in emulated or virtualized environments. After each such analysis, our system was able to efficiently restore the bare-metal system so that the next analysis could be initiated.

References

[1] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: malware analysis via hardware virtualization extensions," in *CCS '08*, 2008.

[2] "Anubis: Analyzing unknown binaries." [Online]. Available: http://anubis.iseclab.org/

[3] G. Pék, B. Bencsáth, and L. Buttyán, "nether: in-guest detection of out-of-the-guest malware analyzers," in *EUROSEC '11*, 2011.

[4] R. Paleari, L. Martignoni, G. Fresi Roglia, and D. Bruschi, "A fistful of red-pills: How to automatically generate procedures to detect CPU emulators," in *3^{rd} USENIX (WOOT)*. Montreal, Canada: ACM.

New Hardware Description Language for Secure Information Flow

Xun Li Mohit Tiwari Jason K. Oberg* Vineeth Kashyap
Ben Hardekopf Timothy Sherwood Frederic T. Chong
Department of Computer Science *Department of Computer Science and Engineering
University of California, Santa Barbara University of California, San Diego
{xun,tiwari,vineeth,chong,sherwood,benh}@cs.ucsb.edu jkoberg@cs.ucsd.edu

***Abstract*—Information flow is an important security property that must be incorporated from the ground up, including at hardware design time, to provide a formal basis for a system's root of trust. We incorporate insights and techniques from designing secure programming languages to provide a new perspective on designing secure hardware—i.e., a new hardware description language, Caisson, that combines domain-specific abstractions common to hardware design with insights from type-based techniques used in secure programming languages.**

I. Motivation

High-assurance embedded systems such as those used in banks, aircraft and cryptographic devices all demand strong guarantees on information flow. Policies may target confidentiality, so that secret information never leaks to unclassified outputs, or they may target integrity, so that untrusted data can never affect critical system data. The high cost of a policy violation ensures that these systems are evaluated extensively before being deployed.

Information flow policies are expressed using a lattice of security levels [3] such that higher elements in the lattice correspond to information with more restricted flow (i.e., secret information for confidentiality or untrusted information for integrity). A simple example of a security lattice would be the typical military classification levels: `Unclassified` $\sqsubseteq$ `Secret` `Top Secret`. An important information flow policy based on such lattices is *non-interference* [4], which requires that no information at a given level in the lattice can flow anywhere except to higher elements in the lattice (e.g., `Secret` information can flow to `Top Secret`, but not vice-versa). High-assurance systems require a static guarantee of non-interference and depend on a *hardware-based root of trust* to enforce this policy.

Existing approaches try to enforce information flow security policies at different levels in the computer systems, from application and programming languages to operating system. However even with expensive overhead, none of existing approaches are able to provide full system security guarantees. Numerous attacks exploit hardware structures such as shared data caches [5], instruction caches [2], and branch predictors [1] to leak information about private keys. Complete information flow security must begin with a principled approach to designing hardware that accounts for the intricate interaction among different hardware components, analyzes the hardware design in its entirety, and does so efficiently enough to be useful in practice.

In contrast to existing approaches, we take language-level techniques for secure information flow and apply them to domain-specific abstractions for hardware design (specifically, finite state machines) to create a new Hardware Description Language (HDL). Our goal is to enable the creation of synchronous hardware designs that are statically-verifiable as secure. Additional benefits of new HDLs are that it allows hardware designers to operate at familiar level of abstraction, enables architects to quickly and easily iterate through potential designs without having to wait for synthesis and simulation to test security properties, and the resulting designs do not suffer from the crippling overhead that comes with dynamic tracking in terms of additional chip area and power consumption.

In this abstract, we present our first work towards this line of research, i.e. a hardware description language for designing information flow secure hardware named Caisson [7]. We have designed simple provably secure processors using Caisson and show the overhead is much smaller compared to existing approaches. We also point out the limitations of Caisson and discuss future work.

II. Caisson: Secure State-Machine-Aware HDL

Finite-state representations of hardware control systems are popular in hardware design. Existing tools such as Altera Quartus, Xilinx ISE, Statecharts, and Esterel are widely used to model systems explicitly as state machines. In designing Caisson we wish to capitalize on this trend and allow hardware designers to operate at a familiar level of abstraction, allowing hardware designs to be easily and transparently modeled using Caisson. For this reason, we base the Caisson language on *finite-state machines*.

While there are existing HDLs based on finite state machines, none target security as a first-class concern. Caisson employs two novel features on top of finite state machines, *nested states* and *parameterized states* to enable precise and expressive enforcement of security policies.

1) Nested States: One of the fundamental methods in the construction of secure systems to high levels of assurance is to decompose the system into trusted and untrusted components, arranged in an architecture that constrains the possible causal

effects and flows of information between these components. On the other hand, resource limitations and cost constraints may make it desirable for trusted and untrusted components to share resources. This leads to systems designs and implementations in which the desired constraints on flows of information between trusted and untrusted components need to be enforced in spite of the fact that these components share resources. To express the kind of state machine in which untrusted states execute alternatively with trusted states without any interference, we allow designers to organize states hierarchically so that all the logic responsible for controlling the switch between trusted and untrusted components completely reside in the parents states, while untrusted computation stays as child states, without being able to affect the rest of the system.

2) Parameterized States: Since Caisson is a statically typed language, every piece of hardware logic has to be tagged with a static security type, and hence in order to compute on data with different security levels, hardware logic also needs to be duplicated. It would be more efficient in terms of chip area to synthesize the *same* logic circuit for different security levels and reuse that circuit by securely multiplexing the different data onto that circuit. This observation motivates the second Caisson language feature, *parameterized states*. The benefit of parameterized states is that Caisson can synthesize a single logic circuit that can safely be used at multiple security levels. In other words, the data being operated on (the Caisson registers) must have distinct types, but the logic operating on the data (the Caisson states) can be parameterized over the types of the data, making the synthesized circuit much more efficient.

III. Information Flow Secure Processor

To demonstrate the utility of these features and of Caisson in general, we design an information-flow secure processor in Caisson. Designing secure hardware controllers and specifying a statically-verifiable secure general-purpose processor is an open problem. Such processors have an important application in high-assurance embedded systems such as those found in aircraft and automobiles.

Our processor implements a standard ISA (derived from the commercial Altera Nios processor) . The ISA is implemented using a combination of hardware data- and control-flow controlled by the execution pipeline with four-stage pipeline: Fetch, Decode, Execute, and Commit. Additional microarchitectural features such as caches, prefetchers, and branch predictors can be attached to the pipeline to improve processor performance. We implemented a cache to illustrate how microarchitectural features can be designed for security; other features can be implemented using similar strategies.

To ensure that `Untrusted` data can never affect `Trusted` computation, our processor *time multiplex* the `Trusted` and `Untrusted` computation on the same physical hardware. The key to secure hardware design is to guarantee that any state changes due to `Untrusted` computation never affect any `Trusted` computation even when the computations share the same pipeline stages, cache, and other processor features. Caisson's language abstractions and type system collaborate to provide the hardware designer with the tools needed to easily encode and verify a secure design.

	Base	GLIFT		Caisson	
Synthesis Time (min)	1:50	153:56	**83.96X**	4:04	**2.22X**
Area (μm^2)	38462.86	128506.89	**3.34X**	52088.78	**1.35X**
CPU delay (ns)	2.96	7.79	**2.63X**	4.32	**1.46X**
Power (μW)	614.148	1730	**2.82X**	666.912	**1.09X**

Fig. 1. Comparison among a non-secured commercial processor (Base), GLIFT processor, and a secure processor designed using Caisson.

To quantify the hardware design overhead introduced by our approach we compare our processor design (**Caisson**) with a non-secured, simplified version of the commercial Nios Processor (**Base**) and the same Nios processor augmented to dynamically track information flow using GLIFT (**GLIFT**) [6] (the only secure processor design in the literature). Synthesis results of three designs is given in Figure 1. Results show that designing a secure processor using Caisson not only provides a strong static guarantee about information flow security, but also (1) allows a more straightforward and natural way of designing a secure processor, and (2) introduces much less area (1.35X in Caisson vs. 3.34X in GLIFT), timing and power overhead (1.09X in Caisson vs. 2.82X in GLIFT) than dynamic tracking techniques such as GLIFT.

IV. Conclusion and Future Work

In this work, we combine insights from traditional type-based secure languages with domain-specfic design patterns used for hardware design and present a new hardware description language, Caisson, for constructing statically-verifiable secure hardware designs. By formalizing certain security design patterns and providing direct language support for enforcing their correct use, Caisson promotes *thinking* about secure hardware design in new, useful ways that don't naturally arise in existing languages.

However as a statically typed language, Caisson requires all the resources in the design to be statically partitioned based on the security lattice, making the design inflexible. Extra hardware overhead still exist due to the need to multiplexing among different partitions. In our future work, we seek to improve Caisson to deal with more dynamic designs with less hardware overhead, and to design more powerful and practical secure processors using our language.

References

[1] O. Accigmez, J. pierre Seifert, and C. K. Koc. Predicting secret keys via branch prediction. In *The Cryptographers' Track at the RSA Conference*, pages 225–242. Springer-Verlag, 2007.

[2] O. Aciiçmez. Yet another microarchitectural attack: Exploiting i-cache. In *CCS Computer Security Architecture Workshop*, 2007.

[3] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7):504–513, 1977.

[4] J. A. Goguen and J. Meseguer. Security policies and security models. In *IEEE Symposium on Security and Privacy*, 1982.

[5] C. Percival. Cache missing for fun and profit. In *Proc. of BSDCan*, 2005.

[6] M. Tiwari, H. Wassel, B. Mazloom, S. Mysore, F. Chong, and T. Sherwood. Complete information flow tracking from the gates up. In *ASPLOS*, March 2009.

[7] X. Li, M. Tiwari, J.K. Oberg, V. Kashyap, F.T. Chong, T. Sherwood, B. Hardekopf. Caisson: a hardware description language for secure information flow. In PLDI 2011.

The Implications of MIMO and Channel Bonding on Channel Management in 802.11n

Lara Deek, Elizabeth Belding, Kevin Almeroth
{laradeek, ebelding, almeroth}@cs.ucsb.edu

I. Introduction and Background

Wireless devices such as laptops, smartphones, and smartpads have gained rapid popularity in our modern age, and, for users of such devices, it is not uncommon to come across the term *WiFi*. Whether we are stopping at a hotel or having coffee at our local coffee shop, we identify our ability to connect to the Internet with the availability of a *WiFi connection*. WiFi is a wireless standard that enables our devices to connect to the Internet when within range of a wireless network, or WLAN (Wireless Local Area Network), that is connected to the Internet. The source of Internet connection in a WLAN is referred to as a *WiFi hotspot*, or *Access Point* (*AP*), which users must associate with in order to connect to the Internet. Now that a connection is established, the method of exchanging data in a WLAN is controlled by a set of standards referred to as IEEE 802.11. In fact, WiFi was originally created as a simpler term for the IEEE 802.11 standard.

Over the years, the 802.11 wireless standard has been amended, and now consists of a family of standards, the more commonly used ones being 802.11 a/b/g referred to as *legacy clients*. Recently, a new standard has emerged, IEEE *802.11n*, which improves considerably on previous legacy clients. The performance benefits provided by 802.11n as well as the challenges to exploit them are the focus of our work.

IEEE 802.11n is a next generation WLAN technology that adds major upgrades to legacy 802.11 a/b/g clients. Traditionally, legacy clients have transmitted data on fixed 20MHz-width channels. With the emergence of 802.11n, clients can now operate on wider channels that achieve higher transmission rates, and, more specifically, on 40MHz-width channels. Operating on 40MHz channels is referred to as *channel bonding*. Furthermore, where legacy 802.11 clients followed a SISO (Single-Input Single-Output) antenna technology, 802.11n incorporated a *MIMO* (Multiple-Input Multiple-Output) smart-antenna technology. In comparison to SISO systems which transmit data over one antenna, MIMO transmits data over multiple antennas simultaneously and takes advantage of this multiplicity of data to improve either the signal-to-noise ratio (SNR) at a particular location or the data rate, using *spatial diversity* and *spatial multiplexing* techniques, respectively [2], [3]. Using *spatial diversity*, one data stream is transmitted over multiple antennas to achieve data redundancy and thus improve SNR at the receiver. Using *spatial multiplexing*, different data streams are transmitted over each antenna thereby increasing the amount of data per transmission, referred to as data rate. We show that, in comparison to legacy clients, MIMO and channel bonding in 802.11n exhibit behavior and add complexity to the network that needs to be investigated to efficiently manage bandwidth in WLANs.

In this work, we present a subset of our investigations. We show that standard metrics, such as signal strength (RSSI) between a transmitter and receiver pair, do not accurately reflect performance. Due to the MIMO feature, the impact of environment conditions and the presence of a rich scattering environment play a major role in characterizing performance. This knowledge allows for more accurate decision making when assigning bandwidth to nodes to maximize performance.

II. Evaluation Environment

We conduct our experiments using a stationary testbed deployed in a semi-open office environment. The nodes are laptops running the Ubuntu 10.04 LTS distribution with Linux kernel version 2.6.32. All the laptops are equipped with an 802.11n AirMagnet 2x3 MIMO PC card. The PC card allows modulation and coding scheme (MCS) indices 0 to 15. We categorize MCS indices into two groups based on their corresponding MIMO mode and refer to these groups as *sets*: a set for MCS 0 to 7, which exploits *spatial diversity*, and a set for MCS 8 to 15, which achieves *spatial multiplexing*.

We vary the locations of transmitter and receiver pairs in our testbed in order to obtain a rich set of link conditions and qualities. In our experiments, we generate constant bit-rate UDP traffic between transmitter and receiver pairs using the *iperf* tool. We monitor the performance of UDP flows and evaluate performance in terms of MAC layer throughput. We conduct our experiments exclusively on the 5GHz frequency range in an interference-free environment. We control the transmission MCS as well as the channel width. We run our experiments for all supported MCS values from 0 to 15 to identify the best MCS for each tested link and channel width configuration. We use the term *best throughput* to reflect the application layer throughput yielded by the MCS that achieves the highest throughput for the link under study.

III. Results

We now take a close look at the effect of network parameters on the performance between single transmitter and receiver pairs.

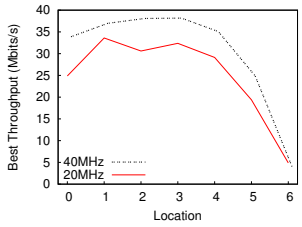
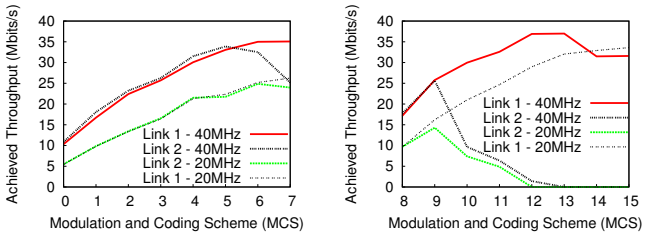
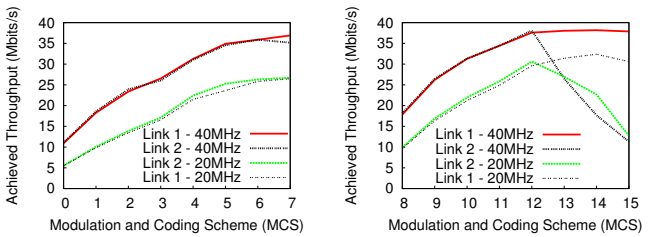


Fig. 1. Throughput achieved between single transmitter and receiver pairs at varying locations. The locations are sorted in order of decreasing RSSI, from highest to lowest.



(a) Good signal quality



(b) Moderate signal quality

Fig. 2. Throughput achieved between transmitter and receiver pairs with similar signal qualities. For similar RSSI values, performance is impacted by the richness of the scattering environment, particularly when using high modulation schemes that use *spatial multiplexing* to increase transmission data rates.

A. What knowledge can we extract from RSSI in 802.11n environments?

Figure 1 plots the highest achieved throughput between single transmitter/receiver pairs at varying locations, sorted in decreasing order of received signal strength indicator (RSSI) of each node pair. Location 0 represents the station receiving the strongest signal, and location 6 is where the lowest RSSI is measured. We would expect that the highest RSSI would allow the most accurate decoding of the transmitted signal, and the best performance. Therefore, we expect throughput to monotonically decrease as RSSI decreases. However, Figure 1 shows that this is, in fact, not the case. For example, regardless of the channel width, locations 1 to 4 outperform location 0, even though the latter is the station receiving the strongest signal. As a result, we can affirm that RSSI alone is not a reliable link quality metric due to MIMO. In Section III-B, we discuss how MIMO transmissions can take advantage of different propagation phenomena. These phenomena depend on the characteristics of the path between a transmitter and a receiver, and hence can be highly unpredictable.

B. What is the impact of the environment on performance?

As shown in Section III-A, the behavior of transmissions in 802.11n environments cannot be explained using RSSI information alone. In fact, for links with similar signal quality, performance values differ considerably. We now analyze whether rich scattering contributes to this behavior.

With the incorporation of MIMO in 802.11n networks, the traditionally negative impact of multi-path diversity now contributes to performance, where MIMO overcomes fading effects and instead uses multi-path diversity to improve signal quality. We evaluate the impact of MIMO by comparing the throughput achieved between links with similar signal quality. In Figure 2(a), we compare two links with good signal quality (-30dBm), where the client for Link 2 is in direct line-of-sight of the transmitter while the client of Link 1 is separated by obstacles. In Figure 2(b), we compare two links with moderate signal quality (between -43 and -46dBm), where the receivers are placed at different locations and their clients are separated by different obstacles. For the *spatial diversity set* (MCS 0-7), we observe little difference between links of similar strength. That is to say, with spatial diversity, RSSI still provides reliable information about the potential performance of the link. However, for the *spatial multiplexing set* (MCS 8-15), we observe considerable differences in throughput. For example, in Figure 2(a), Link 1 and Link 2 achieve similar throughput values for low MCS indices, but for MCS greater than 8, Link 2's performance drops while Link 1 maintains or improves its performance. In order for the signals to be correctly separated and decoded using *spatial multiplexing*, they should arrive at the receiver across independent spatial paths with sufficiently different spatial signatures. Therefore, we can attribute the performance differences in Figure 2 to the extent to which an environment is rich in scattering. The impact of poor scattering is observed more accurately for strong links where the transmitter and receiver are likely to be in close range with each other, as seen for Link 2 in Figure 2(a), where both nodes are in line-of-sight. In such cases, performance varies considerably due to the potential scarcity of independent spatial paths between transmitter and receiver pairs.

IV. Conclusion

The addition of MIMO and channel bonding in the 802.11n standard creates new opportunities to exploit bandwidth in WLANs. However, as we have shown in this work, these new features create new complexities in the network that need to be evaluated to efficiently manage scarce bandwidth.

References

[1] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A case for adapting channel width in wireless networks," *ACM SIGCOMM Computer Communications Review*, vol. 38, no. 4, pp. 135–146, October 2008.

[2] K. Pelechrinis, I. Broustis, T. Salonidis, S. V. Krishnamurthy, and P. Mohapatra, "Design and deployment considerations for high performance MIMO testbeds," in *WICON*, 2008.

[3] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "802.11 with multiple antennas for dummies," *ACM SIGCOMM Computer Communications Review*, vol. 40, pp. 19–25, January 2010.

VillageCell: Cellular Connectivity in Rural Areas

Abhinav Anand
University of California, Santa Barbara
abhinav_anand@umail.ucsb.edu

Veljko Pejovic
University of California, Santa Barbara
veljko@cs.ucsb.edu

Elizabeth M. Belding
University of California, Santa Barbara
ebelding@cs.ucsb.edu

***Abstract*—Mobile telephony brings economic and social benefits to its users. As handsets have become more affordable, the ownership has reached staggering numbers, even in the most remote areas of the world. However, network coverage is often lacking in low population densities and low income rural areas of the developing world, where big telecoms often defer from deploying expensive infrastructure. To solve this coverage gap, we propose VillageCell, a low-cost alternative to high-end cell phone networks. VillageCell relies on software defined radios and open-source solutions to provide free local and cheap long-distance communication for remote regions. Our architecture is simple and easy to deploy, yet robust and requires no modification to GSM handsets. The performance evaluation, done by measuring the call quality metrics and the system capacity under a realistic rural-area network load, shows that VillageCell is indeed an attractive solution for rural area voice connectivity.**

I. Introduction

Voice communication is extremely important in rural areas of the developing world. The lack of transportation infrastructure, high illiteracy levels, and frequent seasonal migrations are some of the characteristics of rural areas that emphasise the need for real-time voice communication. In addition, even more than in the developed world, voice communication in the developing world is a strong enabler of political freedoms, economic growth and efficient health care [5], [2].

In our previous work [4] we investigated how rural Africans indigenize voice communication tools and use voice-over-IP (VoIP) applications. Our findings show that, despite having global connectivity, rural dwellers prefer VoIP applications for local, intra-village, communication. While VoIP communication experiences few problems in the developed world where high quality connectivity is available, due to numerous technical obstacles rural area wireless networks cannot successfully carry VoIP calls even within a single village.

Cellphones are another option for voice communication. They are robust low power devices with a very simple and intuitive user interface, which makes them highly suitable for rural areas in the developing world where energy and infrastructure shortages, as well as illiteracy, are usual problems. Indeed, cellphone penetration has skyrocketed in the developing world [1]. Large telecom operators, however, remain reluctant to deploy cellular infrastructure in remote areas. Deployment of cellular networks is complex and requires installation of Base Tranceiver Stations (BTS) and the supporting infrastructure. The installation cost is high, and it remains difficult for the operators to establish a profitable network in areas with low income and population density.

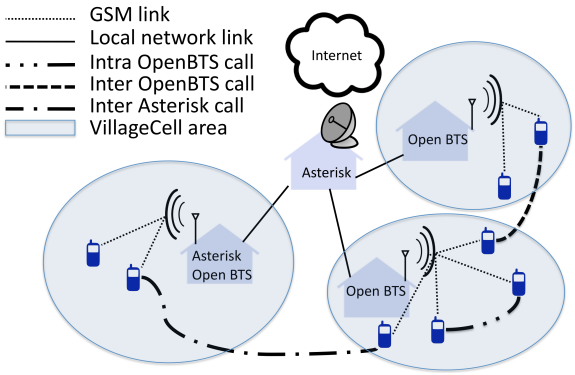


Fig. 1. **Village network architecture.** OpenBTS provides small-scale coverage, while Asterisk nodes enable call management and routing. The expected number of users, their spatial layout, as well as the local network traffic load, impact the OpenBTS and Asterisk interconnection and placement.

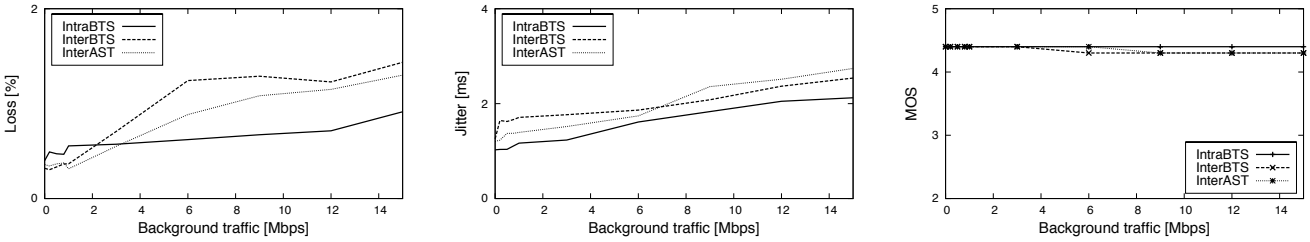
II. VillageCell

In this paper, we propose a cost effective architecture, dubbed VillageCell, for a GSM cellular network in conjunction with a local rural-area network for VoIP services. The solution uses a Software Defined Radio (SDR) controlled by a software implementation of the GSM protocol stack, called OpenBTS[1], which abstracts the BTS and network components into a single entity. OpenBTS uses SDR for transmitting and receiving in the GSM bands and serves as a local cellphone base station. To extend coverage, through a local wireless network, we interconnect multiple BTSs. One or more Private Branch Exchange (PBX) servers implemented in *Asterisk*[2] are also in the network and are used call management and routing.

Integrating GSM with VoIP telephony in this manner is cost effective: OpenBTS provides cellular services for a fraction of the cost of a commercial base station, while a local wireless network brings free VoIP-based communication to cellphone users. In summary, VillageCell delivers free local and cheap long distance cell phone communication; it supports short messaging service (SMS), does not require any modification on the end-user devices and works with existing SIM cards.

Implementation: We implement an instance of VillageCell in a lab setting using USRP2[3], a commercial SDR platform that natively supports OpenBTS software and commodity PCs running Linux and the Asterisk software. Connection among the components is established through Linksys WiFi routers.

[1] http://openbts.sourceforge.net
[2] www.asterisk.org
[3] www.ettus.com



(a) Packet error rate. (b) Delay jitter. (c) MOS.

Fig. 2. **Performance metrics of VillageCell calls with varying level of background traffic and in different setups.**

	IntraBTS	InterBTS	InterAST
iperf-generated TCP	1.00%	1.32%	1.81%
Trace from Macha, Zambia	0.69%	0.82%	0.88%

TABLE I
PACKET LOSS WITH REALISTIC BACKGROUND TRAFFIC.

III. EXPERIMENTAL EVALUATION

Three different scenarios of a VillageCell phone call exist depending on the relationship between the call origin/destination and the architecture layout (figure 1). In each of the scenarios we measure packet loss, delay jitter and the number of supported calls. We experiment with different amounts of artificial background traffic as well as with a real-world network trace from Macha, Zambia.

VillageCell call quality: In figures 2(a), 2(b), and 2(c) we show end-to-end VoIP packet loss, delay jitter and mean opinion score[4] in all scenarios. Both VoIP loss and delay jitter grow linearly with the background traffic. The default GSM voice encoding is G.711, a codec with high loss tolerance, and as long as the packet loss stays below 10% speech communication is possible. In our case loss remains under 2% even with very high background load. To cope with high jitter, VoIP applications often implement receiver-side buffers that store packets for some time (usually less than 100ms) and then sent them out to the decoder in regular intervals. In our setup, the maximum jitter is always below 3ms, thus even a short amount of buffering suffices. Finally, MOS values for each of the scenarios with increasing background traffic remain above 4, implying very good call quality.

Next we investigate VillageCell performance when the voice traffic is mixed with a traffic trace gathered from a wireless network in Macha, Zambia. We replay a randomly selected, ten minute long, snippet of traffic from Macha. We use that traffic in the same way we used the UDP background traffic earlier, and measure the packet loss that a single call experiences in each of the three configurations. To put the results in a perspective, we also use iperf-generated TCP traffic as the background traffic and repeat the experiments. Table I shows that VillageCell looses only a small fraction of packets, thus maintains a good call quality.

VillageCell capacity: We evaluate the VillageCell capacity when it comes to multiple simultaneous calls. In our VillageCell implementation we establish a call and incrementally add more calls. Once we have all the calls going, we measure the packet loss rate in each of the calls and calculate the average value. We show the results (figure 3) with both no background UDP traffic and with 1Mbps constant UDP traffic. In all four cases calls experience less than 0.3% increase in the packet error rate as we activate all six calls.

[4]Voice call quality is often expressed in mean opinion score (MOS) and ranges from perfect (5) to impossible to communicate (1). Any score higher than 3 is considered acceptable.

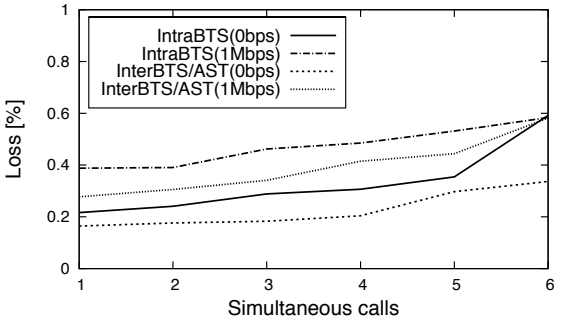


Fig. 3. **VillageCell supported calls scalability.**

IV. CONCLUSION

In this paper we presented VillageCell, a low-cost localized cell phone system for rural areas that solves an important problem of providing localized voice connectivity. In addition, through VillageCell SMS capability, or data-over-voice solutions such as [3], our system also enables free local data service. In future, we plan to develop applications specifically suited for VillageCell's unique affordances. Finally, the existence of a community WiFi network and our familiarity with the Internet usage and needs of local population [4], present a solid foundation for our planned work on deploying a full-scale VillageCell deployment in Macha, Zambia.

REFERENCES

[1] *ITU World Telecommunication/ICT Indicators Database*, December 2010.
[2] R. Abraham. Mobile Phones and Economic Development: Evidence from the Fishing Industry in India. In *ICTD'06*, Berkeley, CA, May 2006.
[3] A. Dhananjay, A. Sharma, M. Paik, J. Chen, J. Li, and L. Subramanian. Data Transmission over Unknown Voice Channels. In *MobiCom'10*, Chicago, IL, September 2010.
[4] D. Johnson, V. Pejovic, E. Belding, and G. van Stam. Traffic Characterization and Internet Usage in Rural Africa. In *WWW'11*, Hyderabad, India, March 2011.
[5] H. V. Milner. The Digital Divide: The Role of Political Institutions in Technology Diffusion. *Comparative Political Studies*, 39:176 – 199, March 2006.

Barely Alive Memory Servers: Keeping Data Active in a Low-Power State

Vlasia Anagnostopoulou*, Ricardo Bianchini†, Tao Yang*, Frederic T. Chong*
*Department of Computer Science, University of California, Santa Barbara
†Department of Computer Science, Rutgers University
{vlasia, tyang, chong}@cs.ucsb.edu, ricardob@cs.rutgers.edu

***Abstract*—Current resource provisioning schemes in Internet services leave servers less than 50% utilized almost all the time. At this level of utilization, the servers' energy efficiency is substantially lower than at peak utilization. A solution to this problem could be dynamically consolidating workloads into fewer servers and turning others off. However, services typically resist doing so, because of high response times during re-activation in handling traffic spikes. Moreover, services often want the memory and/or storage of all servers to be readily available at all times. In this paper, we propose a new *barely-alive* low-power server state that facilitates both fast re-activation and access to memory while in a low-power state. We compare this state to on-off server consolidation, as well as state-of-the-art approaches such as PowerNap and Somniloquy. Specifically, we evaluate provisioning for unexpected load bursts in each energy-saving scheme and find that the barely-alive state results in reduced power consumption compared to the other schemes, while preserving the memory data during consolidation.**

I. Introduction

Energy represents a large fraction of the operational cost of Internet services. As a result, previous works have proposed approaches for conserving energy in these services, such as consolidating workloads into a subset of servers and turning others off, and leveraging dynamic voltage and frequency scaling of the CPUs [4]. Consolidation is particularly attractive for two reasons. First, current resource provisioning schemes leave server utilizations under 50% almost all the time. At these utilizations, server energy efficiency is very low Second, current servers consume a significant amount of energy even when they are completely idle [2]. Despite its benefits, services typically do not use this technique. A major reason is the fear of high response times during re-activation in handling traffic spikes.

In this paper, we propose an approach that does not completely shutdown idle servers, enables fast state transitions, and keeps in-memory application code/data untouched. Specifically, we propose to send servers to a new "barely-alive" power state, instead of turning them completely off after consolidation. *In a barely-alive state, a server's memory can still be accessed, even if many of its other components are turned off. Keeping data active and accessible in a barely-alive state enables software to implement cluster-wide (or "cooperative") main-memory caching, data replication and coherence, or even cluster-wide in-memory data structures, while conserving a significant amount of energy.* We compare the barely-alive state to conventional consolidation via complete server shutdown, as well as more recent hybrid proposals such as PowerNap and Somniloquy. In particular, we evaluate the effect of server restart latency on response time during unexpected load spikes. Unexpected spikes may occur due to a variety of reasons, including external events (e.g., Slashdot effect, attacks), the temporary unavailability of a mirror datacenter, operator mistakes, or software bugs. Under latency constraints, greater restart latency translates to a larger number of extra active servers provisioned to absorb unexpected load. We evaluate the sensitivity of each energy conserving scheme to the duration and magnitude of load spikes, as well as to modifications to data while in energy-conserving server states.

II. Background and Related Work

Many papers have studied dynamic workload consolidation and server turn off [4], [5]. The idea is to adjust the number of active servers dynamically, based on the load offered to the service. During periods of less-than-peak load, the workload can be concentrated (either through state migration or request distribution) on a subset of the servers and others can be turned off.

Somniloquy [1] and PowerNap [3] are two recent works, where low-power server states have been proposed. Somniloquy augments the network interface to be able to turn most other components off during periods of idleness, while retaining network connectivity. In the low-power state, *main memory becomes inaccessible*, so accesses can only be performed to the small memory of the network interface. Moreover, *updates to main memory can only be performed after activation*, thereby increasing delay. In contrast, our state allows read and write accesses to the entire main memory. PowerNap rapidly transitions servers between active and "nap" state, obviating the need for consolidation. *In nap state, a server is not operational.* PowerNap requires server software to avoid unwanted transitions to active state (e.g., due to clock interrupts). More challengingly, PowerNap requires the server to be *completely idle*, which is becoming harder as the number of cores per CPU increases (the idle times of all cores must perfectly overlap). We compare the barely-alive state against these alternative power-states extensively in Section III.

III. Quantitative Evaluation of Barely-Alive

A. Benefits of Fast Activation

A significant challenge for all load consolidation schemes is handling unexpected load spikes without violating latency

constraints in service-level agreements. In this section, we present a simple analysis of barely-alive and previous schemes when faced with a parameterized unexpected load spike. We use this analysis to estimate extra server provisioning and illustrate the tradeoffs of activation latency, standby power consumption, and data update latency.

In Figure 1(left), we present an example of a synthetic load spike. We can parameterize this load spike by duration and amplitude, and choose parameters consistent with observed behavior such as from studies of an HP customer's Web server trace. BA-2 stands for our barely-alive state.

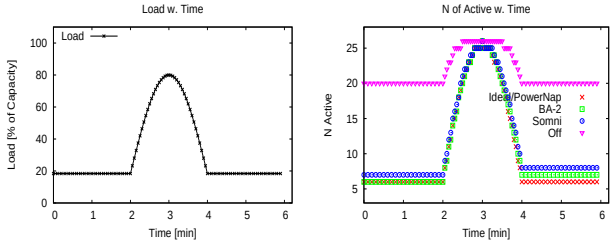


Fig. 1: Load spike (left) and server provisioning (right).

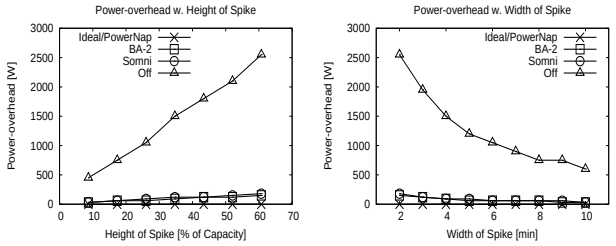


Fig. 2: Impact of spike height (left) and width (right).

To avoid excessive latency, extra active servers must be provisioned to absorb unexpected load until more servers can be activated. The number of extra active servers must match the typical increase in load during the activation time. Thus, the higher the latency of server activation, the more extra servers must be provisioned. This can be seen Figure 1(right), where the number of active servers before the load spike is significantly higher for the On/Off system than for the more sophisticated BA2, PowerNap, and Somniloquy systems. For a baseline comparison, the *Ideal* system is an On/Off system in which servers can be brought up with zero latency and no energy overhead. In general, BA2, PowerNap and Somniloquy are equivalent with respect to load spike provisioning, as long as no data needs to be modified at servers in a low-power state.

Figure 2 shows how the power overhead of extra server provisioning (with respect to the ideal system) varies with spike magnitude and duration. We can see in Figure 2(left) that the On/Off system entails a modest level of overhead with spikes of low magnitude (10,000 connections per second). However, the overhead grows significantly as the spike increases in magnitude. Figure 2(right) shows that, if the duration of the spike is 2 minutes, the over-provisioning overhead is large. The overhead drops to more modest levels for spikes lasting 10 minutes.

B. Benefits of Allowing Immediate Data Updates

Services modify data that may reside on servers that are off or in a low-power state. A key advantage of barely-alive systems is the ability to directly modify data in main memory while in a low-power state. Modification of such data is somewhat impractical for On/Off systems and PowerNap. For On/Off systems, writes would need to be source buffered and deferred to wake up, which can be problematic if systems are off for long periods of time. PowerNap can avoid this problem by waking up the server to perform data updates. However, for all but the most insignificant of write intensities, PowerNap would spend too long in active state.

Other than barely-alive, Somniloquy offers the best solution to data updates while in a low-power state. Writes can be buffered in the Somniloquy device. However, with the limited size of the Somniloquy memory (64 MB), we assume that writes would need to be buffered in their SD card auxiliary storage. In Figure 3, we compare BA2 and Somniloquy as the number of deferred writes varies. Writes are to objects typical of our Web application (6 KB each). Figure 3(left) quantifies the number of servers provisioned in each system. As the number of buffered writes increases (either due to higher write traffic or longer time in a low-power state), the latency at server activation to read the writes from the SD card becomes significant. This growing latency quickly results in a large number of extra servers provisioned for unexpected spikes. Figure 3(right) shows the same comparison in terms of total power for active servers provisioned and power for servers in the low-power state.

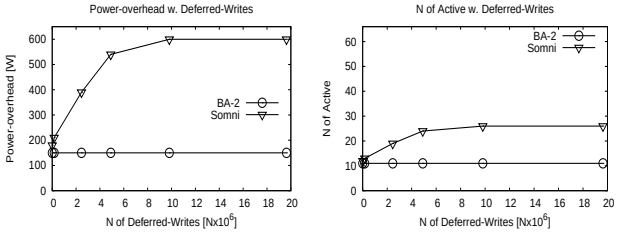


Fig. 3: Impact of deferred writes on active servers (left) and power overhead (right).

IV. Conclusion

In this paper, we introduced a new low-power server state called barely-alive. We compared this to conventional on-off consolidation and other low-power server schemes. We found that the ability to access memory while in a low-power state can save significant energy, and has important advantages for keeping data current.

References

[1] Y. Agarwal et al. Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage. In *NSDI*, 2009.

[2] L. A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12), 2007.

[3] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating Server Idle Power. In *ASPLOS*, 2009.

[4] E. Pinheiro et al. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *COLP*, 2001.

[5] K. Rajamani and C. Lefurgy. On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters. In *ISPASS*, 2003.

Ranking Security-Important Assets in a Corporate Network

Ali Zand*, Christopher Kruegel*, Richard Kemmerer*, and Giovanni Vigna*
*Computer Security Lab
University of California Santa Barbara,
Santa Barbara, California 93106-5110
Email(s): {zand,chris,kemm,vigna}@cs.ucsb.edu

***Abstract*—Today, we are observing an increase in the complexity of computer attacks as a result of the ever-increasing complexity of the network computer systems. To provide security for a complex distributed computer system, it is necessary to know the system assets (services) and missions, their relationship and their importance. In this paper we present a systematic approach for extracting the missions of a corporate network and the assets that contribute to those missions. We use this information to assign an importance score value to each asset that indicates the importance of protection of that asset.**

I. Introduction

Everyday, the networked computer systems are being used for more complex missions, resulting in greater complexity in the system itself. In these circumstances, detecting and defending each attack individually does not seem practical. The protectors of the system need an evolving whole view of the system to be able to make good decisions in a timely manner. They need to have more situation awareness.

Endsley defines situation awareness as "the perception of the elements of the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" [1]. To provide cyber situation awareness, we need to provide the elements of the system and their relationship, and the missions of the system they contribute to. The existing approaches to provide cyber situation awareness consist of vulnerability analysis and attack graphs, intrusion detection and alert correlation, attack trend analysis, causality analysis and forensics, taint and information flow analysis, damage assessment, and intrusion response [2]. Unfortunately, none of the mentioned approaches can provide an acceptable situational awareness independently from the others. Ideally, we need a system using and integrating all the mentioned approaches. To integrate these approaches, one needs to know the elements (assets/services) of the system, the goals (missions) of the system, and the way the elements interact to deliver the goals.

The basis for any cyber situation awareness system is assets (services e.g, NFS service, LDAP service, and IMAP service), users, tasks, and missions(e.g, providing email service to users). Every other piece of information should be directly or indirectly linked to these entities. In order that the system be consistent, we need to have the information about these entities as accurate as possible, because the error in this information will propagate through all other components of the system. Unfortunately the current cyber systems are large and most of the time there is no complete database of missions and assets and their relationship. Furthermore, the administrators of the system often have a hard time keeping track of the changes in the system. Another problem is that the administrators have a demanding job which leaves very little (if any) extra time for them. Therefore, any practical system trying to address this problem should consider administrators as a scarce resource, which cannot be used to solve the problem altogether. Hence, we need an automated asset and mission discovery system which detects the assets, users, tasks, and missions of a computer network system.

The only work, attacking the same problem, that we are aware of, is Camus [3], which tries to automatically extract the relationship between assets and missions and users. Camus assumes that the required data about the missions and assets is available in different formats in different places, and therefore it takes a combination of data mining, inference, and fusion approaches [3].

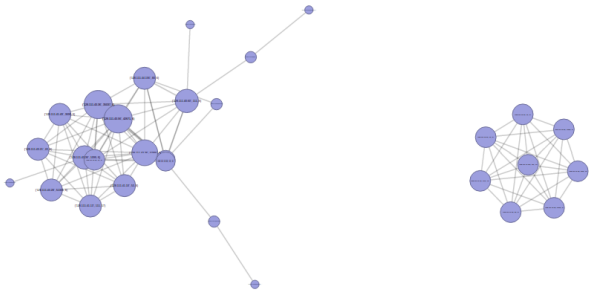
Considering the fact that the network information is maintained in different formats in different places, this approach will need the administrator involvement in the process of data fusion. Another problem is the missions are most of the time documented in a very high-level (human-readable) language which makes it difficult and many times impractical for automated processing.

To address these problems, we use low-level netflow data to extract information about missions and assets. More specifically, we extract the list of assets, tasks and missions that these assets take part in. Finally we use the extracted information about the assets to assign importance scores to each asset. The output of the system can be used by other components of a situation awareness system, or can be directly used by the administrators to prioritize the protection of the assets.

II. Our approach

To provide situation awareness, we need an automated method to extract information about assets, users, tasks and mission from low-level data. We use netflow data gathered, in a period of three months, from network switches of computer science department. A netflow record specifies a TCP or UDP connection start time, end time, source and destination IP address and port number, and the number of packets and bytes transferred.

Fig. 1. Sample correlated graph



A. Extracting Assets and Missions

We first extract all the services[1]. To extract the missions, we build a correlated activity graph for the services. We expect that if two services are part of the same mission, they will have a relatively high correlation. In correlation graph, each service is presented as a node, and two nodes are connected if their corresponding services' activities are correlated. To find the correlated services we need to compute correlations of services with each other.

To compute the correlation of two services, each service is represented as three time series: requests, packets, and bytes time series. The request time serie represents the number of requests that specific service handled in consecutive periods of time. Similarly, packets, and bytes time series represent the same information for number of packets and number of bytes sent by the server respectively. To compute the correlation of two services, three correlation values between their corresponding time series are computed. Two services are considered correlated if any of these three correlation values is greater than a threshold (we used 0.5 as the threshold). We used Pearson correlation, because it is invariant to separate linear scaling of two time-series. We create a dependency graph for the services activity for each time interval t (we used one hour intervals).

To find the missions of the system, we search for maximal cliques[2] in the correlated activity graph, and we consider any maximal clique with more than two nodes a candidate mission. A clique in the correlated activity graph represents a set of services that their load increases and decreases together. For each of these maximal graphs (candidate missions), we count the number of time intervals they were present in the dependency graph. Any candidate mission that appears a considerable number of times, that is more than a threshold (we used 24 in the experiments), is considered a mission.

Figure 1 shows a part of the correlated graph. On the right you see a clique representing a candidate mission. Using this approach, we were able to find data center updates, department e-mails and web, and department roaming profile as different missions being served by the department computer network.

[1]We present a service as a tuple of (IP address, port number, and protocol). We present a mission as a set of services, working together to provide the same goal.

[2]The problem of enumerating all maximal cliques is NP-Hard in general, but it is polynomial in the number of maximal cliques.

B. Ranking Assets

To rank the services, we score them based on the following list of criteria:

- number of bytes/packets sent/received (4 different scores)
- number of handled requests
- number of dependent services
- number of missions it is involved in
- number of failures

Each of these scores are linearly scaled between zero and one, such that the maximum number is mapped to one and the minimum number is mapped to zero. We use a weighted sum of the mentioned scores as the final score of each service. Specifying different weights, a security analyst can put more stress into the criterion she is more interested in. We used the same weight for all criteria. We checked the resulting list of ranked important services with the administrators of CS department and they verified the correctness of the list by checking every important service they recalled in the top of the list.

C. Extracting Meta-Missions

A meta mission is an abstract mission. For example, an abstract email mission using IMAP server, SMTP server, LDAP, and HTTP server is a meta-mission. A particular IMAP server, SMTP server, LDAP server and HTTP server can be used to implement a mission of that type. To extract meta-missions, we look into missions that have the same number of elements with the same type. Two missions are considered of the same type if their constituting elements are from the same port numbers or from the same service clusters. We detected four verified meta-missions, which made us able to find four backup services.

III. Results and Conclusions

We were able to extract 156 network services and 117 missions of the network. These missions were later grouped into seven groups that correspond with six main missions of the department and a mission from a lab. We also ranked the services based on a list of criteria. We asked the administrators to verify our results by comparing it with what they already know. They verified the service ranking, and also the discovered missions. We were also able to discover a mission that the administrators were not aware of. The mission was run by a lab to analyze binary programs. The administrator of the mission verified the discovered components constituting the mission.

References

[1] M. Endsley, "Toward a theory of situation awareness in dynamic systems: Situation awareness," *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[2] P. Barford, M. Dacier, T. G. Dietterich, M. Fredrikson, J. Giffin, S. Jajodia, S. Jha, J. Li, P. Liu, P. Ning, X. Ou, D. Song, L. Strater, V. Swarup, G. Tadda, and C. Wang, *Cyber Situational Awareness*, ser. Advances in Information Security, S. Jajodia, P. Liu, V. Swarup, and C. Wang, Eds. Boston, MA: Springer US, 2010, vol. 46.

[3] J. R. Goodall, A. D'Amico, and J. K. Kopylec, "Camus: Automatically mapping Cyber Assets to Missions and Users," *MILCOM 2009 - 2009 IEEE Military Communications Conference*, pp. 1–7, Oct. 2009.

Complete Information Pursuit Evasion in Polygonal Environments

Kyle Klein and Subhash Suri
Department of Computer Science
University of California Santa Barbara
{kyleklein, suri}@cs.ucsb.edu

***Abstract*—Suppose an unpredictable evader is free to move around in a polygonal environment of arbitrary complexity that is under full camera surveillance. How many pursuers, each with the same maximum speed as the evader, are necessary and sufficient to guarantee a successful capture of the evader? The pursuers always know the evader's current position through the camera network, but need to physically reach the evader to capture it. We allow the evader the knowledge of the current positions of all the pursuers as well—this accords with the standard worst-case analysis model, but also models a practical situation where the evader has "hacked" into the surveillance system. Our main result is to prove that *three* pursuers are always *sufficient and sometimes necessary* to *capture* the evader. The bound is independent of the number of vertices or holes in the polygonal environment.**

I. Introduction

Pursuit-evasion games provide an elegant setting to study algorithmic and strategic questions of exploration or monitoring by autonomous agents. Their mathematical history can be traced back to at least 1930s when Rado posed the now-classical Lion-and-Man problem: a lion and a man in a closed arena have equal maximum speeds; what tactics should the lion employ to be sure of his meal? An important aspect of this problem, and its solution, is the assumption of continuous time: each players motion is a continuous function of time, which allows the lion to get arbitrarily close to the man but never capture him. If, however, the players move in discrete time steps, taking alternating turns but still in continuous space, the outcome is different, as first conjectured by Gale [2] and proved by Sgall [4].

A rich literature on pursuit-evasion problem has emerged since these initial investigations, and the problems tend to fall in two broad categories: discrete space, where the pursuit occurs on a graph, and continuous space, where the pursuit occurs in a geometric space. Our focus is on the latter: visibility-based pursuit in a polygonal environment in two dimensions for which it was shown in a simply-connected n-gon, $O(\log n)$ pursuers are always sufficient, and sometimes necessary [1]. When the polygon has h holes, the number of necessary and sufficient pursuers turns out to be $O(\sqrt{h}+\log n)$ [1]. However, these results hold only for *detection* of the evader, not for the capture.

For capturing the evader, it is reasonable to assume that the pursuers and the evader all have the same maximum speed. Under this assumption, it was shown by Isler et al. [3] that two pursuers can capture the evader in a *simply-connected* polygon using a *randomized* strategy whose expected search time is polynomial in n and the diameter of the polygon. When the polygon has holes, no non-trivial upper bound is known for capturing the evader.

We attempt to *disentangle* these two orthogonal issues inherent in pursuit evasion: *localization*, which is purely an informational problem, and *capture*, which is a problem of planning physical moves. In particular, we ask how complex is the capture problem *if the evader localization is available for free*? Besides being a theoretically interesting question, the problem is also a reasonable model for many practical settings. Given the rapidly dropping cost of electronic surveillance and camera networks, it is now both technologically and economically feasible to have such monitoring capabilities. These technologies enable cheap and ubiquitous detection and localization, but in case of intrusion, a physical capture of the evader is still necessary.

Our main result is that under such a complete information setting, *three pursuers* are always sufficient to capture an equally fast evader in a polygonal environment with holes, using a *deterministic* strategy. The bound is independent of the number of vertices n or the holes of the polygon, although the capture time depends on both n and the diameter of the polygon. Complementing this upper bound, we also show that there exists polygonal environments that require at least three pursuers to capture the evader even with full information.

II. The Problem Formulation

We assume that an evader e is free to move in a two-dimensional closed polygonal environment P, which has n vertices and h holes. A set of pursuers, denoted $p_1, p_2, \ldots$, wish to capture the evader. All the players have the same maximum speed, which we assume is normalized to 1. The bounds in our algorithm depend on the number of vertices n and the diameter of the polygon, $\mathrm{diam}(P)$, which is the maximum distance between any two vertices of P under the shortest path metric.

We model the pursuit-evasion as a continuous space, discrete time game: the players can move anywhere inside the polygon P, but they take turns in making their moves, with the evader moving first. In each move, a player can move to any position whose shortest path distance from its current

This research was supported in part by NSF grant IIS 0904501.

position is at most one; that is, within *geodesic disk* of radius one. We say that e is successfully captured when some pursuer p_i becomes collocated with e.

In order to focus on the complexity of the capture, we assume a complete information setup: each pursuer knows the location of the evader at all times. We also endow the evader the same information, so e also knows the locations of all the pursuers.

III. Strategy for Capture

In the following, we use the notation $d(x, y)$ to denote the shortest path distance between points x and y.

Our overall strategy is to progressively trap the evader in an ever-shrinking region of the polygon P. The pursuit begins by first choosing a path Π_1 that divides the polygon into sub-polygons (see Figure 1(a))—we will use the notation P_e to denote the sub-polygon containing the evader. The pursuers choose a path satisfying the following definition:

Definition 1. ***(Minimal Path:)*** *Suppose Π is a path in P dividing it into two sub-polygons, and P_e is the sub-polygon containing the evader e. We say that Π is* minimal *if*

$$d_\Pi(x, z) \;\leq\; d(x, y) + d(y, z)$$

for all points $x, z \in \Pi$ and $y \in (P_e \setminus \Pi)$.

Intuitively, a minimal path cannot be shortcut: that is, for any two points on the path, it is never shorter to take a detour through an interior point of P_e. We omit the details, however, after an initialization period of $O(\mathrm{diam}(P)^2)$ moves, the pursuer p_1 can successfully guard the path Π_1, meaning that e cannot move across it without being captured.

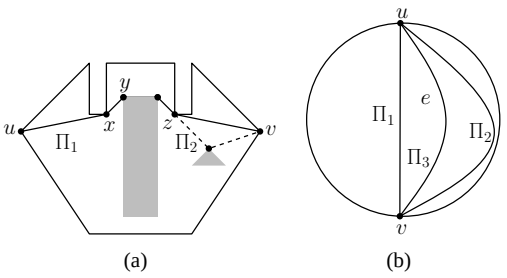


Fig. 1. The left figure shows a polygonal environment, with two holes (a rectangle and a triangle). Π_1 and Π_2 are the first and the second shortest paths between anchors u and v. The right figure illustrates the main strategy of trapping the evader through three paths.

In a general step of the algorithm, assume that the evader lies in a region P_e of the polygon bounded by two minimal paths Π_1 and Π_2 between two anchor vertices u and v. (Strictly speaking, the region P_e is initially bounded by Π_1, which is minimal, and a portion of P's boundary, which is not technically a minimal path. However, the evader cannot cross the polygon boundary, and so we treat this as a special case of the minimal path to avoid duplicating our argument.) We assume that the region P_e contains at least one hole—otherwise, the evader is trapped in a simply-connected region, where a single (the third) pursuer can capture it [3].

The main idea of our proof is to show that, if we compute a *shortest path* from u to v that is distinct from both Π_1 and Π_2, then it divides P_e into *only* two regions, and that the evader is trapped in one of those regions (see Figure 1(b)). We will call this new path the *third* shortest path Π_3. We claim a pursuer can guard Π_3 in $O(n \cdot \mathrm{diam}(P)^2)$ moves, but omit the details. This frees one of the other two pursuers to apply the same strategy to the new smaller P_e, and this can be recursively applied until there are no holes in P_e.

By formally proving the preceeding strategy we can show the following theorem.

Theorem 1. *Three pursuers are always sufficient to capture an evader in $O(n \cdot \mathrm{diam}(P)^2)$ moves in a polygon with n vertices and any number of holes.*

Lastly, by carefully constructing a polygon from a graph known to require three pursuers in the discrete setting, we can show the following theorem.

Theorem 2. *There exists an infinite family of polygons with holes that require at least three pursuers to capture an evader even with complete information about the evader's location.*

IV. Closing Remarks

Traditionally, the papers on continuous space, visibility-based pursuit problem have focussed on simply detecting the evader, and not on capturing it. One of our contributions is to isolate the *intrinsic* complexity of the capture from the associated complexity of detection or localization. In particular, while $\Theta(\sqrt{h} + \log n)$ pursuers are necessary (and also sufficient) for detection or localization of an evader in a n-vertex polygon with h holes [1], our result shows that full localization information allows capture with only *3* pursuers. On the other hand, it still remains an intriguing open problem whether $\Theta(\sqrt{h} + \log n)$ pursuers can *simultaneously* perform localization and capture. We leave that as a topic for future research.

References

[1] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *IJCGA*, 9(5):471–494, 1999.

[2] R. K. Guy. Unsolved problems in combinatorial games. In *Games of No Chance*, pages 475–491, 1991.

[3] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *Robotics, IEEE Transactions on*, 21(5):875 – 884, 2005.

[4] J. Sgall. Solution of david gale's lion and man problem. *Theor. Comput. Sci.*, 259(1-2):663–670, 2001.

Temporal Cross-Sell Optimization Using Action Proxy-Driven Reinforcement Learning

Nan Li
Computer Science Department
University of California, Santa Barbara
Santa Barbara, CA 93106, U.S.A.
nanli@cs.ucsb.edu

Naoki Abe
Business Analytics & Mathematical Sciences Department
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
nabe@us.ibm.com

***Abstract*—Customer lifetime value modeling and temporal cross-sell mining have emerged as two important directions to study customer behaviors. In this paper, we examine both in a unified manner and address the challenge posed by the lack of marketing action data. We propose a variant of reinforcement learning using action proxies to optimally allocate immediate rewards over product categories to maximize their *cross-sell* effects on the *long-term* rewards.**

***Keywords*-Customer Lifetime Value; Cross-Sell; Markov Decision Process; Reinforcement Learning;**

I. Introduction

Customer *lifetime value* (LTV) plays a critical role in marketing management. Recent works in data analytics have developed an approach to maximize LTV based on *reinforcement learning* (RL) and *Markov Decision Process* (MDP) [1], [2]. One shortcoming, however, is its requirement on historical marketing action data to learn the effects of the actions on the rewards. In many business scenarios, historical marketing action data, such as promotion and resource allocation, may very well be missing. We design a novel variant of RL, ***a**ction **p**roxy-driven **r**einforcement **l**earning* (APRL), where certain observable features are used as *action proxies* (APs). An important instantiation is immediate reward targets over a number of product categories ("categorical action proxies"). Since such quantities are tied directly to the rewards, unconstrained formulation would result in an unbounded behavior. We propose to further impose constraints so that the assigned immediate reward targets are bounded. The goal of learning becomes the optimal allocation of immediate reward targets to maximize their effects on the *long-term* rewards. With this concrete formulation, the proposed approach intends to allocate sales/profits target values for multiple product categories given a bounded budget, with the goal of maximizing the long-term profits.

II. Markov Decision Process

Customer LTV modeling can be formulated as maximizing the discounted cumulative reward in the standard MDP terminology. Fig. 1 shows that MDP estimates the LTV along the optimal path (black arrow), while observed policy only leads the customer along the historical path (white arrow). The key components of a standard MDP include: **(1)** the state space, $S = \{s_1, s_2, \ldots, s_n\}$, and an initial state distribution $\phi : S \to \mathbb{R}$; **(2)** the action space $A = \{a_1, a_2, \ldots, a_m\}$, with a transition probability function $\tau : S \times A \times S \to [0, 1]$, such that $\forall s \in S$, $\forall a \in A$, $\Sigma_{s' \in S} \tau(s'|s, a) = 1$, where $\tau(s'|s, a)$ is the probability of transiting to state s' from s via action a; and **(3)** the expected immediate reward function $R : S \times A \to \mathbb{R}$, where $R(s, a)$ is the expected immediate reward of taking action a at state s. Given an MDP, a policy $\pi : S \to A$ determines the action in any state s. Let $V_\pi(s)$ denote the expected long-term cumulative reward at state s if policy π is followed at every step in the future, we have:

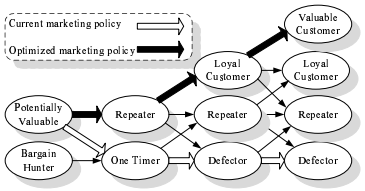


Fig. 1. Customer State Transition Example

$$V_\pi(s) = E_\tau[R(s, \pi(s)) + \gamma V_\pi(\tau(s, \pi(s)))], \quad (1)$$

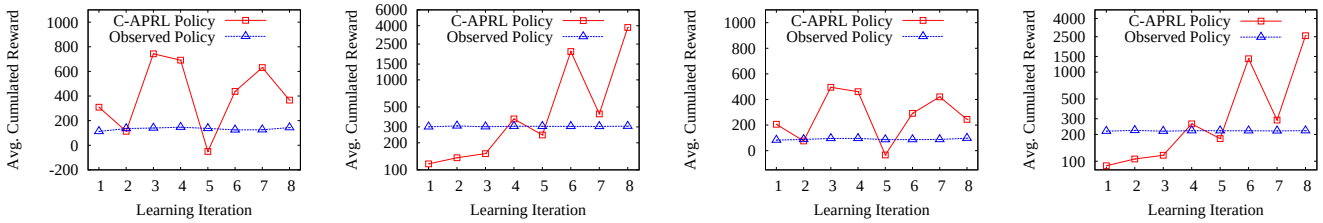
where γ is a discount factor within $[0, 1]$, E is the expectation and $\tau(s, a)$ is a random variable that $\Pr[\tau(s, a) = s'] = \tau(s'|s, a)$. It is proven that for any MDP, there exists an optimal policy π^* that satisfies Bellman's fixed-point equation:

$$V_{\pi^*}(s, a) = \max_{a' \in A} E[R(s, a) + \gamma V_{\pi^*}(\tau(s, a), a')]. \quad (2)$$

Nonetheless, difficulties arise when the exact forms of $\tau(s, a)$ and $R(s, a)$ are unknown. This turns solving an MDP into an RL problem, a variant of which is Q-learning.

$$\begin{aligned} Q_0(s_t, a_t) &= R(s_t, a_t), \\ Q_{k+1}(s_t, a_t) &= (1 - \alpha_k) Q_k(s_t, a_t) \\ &\quad + \alpha_k (R(s_t, a_t) + \gamma \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1})), \\ \pi^*(s_t) &= \text{argmax}_{a_t} Q_\infty(s_t, a_t), \end{aligned} \quad (3)$$

where α_k is the learning ratio at iteration k. Thus, one has an array Q that is updated directly using experiences.



(a) *Saks*, $f = 1$ (b) *MovieLens*, $f = 1$ (c) *Saks*, $f = 2$ (d) *MovieLens*, $f = 2$

Fig. 2. Iterative Lift Evaluation on C-APRL and Observed Policies

III. Constrained-RL Model

A. Action Proxies

In this paper, we propose *action proxy* (AP), which chooses a semi-controllable set of observable features to substitute missing actions. We employ the notion of "categorical action proxies" (CAPs), which are changes of purchase amounts in various product categories.

A policy in the context of CAPs is: $\pi : \boldsymbol{S} \rightarrow \boldsymbol{A} = \{\vec{\delta}\}$, where $\vec{\delta} = (\delta(1), \ldots, \delta(C))^T$ is the purchase amount change vector of length C with C as the number of categories. Specifically, $\pi^*(s_t) = \boldsymbol{a}_t = (\delta_t(1), \ldots, \delta_t(C))^T$, where $\delta_t(c), c \in \{1, \ldots, C\}$ is the purchase amount change for category c from time interval $(t-1, t)$ to $(t, t+1)$. $\boldsymbol{a}_t$ is hereinafter used to denote the action vector at state s_t.

$$\begin{aligned} \boldsymbol{a}_t &= (\delta_t(1), \ldots, \delta_t(C))^T \\ &= (f_t(1), \ldots, f_t(C))^T - (f_{t-1}(1), \ldots, f_{t-1}(C))^T. \end{aligned} \tag{4}$$

B. C-APRL Model

It is critical to constrain the learned policy so that the assigned action proxies are comparable to those observed empirically. The *constrained MDP* (C-MDP) enforces a set of permissible policies, Π, which is determined w.r.t. a set of l constraints on cumulative "costs". We propose a variant of C-MDP with empirical bounds expressed as follows:

$$\begin{aligned} \Pi &= \{\pi \big| \boldsymbol{B}_i^L \leq E_\phi[\mathcal{C}_i^{\boldsymbol{s}_t} \times \boldsymbol{a}_t] \leq \boldsymbol{B}_i^U, i = \{1, \ldots, l\}\} \\ &= \{\pi \big| \boldsymbol{B}_i^L \leq E_\phi[\mathcal{C}_i^{\boldsymbol{s}_t} \times \big(\delta_t(c)\big)^T] \leq \boldsymbol{B}_i^U, i = \{1, \ldots, l\}\} \\ &= \{\pi \big| \wedge_c \boldsymbol{B}_i^L(c) \leq E_\phi[\mathcal{C}_i^{\boldsymbol{s}_t}(c,c)\delta_t(c)] \leq \boldsymbol{B}_i^U(c), i = \{1, \ldots, l\}\} \end{aligned}$$

where $\mathcal{C}_i^{\boldsymbol{s}_t}$ is a $C \times C$ diagonal matrix with $\mathcal{C}_i^{\boldsymbol{s}_t}(c,c)$ as the unit cost incurred by action proxy $c, c \in \{1, \ldots, C\}$ in state $\boldsymbol{s}_t$ for constraint i. $\boldsymbol{B}_i^L$ and $\boldsymbol{B}_i^U$ are both $C \times 1$ vectors, where $\boldsymbol{B}_i^L(c)$ and $\boldsymbol{B}_i^U(c)$ are the lower and upper bounds of constraint i for action proxy c. This leads to a constrained value iteration procedure expressed by:

$$\pi_k^*(\boldsymbol{s}_t) = \underset{\pi \in \Pi}{\operatorname{argmax}} E[R(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma V_{k-1}(\tau(\boldsymbol{s}_t, \boldsymbol{a}_t), \pi(\tau(\boldsymbol{s}_t, \boldsymbol{a}_t)))].$$

We propose ***c**onstrained **a**ction **p**roxy-driven **r**einforcement **l**earning* (C-APRL), an extension of reinforcement learning (RL), to find the optimal policy.

IV. Experiments

Saks Fifth Avenue Data. A random sample of 5,000 customers is used. A sequence of 68 states is generated for each, corresponding to 68 marketing campaigns in 2002, amounting to 340,000 data records. APs are the purchase changes from 9 categories, and the immediate reward is the total purchase in the next state.

MovieLens[1] **Data.** The data contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users. We re-process and segment it into 15 time windows for each user. APs are the ratings of 6 movie categories in the current time window and the immediate reward is the total rating in the next one.

In our experiments, policy advantage is calculated using the estimated average cumulated reward (EAC-reward) over a specified number (f) of time stamps into the future. Fig. 2 plots how the EAC-rewards of the two policies change as the learning iteration progresses. For C-APRL, a typical run starts with a policy that is relatively uninformed, which does not show apparent advantage over the observed policy. In later iterations, C-APRL framework achieves significant policy advantage via iterative re-modeling and re-optimization.

V. Conclusions

We have identified a general problem common to customer LTV modeling, and proposed a solution based on a novel variant of RL with "action proxy". Our future works include: **1)** explore other alternatives of formulating action proxies; **2)** incorporate more types of business constraints.

References

[1] N. Abe et al., Optimizing debt collections using constrained reinforcement learning, *KDD'10*, pages 75-84.

[2] N. Abe, N. K. Verma, C. Apté and R. Schroko, Cross channel optimized marketing by reinforcement learning, *KDD'04*, pages 767-772.

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

A Flexible Open-Source Toolbox for Scalable Complex Graph Analysis[1]

Adam Lugowski
Department of Computer Science
University of California, Santa Barbara
alugowski@cs.ucsb.edu

David Alber
Technical Computing Group
Microsoft Corporation
david.alber@microsoft.com

Aydın Buluç
High Performance Computing Research
Lawrence Berkeley National Laboratory
abuluc@lbl.gov

John R. Gilbert
Department of Computer Science
University of California, Santa Barbara
gilbert@cs.ucsb.edu

Steve Reinhardt
Technical Computing Group
Microsoft Corporation
steve.reinhardt@microsoft.com

Yun Teng, Andrew Waranis
Department of Computer Science
University of California, Santa Barbara
(yunteng, andrewwaranis)@umail.ucsb.edu

***Abstract*—The Knowledge Discovery Toolbox (KDT) enables domain experts to perform complex analyses of huge datasets on supercomputers using a high-level language without grappling with the difficulties of writing parallel code, calling parallel libraries, or becoming a graph expert. KDT delivers competitive performance from a general-purpose, reusable library for graphs on the order of 10 billion edges and greater.**

I. Introduction

Analysis of very large graphs has become indispensable in fields ranging from genomics and biomedicine to financial services, marketing, and national security, among others. Our Knowledge Discovery Toolbox (KDT) is the first package that combines ease of use for domain experts, scalability on supercomputers (large HPC clusters) where many domain scientists run their large scale experiments, and extensibility for graph algorithm developers. KDT addresses the needs both of graph analytics users (who are not expert in algorithms or high-performance computing) and of graph analytics researchers (who are developing algorithms and/or tools for graph analysis). KDT is an open-source (available at `http://kdt.sourceforge.net`), flexible, reusable infrastructure that implements a set of key graph operations with excellent performance on standard computing hardware.

Figure 1 is a snapshot of a sample KDT workflow. First we locate the largest connected component of the graph; then we divide this "giant" component of the graph into clusters of closely-related vertices; we contract the clusters into supervertices; and finally we perform a detailed structural analysis on the graph of supervertices. Figure 2 shows most of the actual KDT Python code that implements this workflow.

II. Examples of use

We describe experiences using the KDT abstractions as graph-analytic researchers, implementing complex algorithms intended as part of KDT itself.

[1]This work was partially supported by NSF grant CNS-0709385, by DOE grant DE-AC02-05CH11231, by a contract from Intel Corporation, by a gift from Microsoft Corporation and by the Center for Scientific Computing at UCSB under NSF Grant CNS-0960316.

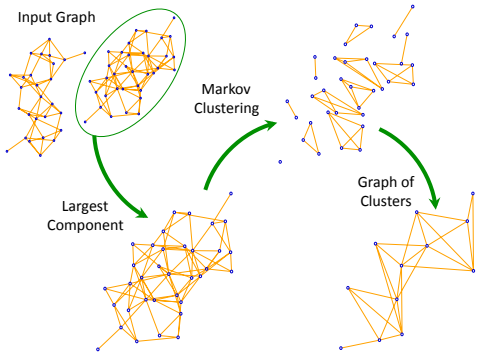


Fig. 1: An example graph analysis mini-workflow in KDT.

```
# the variable bigG contains the input graph
# find and select the giant component
comp = bigG.connComp()
giantComp = comp.hist().argmax()
G = bigG.subgraph(comp==giantComp)
# cluster the graph
clus = G.cluster('Markov')
# contract the clusters
smallG = G.contract(clus)
```

Fig. 2: KDT code for mini-workflow in Figure 1.

Breadth-first search explores all the edges out of the current frontier vertices. This is the same computational pattern as multiplying a sparse matrix (the graph's adjacency matrix) by a sparse vector (whose nonzeros mark the current frontier vertices). The example in Figure 3 discovers the first two frontiers `f` from vertex 7 via matrix multiplication with the adjacency matrix G, and computes the parent of each vertex reached. The matrix multiplication is KDT's `SpMV` primitive. The KDT programmer can specify what operations should be used to combine edge and vertex data, which is akin to specifying edge and vertex visitors.

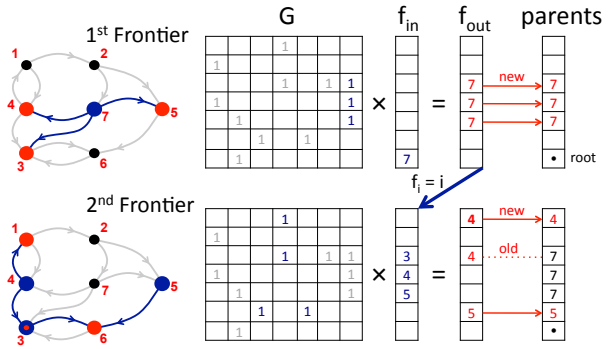


Fig. 3: Two steps of breadth-first search, starting from vertex 7, using sparse matrix-sparse vector multiplication with "max" in place of "+".

The Graph500 benchmark [4] measures the speed of a computer doing a BFS on a specified input graph in *traversed edges per second* (TEPS). The intent is to rank computer systems by their capability for basic graph analysis, just as the Top500 list ranks systems by capability for floating-point numerical computation. KDT's absolute TEPS scores are competitive; the purpose-built application used for the official June 2011 Graph500 submission for NERSC's Hopper machine has a TEPS rating about 4 times higher (using 8 times more cores) than KDT on Hopper, while KDT is reusable for a variety of graph-analytic workflows.

Betweenness centrality (BC) is an importance measure for the vertices of a graph, where a vertex is "important" if it lies on many shortest paths between other vertices. KDT implements both exact and approximate BC by sampling starting vertices in Brandes' algorithm [2]. It constructs a batch of k BFS trees simultaneously by using the `SpGEMM` primitive on $n \times k$ matrices rather than k separate `SpMV` operations. It then backtracks through the frontiers to update a sum of importance values at each vertex. The straightforward KDT code is able to exploit parallelism on all three levels: multiple BFS starts, multiple frontier vertices per BFS, and multiple edges per frontier vertex.

PageRank is naturally structured with linear algebraic primitives and is composed of only half a page of KDT code.

Our Gaussian Belief Propagation (GaBP) [1] compared favorably to the GraphLab [5] GaBP implementation on a 32-core shared-memory system for solving finite element mesh linear systems.

After initial explorations to understand the Markov Clustering [6] algorithm and KDT well, an undergraduate student produced our Markov Clustering routine in only six hours.

III. KDT Architecture

KDT's productivity benefits extend beyond simply providing an opaque set of built-in graph algorithms.

The `kdt` Python module exposes two types of classes: graph objects and their supporting linear algebraic objects. It includes classes representing directed graphs (`DiGraph`), hypergraphs (`HyGraph`), as well as sparse matrices (`SpParMat`), sparse vectors (`SpParVec`) and dense vectors (`ParVec`). Computation is performed using a set of pre-defined patterns:

- Matrix-Matrix multiplication (SpGEMM), Matrix-Vector multiplication (SpMV)
- Element-wise (EWiseApply)
- Querying operations (Count, Reduce, Find)
- Indexing and Assignment (SubsRef, SpAsgn)

Each one is implemented for parallel execution and accepts user-defined callbacks that act similarly to visitors yet follow pre-defined access patterns that account for the bulk of processing time. This allows KDT code to appear serial yet have parallel semantics.

The sparse matrix and vector classes that support the graph classes are exposed to allow complex matrix analysis techniques (*e.g.*, spectral methods). User-defined callbacks can take several forms. KDT operations accept unary, binary and n-ary operations, predicates, and semiring functions. Each one may be a built-in function or a user-written Python callback or wrapped C routine for speed.

KDT's backend is the Combinatorial BLAS [3], which is a proposed standard for combinatorial computational kernels. It is a highly-templated C++ library and provides excellent and highly scalable performance on distributed-memory HPC clusters.

Taken together, these building blocks and finished algorithms provide KDT with a high degree of power and flexibility.

IV. Evolution of KDT

The design of KDT intentionally separates its user-level language and interface from its computational engine. This allows us to extend KDT easily along at least two axes: an architectural axis, and a capability axis.

On the architectural axis we are currently working on two engines: one for manycore shared-address-space architectures, and one for more loosely coupled distributed-computing cloud architectures. Specialized engines for architectures like GPUs or the Cray XMT may follow.

On the capability axis, we are extending the set of algorithms and primitives that underlie KDT in various ways, including numerical computational primitives such as linear equation solvers and spectral analysis (computing eigenvalues, singular values, eigenvectors, etc.).

References

[1] D. Bickson. Gaussian Belief Propagation: Theory and Application. *CoRR*, abs/0811.2518, 2008.
[2] U. Brandes. A Faster Algorithm for Betweenness Centrality. *J. Math. Sociol.*, 25(2):163–177, 2001.
[3] A. Buluç and J.R. Gilbert. The Combinatorial BLAS: Design, Implementation, and Applications. *The International Journal of High Performance Computing Applications*, online first, 2011.
[4] Graph500. http://www.graph500.org/.
[5] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J.M. Hellerstein. GraphLab: A New Parallel Framework for Machine Learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, California, July 2010.
[6] S. van Dongen. Graph Clustering via a Discrete Uncoupling Process. *SIAM J. Matrix Anal. Appl*, 30(1):121–141, 2008.

Melody Matcher: A Music-Linguistic Approach to Analyzing the Intelligibility of Song Lyrics

Jennifer "Jenee" G. Hughes
Cal Poly San Luis Obispo
Computer Science Graduate Student
jhughes@calpoly.edu

***Abstract*—Melody Matcher is a semi-automated music composition support program. It analyzes English lyrics along with a melody, and alerts the composer of the locations in the song where the lyrics are not deterministically understandable. Basically, it's grammar- and spell-check for songs. This is significant, because very little research has been done specifically on the quantifiable measurement of English-language lyric intelligibility, other than our project.**

I. Introduction

Melody Matcher aims to replicate the human ability to identify lyrics in a song that are easily misheard. We started on this project, thinking that there would be carefully-specified research on how lyrics match melodies, mathematically. As it turned out, there was very little objective literature on the subject. Because of the lack of objective information of the subject, we had to develop our method from scratch. As we progressed through our work, we went from thinking that understandability depended only on emphasis-matching, to realizing that syllable length played a huge part as well, to realizing that there are many other musical, harmonic, and linguistic factors.

Melody Matcher analyzes the intelligibility of song lyrics by investigating several root causes:

- Lyric/Music emphasis mismatch, due to:
  - Note intervals
  - Phrase emphases
  - Word emphases
- Word "cramming", due to:
  - Syllable lengths that exceed that of note length
  - Mouth movement delta time intervals
- Word misidentification, due to:
  - Altered pronunciation of words
  - Phone similarity
    - Voicing (voiced vs. voiceless)
    - Beginning/end mouth positions
    - Type (Plosive, Fricative, affricate, nasal, lateral, approximant, semivowel)
  - Phone sequences with multiple syntactically-correct interpretations

The fully-implemented Melody Matcher program will eventually take into account all of these causes of unintelligibility. In this abstract, we will focus on lyric/emphasis mismatch, which has already been implemented and is fully functional in primary testing. The other sections have been implemented, but are not fully tested and/or integrated into the main program.

A. Target Audience and Goals

This program is to be used as a compositional aid by anyone who wants to write songs and make them sound good, technically. It should allow the song writer to focus on more subjective criteria of what makes a song "good", because it will make the structural rules of lyric composition immediately apparent.

Our hope for this project is that it will be useful to burgeoning songwriters, who have the creative spark to make wonderfully poetic lyrics, but lack the "ear" to match their lyrics successfully to music. It should be particularly helpful to songwriters who place a high emphasis on understandability of lyrics (such as parody song writers, or lyricists for musical theater).

Additionally, Melody Matcher will be useful for songwriters for whom English is a second language. While they may be a master lyricist in their native language, writing lyrics in English can be a particular challenge, since so much of lyric-writing is dependent upon knowing the cadence of the language you're writing lyrics in, and since English has no easily-discernible rules for emphasis placement in words.

II. Practical Example of Underlying Theory

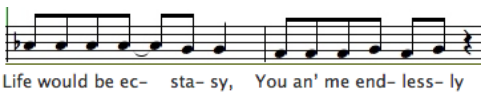
The structural rules of lyric placement are important, because without them, lyrics can become muddled and/or unintelligible. For example, in the song *"Groovin' (on a Sunday Afternoon)"*, by the Young Rascals, there's a part in the bridge that many people hear as *"Life would be ecstasy, you an' me **an' Leslie"***. In fact, the line is *"Life would be ecstasy, you and me **endlessly"***. The confusion lies with the last three syllables of the phrase. The pronunciation of each version, if spoken normally, is as follows:

Alphabetic:	and Les- lie	end- less- ly
SAMPA:	@nd "lEs li	"End l@s li

So, in the first phrase, we see that the emphasis pattern can be simplified to "dum DUM-dum", where the first syllable of "Leslie" is emphasized. The second phrase's emphasis pattern is "DUM-dum-dum", so the first syllable of "endlessly" is emphasized.

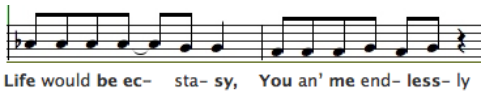
When words are put to music, however, the musical emphasis overrides the textual emphasis. Sometimes, the meaning of the phrase can change, if a previously un-emphasized syllable becomes emphasized, or a previously emphasized syllable loses its emphasis.

For "Groovin' ", the lyrics match up to the music in the song as follows:



In this musical phrase, the emphasis always goes on the first part of a beat (for the purposes of this example, a "beat" is defined as a quarter note).

In this case, the first measure is emphasized for the notes that correspond to the lyrics, "Life", "be", "ec-"(as in ec-sta-sy) and "sy"(again, as in ec-sta-sy) (This is a vast oversimplification, but it works for now). So, the lyrics would be emphasized as such:



Or, more simply:

Life would **be ec**-sta-**sy**

This musical emphasis matches the spoken emphasis of the phrase, so it is intelligible as a lyric. (Though ecstasy's first syllable doesn't start on the first part of beat three, it is still on the first part of beat three, and therefore still emphasized. Alternatively, since the first part of beat two didn't have a hard stop to it, the emphasis could have rolled over to the second part, "ec", which does have a hard stop.)

In contrast, take the second measure: the syllables "You", "me", and "less" are emphasized in the music. This leads to conflicting musical and spoken phrasing:

Musical Phrasing: **You** and **me** end**less**ly
Spoken Phrasing: **You** and **me end**lessly

The singer is now singing the phrase, syllable by syllable, which they think of as syllable-note combinations:

YOU and ME end LESS lee

The singer, for his part, is doing what many singers are taught to do, to make it easier to sustain the singing of words that end with unsingable consonants: the unsingable consonant is displaced onto the front of the next word. In this case, the consonant "d" is not singable, so he displaces it onto the next syllable, when he can: "and ME" becomes "an dME", and "end LESS" becomes "en dLESS". So, the singer can effectively think of the sung phrase as:

YOU an dME en dLESS lee

This doesn't cause confusion for listeners, because they're used to hearing it. This does mean, however, that lyric placement does not provide an accurate barometer to a listener of where a word actually ends.

In addition, the singer is singing fudging his vowels, like singers are taught to do, so "and" and "end" sound almost indistinguishable. So, really, what listeners are hearing is this:

YOU en dME en dLESS lee

Now, the listener's brain has to take this syllabic gobbledygook, and parse it into something useful. They've currently got this mess to deal with (represented in SAMPA syllables):

ju *En* ***dmi*** *En* ***dl@s*** *li*

They parse the first part just fine, because the emphases match:

you and **me** *En* ***dl@s*** *li*

But no one says endLESSly. People say ENDlessly. So, the listeners don't recognize it. They have to work with what they have. They already turned one "En d" into an "and", so they do it again:

you and **me** and ***l@s*** *li*

Now, they're just left with LESS lee. And that fits Leslie, a proper noun that fits in context and in emphasis placement. So, the final heard lyric is:

you and **me** and **Les**- lie

The misunderstanding can be traced back to improper emphasis placement. The songwriter probably didn't even think of that, and now he's stuck: a one-hit-wonder with a misunderstood song. We bet that in interview after interview, someone asks him who Leslie is. It's probably very frustrating — especially since he could have just moved the word an eight note later, and it would have been understood perfectly.

That's the sort of situation this program is going to help avoid.

III. Future Work

We plan to continue developing and refining the methods through which Melody Matcher makes its determinations. Eventually, we plan to use this as an underlying framework for an interactive virtual environment where your surroundings are affected and created via musical and lyrical input. This should be completed in March 2012, with incremental updates discussed on www.melodymatcher.com.

IV. Conclusion

In this paper, we have discussed at a high level parts of the music-lingustic approach that Melody Matcher takes to measure the intelligibility of lyrics. We covered some of the major reasons that lyrics get misheard, along with a few examples. Melody Matcher's specific implementation details, while fully specified elsewhere, were outside the scope of this abstract, and we hope to cover them in a later paper.

The Composition Context in Point-and-Shoot Photography

Daniel Vaquero and Matthew Turk
University of California, Santa Barbara
Email: {daniel,mturk}@cs.ucsb.edu

***Abstract*—We present the first study on the process of framing photographs in a very common scenario: a handheld point-and-shoot camera in automatic mode. We call the contextual information given by viewfinder images, their capture parameters, and inertial sensor data collected while the user is framing a photograph the "composition context" of the photograph. By silently recording the composition context for a short period of time while a photograph is being framed, we aim to characterize typical framing patterns. This includes adjusting the camera's orientation and point of view and triggering zoom and autofocus controls. We have collected a large database of framing instances by asking users to take pictures of scenes representing common photographic situations. Our preliminary results indicate that significant and useful variations in capture parameters exist in this dataset, such as field of view, exposure time, focus, zoom, and presence of moving subjects. We plan to use the results of this study to help create new camera functionality. The new camera will preserve the interface of current point-and-shoot cameras. However, it will silently record the composition context and use it to provide the user with alternative photographs of the captured scene, computed by exploring the variations in capture parameters present in the composition context. We expect this capability to expand the photographic abilities of casual and amateur users, who often rely on automatic camera modes, without changing the widespread point-and-shoot paradigm.**

***Keywords*-point-and-shoot photography; human factors; picture framing; contextual imaging; computational photography**

I. Introduction

With the recent popularization of digital cameras and cameraphones, everyone is now a photographer, and the devices provide new opportunities for improving the process and final results. While there has been research on what kinds of subjects users prefer to photograph, and what they do with the images once they are captured [1], no formal studies on the process of framing an image using a camera have been performed. To fill this gap, our study attempts to characterize the actions performed by users while framing photos using a point-and-shoot camera, in preparation for taking a photograph. This includes adjusting the camera's orientation and point of view and triggering zoom and autofocus controls.

II. User Study

We implemented a camera application based on the Frankencamera architecture [2], running on a Nokia N900 smartphone. It mimics the interface of a point-and-shoot camera in "automatic" mode, with a digital viewfinder, automatic metering algorithms, and controls for zoom and autofocus. A sensor box containing accelerometers and gyroscopes is attached to the camera. The camera silently records viewfinder frames (at 25 frames per second) and their capture parameters, as well as inertial sensor data, for the immediate 18 seconds before the user triggers the shutter to capture a photograph. We call the viewfinder frames, their capture parameters, and inertial sensor data collected while the user is framing a photograph the **composition context** of the photograph.

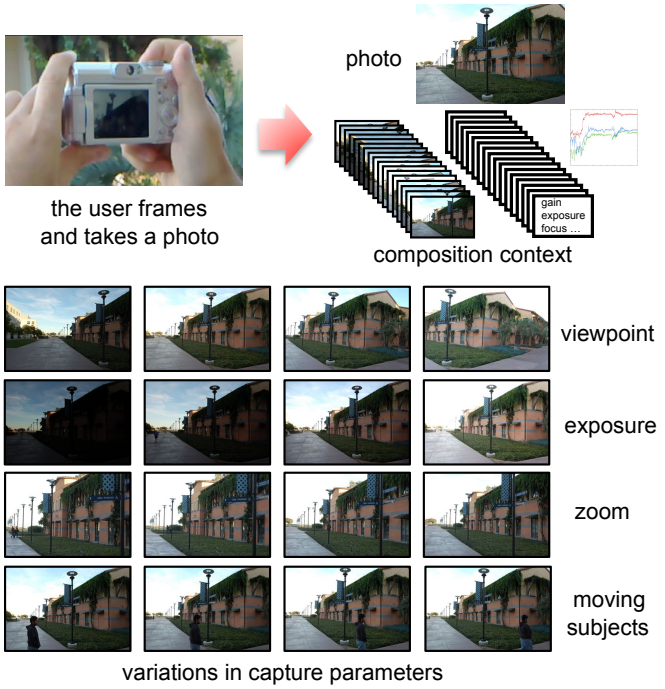


Fig. 1. By silently recording viewfinder images and their capture parameters for a short period of time while a photograph is being framed, we aim to characterize framing patterns. We have collected a large database of framing instances by asking users to take pictures of scenes representing common photographic situations. Our preliminary results show that significant and useful variations in capture parameters exist in this dataset.

We now report preliminary results, already published as a poster [3], and provide additional details on the current state of the project[1]. Initially, we recruited nine volunteers to participate in the user study. The sessions were conducted at the UCSB campus, with variations in times of day and weather conditions. Participants were provided with our camera and asked to take three pictures for each of seven different

[1]A supplementary video, submitted with [3], can be downloaded from http://www.cs.ucsb.edu/~daniel/publications/conferences/siggraph11/VaqueroSIGGRAPH11.mp4

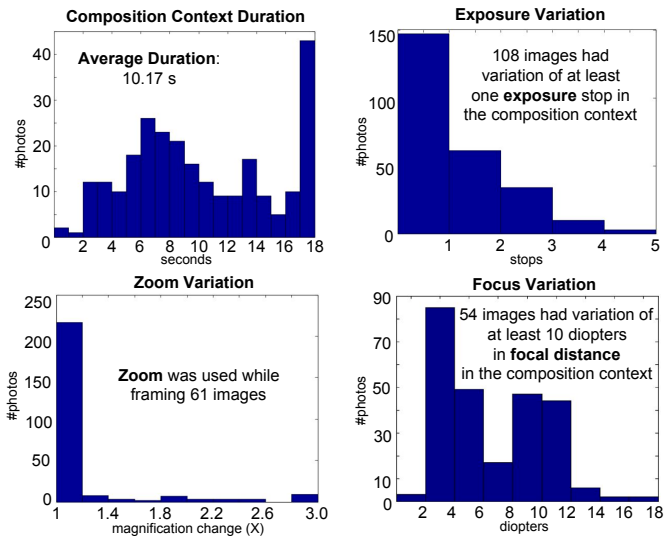


Fig. 2. Preliminary results: statistics from the collected data.

categories that represent common photographic situations: an office environment, a close-up scene, a building, a sign, an open area, a posed picture of a person or group of people, and a moving subject. They were also instructed to try their best to capture compelling images, relying on their own sense of what makes a good picture. Once these 21 pictures were taken, the users were requested to take at least five additional pictures of scenes chosen at their discretion.

A preliminary analysis of the collected data indicates interesting characteristics (Figs. 1 and 2). In this analysis, we considered only the composition context frames that overlap with the final photograph. For a total of 255 pictures, the average duration of a composition context was of 10.17 s. The camera was typically held still, with small movement due to handshake, during short intervals before capture, interrupted by sharp translations or rotations, due to attempts to adjust the composition; this effectively constitutes variations in field of view. 108 photos had exposure variations of at least one stop in their composition contexts due to autoexposure, 54 photos had at least 10 diopters of variation in focus due to autofocus, and zoom was used for framing 61 photos. Moving subjects were present in the composition context of 172 pictures, even though the main subject intended to be photographed was static in most of the pictures.

After this initial study, we proceeded with data collection for more users. We have gathered data from study sessions of 45 participants, and we are working on a detailed analysis, including additional statistics (e.g., quantifying variations in field of view) and correlations to answers provided to a questionnaire of photography knowledge.

III. Implications: Composition Context Camera

We plan to use the results of this study to help create new camera functionality (Fig. 3). From the user's point of view, the camera would preserve the interface of current point-and-shoot cameras, and it would still be used in the same way. However, when users trigger the shutter, instead of obtaining

Fig. 3. By silently recording viewfinder frames and their capture parameters while a photograph is being framed, we enable the automatic generation of variations of the photograph. Our method expands the end result of the photo capture process; instead of obtaining a single photograph, the user may also receive a collection of photo suggestions from the same scene, without requiring additional effort while framing the photograph.

a single photograph, they may also obtain additional photo suggestions from the same scene, created through automatic combination of composition context frames captured under varying parameters [4]. Examples of suggestions that can be generated include panoramas, collages, extended dynamic range, selective focus, synthetic long exposure, synthetic panning, motion trajectories, and moving object removal. Also, alternative views and moments could be suggested by optimizing computational aesthetics measures (such as [5] and [6]).

A complete software solution would include the following steps: (i) automatic image alignment, using, e.g., [7]; (ii) detection of moving areas; (iii) generation of the photo suggestions. With the rapid advances in computational resources of portable devices, we envision that this process could eventually be implemented entirely on a camera. As our study shows that variations in capture parameters in the composition context happen naturally in the point-and-shoot process, no additional input or familiarization with new interfaces would be required from the user. We expect this capability to expand the photographic abilities of casual and amateur users, who often rely on automatic camera modes, without changing the widespread point-and-shoot paradigm.

References

[1] N. A. Van House, M. Davis, M. Ames, M. Finn, and V. Viswanathan, "The uses of personal networked digital imaging: an empirical study of cameraphone photos and sharing," in *CHI '05 Extended Abstracts*, 2005, pp. 1853–1856.

[2] A. Adams, E.-V. Talvala, S. H. Park, D. Jacobs, B. Ajdin, N. Gelfand, J. Dolson, D. Vaquero, J. Baek, M. Tico, H. Lensch, W. Matusik, K. Pulli, M. Horowitz, and M. Levoy, "The Frankencamera: an experimental platform for computational photography," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 29, no. 4, pp. 1–12, 2010.

[3] D. Vaquero and M. Turk, "The composition context in point-and-shoot photography," *ACM SIGGRAPH Posters*, 2011.

[4] R. Raskar, "Computational photography: Epsilon to coded photography," *Emerging Trends in Visual Computing: LIX Fall Colloquium, France. Revised Invited Papers*, 2008.

[5] Y. Ke, X. Tang, and F. Jing, "The design of high-level features for photo quality assessment," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2006, pp. 419–426.

[6] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Studying aesthetics in photographic images using a computational approach," in *European Conf. on Computer Vision*, 2006, pp. 7–13.

[7] M. Brown and D. Lowe, "Recognising panoramas," in *IEEE Intl. Conf. on Computer Vision*, 2003, pp. 1218 –1225.

iSketchVis: Integrating Sketch-based Interaction with Computer Supported Data Analysis

Jeffrey Browne*†, Bongshin Lee*, Sheelagh Carpendale*‡, Timothy Sherwood*†, Nathalie Riche*
*Microsoft Research †University of California, Santa Barbara ‡University of Calgary

***Abstract*—When faced with the task of understanding complex data, it is common for people to step away from the computer and move to the whiteboard, where they may easily sketch simple visualizations. In this work, we present *iSketchVis*, with the goal of making complex underlying data directly available at this point of discussion—the whiteboard itself. With iSketchVis, people draw the basic structure of a visualization, such as the labeled axes of a graph, and the system augments this outline with their own data and further options. We describe the interactive experience resulting from an iterative, feedback-driven design process, and discuss the challenges posed by this new domain of visualization.**

I. Introduction

In our professional and leisure activities, we must manage and understand more data, both individually and collaboratively. One increasingly common way of making sense of this data is through visualizations, which have the potential to leverage the viewer's innate perceptual capabilities to more easily gain insights into complex information. Though powerful visualization tools exist, when first brainstorming about the aspects of data to be explored, or when multiple people are discussing a data set, people often initially work on whiteboards. However, since plotting data in these sketches would require tedious repetition, the underlying data usually remain buried in a computer, where interaction is constrained to classic mouse and keyboard.

In this paper, we explore bringing data to sketched whiteboard visualizations, enabling information to be explored and examined in more fluid, natural ways, yet without interfering with the traditional advantages of the whiteboard. We present iSketchVis, a sketch-based visualization system that enables creation of charts and direct interaction with data through simple, computationally supported sketches. The core idea is to augment the initial structure of a bar chart or scatter plot drawn on a whiteboard with simple visualizations of actual underlying data. After they are plotted within the sketched reference frame, the data can be transformed using methods such as changing the labeling of the axes, filtering particular classes of data, and applying mathematical functions.

II. iSketchVis Interface

We illustrate the iSketchVis interface with a simple usage scenario by following Alice, a researcher who wishes to explore crime rates across seven countries. Before beginning sketching, Alice loads her data set (in comma separated variable form) using a file selection dialog. Interaction with iSketchVis then proceeds solely through a digital pen on a wall-sized presentation screen simulating a whiteboard.

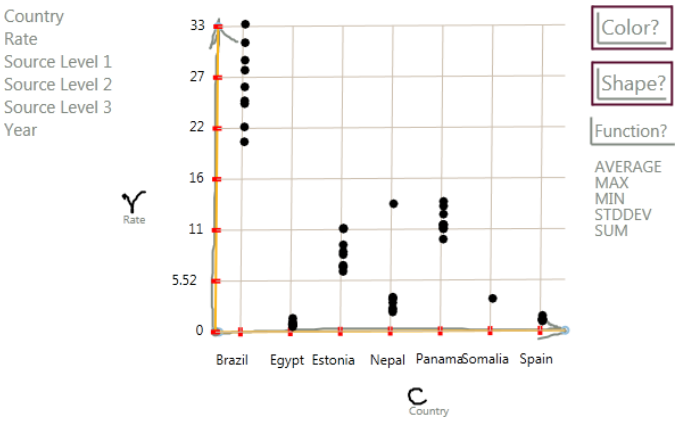


Fig. 1: Two single-stroke arrows specify the size and location of a chart. Handwritten axis labels are inferred from the list of columns in the data file.

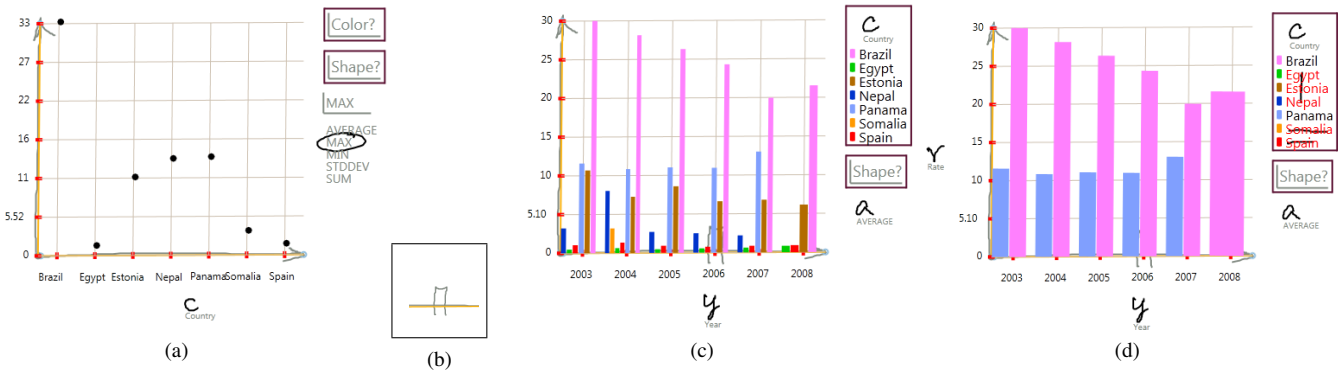
A. Chart Initialization

To start exploration, Alice *draws* two intersecting arrows for X and Y axes as she would draw a chart on a whiteboard; the size, location, and orientation of these arrows determine the physical dimensions of the chart. When iSketchVis recognizes a pair of arrows, it provides the list of data fields populated from the the data file as a side legend in case she does not recall exactly the names of each data field (Figure 1). iSketchVis also provides several text-entry prompts near the *X axis*, *Y axis*, as well as for the *color legend*, *shape legend*, and the *function selector*.

X and Y axes are mapped by matching handwritten text to the first data field that starts with the specified letters, at which point iSketchVis draws straight lines over the hand-drawn axes and displays the specified data between the arrows in a scatter plot, with labeled tic-marks and grid lines to show scale.

B. Data Transforming Functions

From the initial scatter plot, Alice wants to compare the countries' maximum crime rates. She *circles* "MAX" in the *function selector* menu (Figure 2a), and iSketchVis shows maximum values for each country. Once a function is selected, iSketchVis groups points by X values, determines the Y values as the result of the function over the grouped Y values, and displays them. Now that she has compared maximum crime rates, Alice wishes to view the average rate for each country, and thus she *erases* the circle mark and selects "Average" by writing an "a" in the function entry box (Figure 2c), just as



(a) (b) (c) (d)

Fig. 2: Transformation functions are selected either by circling an option (2a) or by writing in the function entry box. Drawing a bar stroke across the X axis (2b) converts the default scatter plot into a bar chart. Values from the data file are drawn with colors as mapped in the color axis legend (2c). Crossing out values in the legend axes filters out data points (2d).

she did to specify axes. Then, iSketchVis updates the view to display average values for each country. iSketchVis currently covers a basic set of functions: average, standard deviation, sum, maximum, minimum, and count.

C. Changing Chart Type

To visualize the relative changes in crime rates between different countries more clearly, Alice then *draws a bar shape* across the X axis (Figure 2b), and the system converts the chart into a bar chart.

D. Color and Shape Distinction

She now wants this information broken down by country, so she writes a "c" in the *color legend* area. Upon selection, the box expands into a legend where each value in the column is mapped to a color, and each point in the plot is colored according to its value in that column (Figure 2c). She then notices that, while Brazil's rate tended to decrease, Panama's rate was actually increasing.

If Alice wishes to further distinguish each year's rate by its data source, she can write an "s" to map "Source Level 1" to the *shape legend* area, and each point will be given a symbol (colored by country) according to its source.

E. Data Filtering and Axis Scaling

To focus on the difference between Brazil and Panama more clearly, Alice *crosses out* (i.e., draws a line through) all other countries (Figure 2d) from the color legend. iSketchVis hides points with values matching the crossed out value and automatically adjusts the scale of the X and Y axes to fit the remaining data.

III. Discussion

Data exploration through sketch-based charting is a new information visualization paradigm. Traditionally, novices have had to put significant effort into generating a single chart, so analysts are used to having a preconception of what their data could look like before they actually see it. By lowering the barrier to creating new charts, iSketchVis allows people to visualize data sets before they understand the meaning of what they plotted. For some of our participants, the speed with which they charted exacerbated their difficulty understanding their data. However, others found the speed and direct nature of our system's sketch interaction as a great boon. One participant, self-identified as a novice to charting and data exploration, compared iSketchVis to chart generation tools she had used before saying, "[iSketchVis] feels like I can get my head around it."

As the first version of iSketchVis, the current implementation has some limitations. While some participants initially had trouble with drawing single-stroke arrows the system could recognize, everyone managed to consistently draw recognized arrows after no more than ten failed attempts in the beginning of the session. Future versions of iSketchVis should support multi-stroke arrows, and we are investigating methods to speed up this recognition.

Additionally, iSketchVis currently only supports two chart types: scatter plot and bar chart. While an ideal system will be much more capable (supporting pie charts, line charts, etc.), people in our participatory design sessions still made significant progress in analyzing their data with only the two supported chart types.

Applying sketch-based interaction to information visualization data charting shows great promise, and many of the metaphors from charting on traditional whiteboards appear to transfer readily to computationally supported whiteboards. However, much work remains to be explored in the extent to which sketch-based interactions support charting.

Reliable Selection of Interest Points for Keypoint Matching

Victor Fragoso Computer Science Department
University of California, Santa Barbara
vfragoso@cs.ucsb.edu
Matthew Turk Computer Science Department
University of California, Santa Barbara
mturk@cs.ucsb.edu

***Abstract*—Establishing correspondence between a reference image and a query image is an important task in computer vision, as it provides information for further computation such as image alignment, object recognition, visual tracking, and other tasks. Normally, the correspondence between images is addressed associating keypoints. Keypoints are locations over the images that possess distinctive information over a pixel neighborhood (patch). In order to produce a better correspondence, current keypoint detectors pretend to locate points that are repeatable given image variations (e.g., view angle, scale, rotation, etc.). However, the correspondence produced by a matcher, the agent that finds correspondence, is affected regularly because the keypoint detector is unable to find points previously detected and/or detects new points given images that present some variation. Consequently, the produced correspondence by the matcher is not perfect, therefore, errors can be propagated in further steps. In this work, we aim to reduce the mismatches produced by the matcher. A confidence measure is assigned to a pair of associated keypoints, and used to discard pairs that present a low confidence. Our ongoing research results indicate that the measure reduces keypoint mismatches at a good rate.**

***Keywords*-keypoint matching, meta-recognition, post-score analysis, ferns**

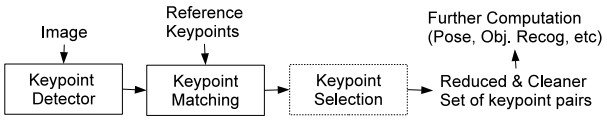


Fig. 1. Overall process of keypoint matching & keypoint selection. The selection block (dashed box) acts as a filter producing a reduced and cleaner set of keypoint pairs produced by the matcher.

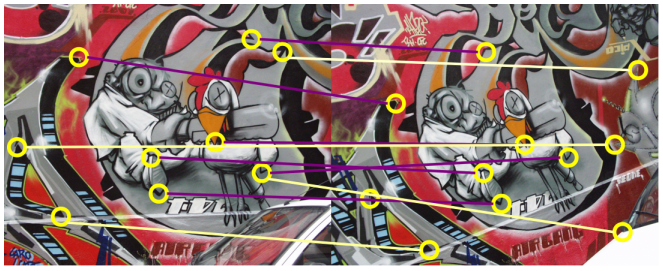


Fig. 2. Keypoint pairs and its confidence: Reference Image (Left) and Query Image (Right). Keypoints detected on both images are circled and a line connecting points represent the association computed by the matcher. Lines in magenta denote a predicted correct match while yellow lines denote a predicted incorrect match. The predictor does some mistakes, however, overall it detects correct and incorrect matches at a good rate.

I. Introduction

Keypoint matching is the process that determines the correspondence between interest points detected on a "reference" image and a "query" image (see Figs. 1 and 2). This correspondence is important as it provides relevant information that is widely used in several tasks. In image alignment, for example, keypoints detected on a query image are "matched" against keypoints detected on a reference image. This matching is then used to compute a transformation matrix that maps the location of every pixel from the query image to the corresponding location on the reference image with a minimal error. However, keypoints detected on a reference image are not necessarily detected on a query image. Moreover, new keypoints can be detected on a query image, causing the matcher to produce mismatches. In this work, we aim to reduce the number of mismatches in order to decrement the error in posterior computation.

Matching two keypoints is normally done using descriptors. A descriptor is a representation of a keypoint computed from a surrounding patch of such point. A descriptor is commonly an n-dimensional vector (e.g., SIFT [1], SURF [2]). Therefore, matching keypoints is the process of finding the closest descriptor of a query descriptor $\mathbf{x}$ over all reference descriptors $\mathbf{y_c}$ corresponding to a reference keypoint c, i.e., a match is found when the distance between $\mathbf{y_c}$ and $\mathbf{x}$ is minimal.

Computing descriptors is computationally expensive as the algorithms are designed to produce robust representations to several image variations (e.g., scale, view angle, rotation). To alleviate this problem, Lepetit and Fua [3] posed the matching process as a recognition task, which implies a training phase considering every keypoint as a unique class, and also introduced a novel and fast-to-compute algorithm for representing a patch. Keypoint recognition trains a Bayes classifier that learns binary encoding of a keypoint's patch with a randomized tree. The binary encoding is based on image-intensity-differences between random locations over the patch: assigning 1 if the difference is positive, and 0 otherwise. A few years later, Ozuysal et al. [4] introduced a new binary patch encoder which used simpler and faster-to-compute structures called "Ferns." These structures encode a patch using the same criteria of

binarizing pixel differences as mentioned earlier. However, the structures are much simpler than a randomized tree, making it faster to compute and reducing memory footprint.

As described earlier, a matching process produces mismatches affecting further computation. To alleviate this issue, we propose to assign a confidence value for every pair produced by the matcher. Therefore, we can discard the pairs that present a low confidence and preserve only those with good confidence (low and good scores is controlled with a threshold). The confidence measure here used was introduced by Scheirer et al. [5]. Albeit the measure was developed for ranking matches in biometric tasks, the measure still can be used for keypoint matching. A set of scores is necessary to compute the confidence measure. The scores are produced by the matcher when selecting the best match: the matcher compares the query representation (descriptor or ferns) against every reference representation and returns the best match by evaluating a distance (score). The produced scores that were computed when matching are retained and are sorted. From this sorted set of numbers only k scores are retained, i.e., the closest scores to the best match score are retained. This set is used to estimate a Weibull distribution which is used to evaluate if the best matching score is an outlier of this distribution, if it is, we declare the matcher's outcome as correct, and incorrect otherwise.

II. Experimental Results

The results here presented show the performance of the confidence measure applied to keypoint matching using Ferns. The dataset used for this experiment, Affine Covariant Features [1], contains the following items: datasets of images that present several variations (e.g., viewpoint, rotation, scale, etc.); and the corresponding geometric transformation: a 3x3 matrix that relates the reference image to a query image. The testing software used a modified OpenCV's Fern descriptor matcher which includes the confidence measure computation. 50 Ferns were used for learning the detected 600 keypoints in the model image with several keypoint detectors (see [6] for the list of detectors). We consider the closest scores ($k = 15$) to compute the Weibull distribution, and we used a decision threshold $\delta = 0.9888$ to indicate correct or incorrect match as the confidence measure has support within $[0, 1]$. The transformations provided by the dataset were used to compute the estimated position of a certain keypoint on a query image. This estimation was used to determine if the "prediction," with this configuration, was guessed correctly.

Table I shows the average confusion matrix, presented as a row in the table, of every dataset, and also shows the ratio of correct matches vs mismatches before and after selection. As observed from the results, the predictor performs well detecting true-negatives (TN) and true-positives (TP), which imply that discarding the pairs predicted as incorrect increases the aforementioned ratio. Consequently, further computation from this reduced set has more chances to compute a cleaner and accurate outcome.

[1] http://www.robots.ox.ac.uk/~vgg/research/affine/

TABLE I
Average performance of predictor with Ferns under several image variations and different keypoint detectors

Dataset	TP	TN	FP	FN	Accuracy	Ratio	Ratio after Selection
Bark	0.03	0.55	0.41	0.01	0.59	0.06	0.13
Boat	0.19	0.44	0.34	0.03	0.61	0.32	0.63
Bikes	0.40	0.30	0.21	0.10	0.70	1.41	2.60
Trees	0.17	0.44	0.36	0.03	0.59	0.22	0.41
Graf	0.17	0.48	0.33	0.03	0.64	0.34	0.70
Wall	0.22	0.41	0.34	0.03	0.63	0.46	0.90
Leuven	0.49	0.24	0.18	0.09	0.73	1.40	2.86
UBC	0.52	0.25	0.17	0.07	0.76	2.27	5.20

III. Conclusions and Future Work

The experiment suggests that the confidence measure proposed by Scheirer et al. [5] works well detecting mismatches using Ferns for keypoint recognition. However, the predictor is still incapable of detecting false alarms, as Ferns in this setup learn a patch and do not capture any geometric information that could help in discarding keypoints more accurately. More keypoint representations, specifically descriptors (e.g., SIFT [1], SURF [2]), must be used to evaluate this measure and gain a better picture of the keypoint matching process.

We would like to explore a potential extended training phase of any keypoint recognition system, that includes a stage of keypoint selection using this confidence measure: selecting only those keypoints that present good reliability among several predefined transformations. Therefore, a smaller set of reliable reference keypoints can be produced.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, June 2008.

[3] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 28, no. 9, pp. 1465 –1479, sept. 2006.

[4] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Trans. on Pattern Anal. and Mach. Intell.*, vol. 32, no. 3, pp. 448–461, march 2010.

[5] W. J. Scheirer, A. Rocha, R. J. Micheals, and T. E. Boult, "Meta-recognition: The theory and practice of recognition score analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, pp. 1689–1695, August 2011.

[6] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *Int. J. Comput. Vision*, vol. 94, no. 3, pp. 335–360, 2011.

A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns

Brett Stone-Gross[§], Thorsten Holz[‡], Gianluca Stringhini[§], and Giovanni Vigna[§]

[§]University of California, Santa Barbara
{bstone,gianluca,vigna}@cs.ucsb.edu

[‡] Ruhr-University Bochum
thorsten.holz@rub.de

***Abstract*—Spam accounts for a large portion of the email exchange on the Internet. In addition to being a nuisance and a waste of costly resources, spam is used as a delivery mechanism for many criminal scams and large-scale compromises. Most of this spam is sent using botnets, which are often rented for a fee to criminal organizations. Even though there has been a considerable corpus of research focused on combating spam and analyzing spam-related botnets, most of these efforts have had a limited view of the entire spamming process. In this paper, we present a comprehensive analysis of a large-scale botnet *from the botmaster's perspective*, that highlights the intricacies involved in orchestrating spam campaigns such as the quality of email address lists, the effectiveness of IP-based blacklisting, and the reliability of bots. This is made possible by having access to a number of command-and-control servers used by the Pushdo/Cutwail botnet.**

I. INTRODUCTION

In August 2010, we obtained access to 13 Command & Control (C&C) servers and 3 development servers (16 servers in total) used by botnet operators of the Cutwail spam engine. This software has been used by some of the most prolific spammers over the last few years, and is frequently installed by a separate Trojan component known as Pushdo. Cutwail utilizes an encrypted communication protocol and an automated template-based spamming system to dynamically generate unique emails with the goal of evading existing spam filters. Each Cutwail bot maintains highly detailed statistics about its own spam activities, which are reported back to the C&C server. The data we obtained from these C&C servers provides us with a novel, deeper insight into the *modus operandi* of cyber criminals and the dynamics behind some of the most sophisticated spam operations to-date.

What makes our research novel is the unique perspective and the depth of our analysis. As a result of having gained access to a large number of C&C servers, we are able to observe an almost complete view of how modern spam operations work. In particular, we can identify the problems that make spam operations difficult, and the value of spam from a financial point of view. In addition, we believe that our findings will improve the understanding of the amount of spam delivered by botnets.

II. THE CUTWAIL BOTNET

The original Cutwail botnet emerged back in 2007, and has evolved in sophistication from using simple HTTP requests to a proprietary, encrypted protocol. A typical Cutwail infection occurs when a compromised machine executes a so called *loader* called Pushdo, that behaves as an installation framework for downloading and executing various malware components. Depending on the victim's system configuration, the Pushdo malware will contact a C&C server, and request additional malware components. After Pushdo contacts the C&C, several malware modules are typically downloaded and installed. This commonly includes a rootkit component to hide the presence of malware on the infected system, the Cutwail spam engine, and a list of IP addresses of Cutwail C&C servers. At this point, the infected machine executes the Cutwail spam engine and becomes a spam bot. Next, the Cutwail bot will contact one of the IP addresses from the list provided through the Pushdo bootstrap process, and wait for instructions. The Cutwail C&C server provides several critical pieces of information to begin a spam campaign. More specifically, the C&C server provides the actual spam content, delivered through the use of *spam templates*, a target list of email addresses where spam will be delivered, a dictionary consisting of 71,377 entries for generating random sender/recipient names, and a configuration file containing details that control the spam engine's behavior, such as timing intervals and error handling. Optionally, a list of compromised SMTP credentials can be distributed to bots for "high-quality" spam campaigns [1]. These techniques are used by similar botnets to perform template-based spamming [2].

The Cutwail spam engine is known in spam forums by the name *0bulk Psyche Evolution*, where it is rented to a community of spam affiliates. These affiliates pay a fee to Cutwail botmasters in order to use their botnet infrastructure. In return, clients are provided with access to a web interface (available in Russian or English language) that simplifies the process of creating and managing spam campaigns (referred to by Cutwail as *bulks*). The interface includes features to fine-tune nearly every part of an email message and spam campaign. For instance, a user can choose to customize the email's headers to impersonate legitimate mail clients (e.g., Microsoft Outlook, Windows Mail, and TheBat), or may opt to define their own headers. After defining the headers, the user may define other fields, including the sender address, email subject line, and body. All of these fields can make use of *macros* that will instruct each individual bot to dynamically generate (and fill-in) unique content for each email sent in order to evade detection by spam filters, similar to other

spam botnets [2]. In order to increase the spam campaign's effectiveness, each Cutwail C&C runs a local instance of SpamAssassin, a free open-source mail filter, that uses a set of heuristics to classify spam. Once the email template has been created, it is automatically passed through SpamAssassin and can be tweaked until it successfully evades detection. After creating the spam message, the user must specify several parameters such as a target email address list, a configuration file that controls a number of bot parameters (e.g., sending rates, timeouts, retries, etc.), and the time when the spam campaign will commence. If a Cutwail user requires assistance, they can refer to an instruction manual that is included, or contact Cutwail's support team.

III. Data Collection

The primary tool that we utilized was ANUBIS [3], a framework for dynamic, runtime analysis of binary programs. ANUBIS runs a Windows executable and records during runtime the program's behavior such as file system modifications and network activity. At the time of writing, the system processes tens of thousands of malware samples per day and offers us an insight into the latest malware trends [4]. By searching through the ANUBIS database, we were able to identify 30 distinct Cutwail C&C servers based on their unique communication signatures. We then contacted the hosting providers whose servers were being used for controlling the botnet. We provided them with evidence that specific servers within their network were used for malicious purposes and requested the take down of these servers.

As a result of our notification and mitigation steps, more than 20 servers were shut down and we were able to obtain access to 16 servers used by Cutwail controllers from some of the hosting providers. These servers contained a wealth of information, including:

- More than 2.35 TB of data.
- 24 databases that contain detailed statistics about the infected machines and overall spam operations.
- Spam templates and billions of target email addresses for spam campaigns.
- The botnet's source code and a highly detailed instruction manual for botnet operators.

IV. Analysis of the botnet

It is evident from the content on these servers that there are several different crews renting these botnets. By analyzing the data on the C&C servers, we found that, on average, there were 121,336 unique IPs online per day, and 2,536,934 total IPs observed over the whole analysis time frame. India (38%), Australia (9%), Russia (4%), Brazil (3%), and Turkey (3%) account for the largest number of spam bots. One possible explanation is that the Cutwail controllers may specifically target Indian machines because the cost per bot is cheaper than those in other geographic regions.

The most interesting information retrieved from the C&C servers was stored in the databases containing meticulous records for each spam bot. More specifically, the botnet controllers maintain detailed statistics per infected machine (identified via a unique IP address) in order to measure the effectiveness of their spam campaigns. We found that a spammer's job is complicated by a number of factors including invalid email addresses, SMTP errors, and blacklisting. As a result, the amount of spam that was actually delivered (i.e., accepted by mail servers) was only around 30.3%, and the actual volume was likely much less after client-side spam filters are taken into account. This delivery rate is slightly higher than the 25% delivery rate of the *Storm* botnet [5].

The largest cause of failure was invalid email addresses accounting for 53.3% of errors, followed by SMTP blacklisting (16.9%), miscellaneous STMP errors (11.8%), and connection timeouts (11.3%). Interestingly, 3.5% of mail servers notified the sender (in this case, the bots), that the content of the email was flagged as spam. Despite these complications, the amount of spam that is sent by Cutwail bots is immense. During one period from July 30, 2010 and August 25, 2010 the database records show 87.7 billion emails were successfully sent.

Overall, records contained on these Cutwail servers dated as far back as June 2009 and reported 516,852,678,718 messages were accepted for delivery out of a total of 1,708,054,952,020 attempts. Note that we obtained roughly one-half to two-thirds of the active Cutwail C&C servers, so the overall numbers are likely higher.

The content of the email messages sent by Cutwail included pornography, online pharmacies, phishing, money mule recruitment, and malware. The malware (e.g., the ZeuS banking Trojan) is typically distributed by enticing a user to open an attachment in the form of a greeting card, resume, invitation, mail delivery failure, or a receipt for a recent purchase. In addition, many of the emails contained links to malicious websites that attempted to covertly install malware on a victim's system through drive-by-download attacks. Cutwail operators also advertised content to Russian speakers such as real estate and ski resorts.

V. Conclusions

We believe that these insights will improve the security community's understanding of the underground economy. In addition, the data that we provide can be used to validate or refute results built on simulation, or by speculations based on the observations of subsets of botnet components.

References

[1] **C. Nunnery, G. Sinclair, and B. B. Kang** , Tumbling Down the Rabbit Hole: Exploring the Idiosyncrasies of Botmaster Systems in a Multi-Tier Botnet Infrastructure, In *USENIX LEET Workshop*, 2010.

[2] **A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G.M. Voelker, V. Paxson, N. Weaver, and S. Savage**, Botnet Judo: Fighting Spam with Itself. In *Network & Distributed System Security Symposium*, 2009.

[3] **International Secure Systems Lab**, Anubis: Analyzing Unknown Binaries. *http://anubis.iseclab.org*.

[4] **U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel**, A View on Current Malware Behaviors. In *USENIX LEET Workshop*, 2009.

[5] **C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage**, Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In *ACM CCS* 2008.

Back to the Future: Replaying Malicious Web-pages for Regression Testing

Yan Shoshitaishvili, Alexandros Kapravelos, Christopher Kruegel, Giovanni Vigna
Computer Science Department
UC Santa Barbara
Santa Barbara, USA
{yans, kapravel, chris, vigna}@cs.ucsb.edu

I. Abstract

The arms race between malicious page detection and the evasion of that detection has produced a lot of historical data with regards to the evolution of malicious pages. In this paper, we present *DeLorean*, a system for tracking malicious pages' evolution through time in order to improve analysis and detection techniques. While doing so, we overcome several challenges ranging from tool and library limitations to clever attempts by the writers of malicious pages to detect and prevent our analysis.

II. Introduction

In recent years, exploitation of unsuspecting webizens by malicious web pages has become one of the largest vectors enabling the spread of malware. Consequently, an arms race has arisen between the malware authors and the good guys (governments, traditional security companies, and also entities such as Google and Microsoft which have stakes in the struggle). As the techniques for detection of malicious pages and for the evasion of that detection improve, the tools on both sides and the pages themselves change. Additionally, malicious pages are frequently removed from their hosts as they are detected, so a once-detected malicious page may no longer be in existence at a later date. These changes in malicious pages make it very difficult to test web security tools as new versions are developed, since the sites that the old versions were tested against may no longer exist. Thus, a utility for emulating the actions of a malicious page is needed to provide an environment for reliable regression testing of evolving security tools.

One tool that will benefit from such functionality is Wepawet [3], deployed at UCSB to analyze submitted pages and determine which of them are malicious. It is a perfect example of a tool that needs regression testing support throughout its development – as modifications are made to the detection engine, old pages need to be re-analyzed to ensure that there is no degradation of detection capabilities.

In this paper we propose *DeLorean*, a system that we developed to provide malicious web page regression testing and allow web security tools to better analyze and understand how malicious web pages work. DeLorean provides this functionality in a transparent way, requiring (depending on its configuration) few to no changes on the client end. This allows our system to be effectivelly used without having to setup a complicated infrastucture to provide the service. It provides the capability to accurately replay the actions of a malicious website, by treating non-deterministic replays with best effort and managing to serve requests that were not identical in the original run by serving the most appropriate replacements. It also supports several configurations to fit the client's requirements, and handles parallel replays of malicious websites to allow for increased regression testing efficiency.

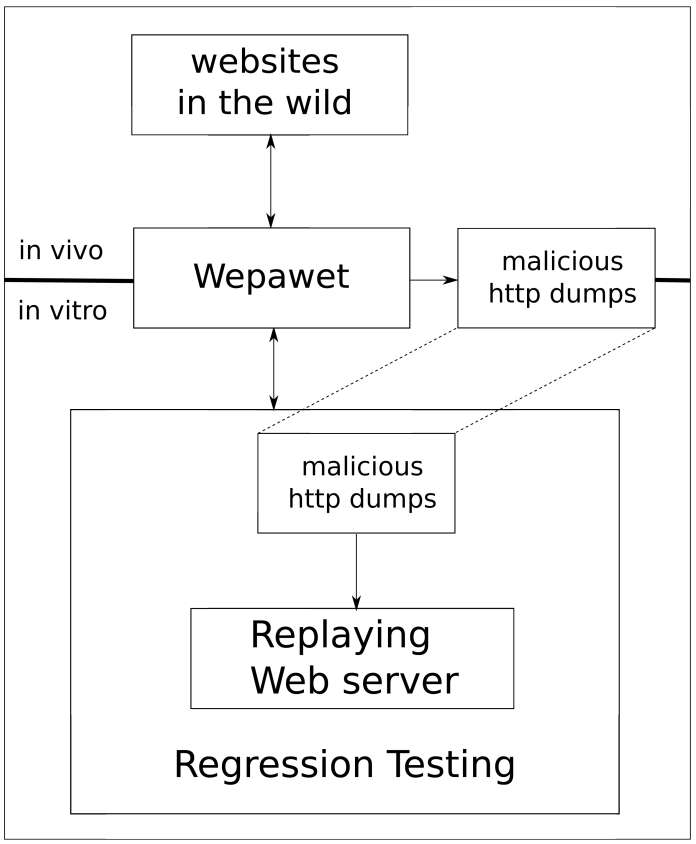


Fig. 1. Regression Testing Architecture

III. Architecture

The first step to replaying the actions of a malicious website is the recording of its actions when it is running on the original hosts. This is provided for us as part of Wepawet's

standard operation. An HTTP dump is saved of all data sent throughout the processing of a request for the page, including any redirects, specific HTTP errors, and all supporting data such as CSS and JS files. These dumps are subsequently parsed by DeLorean to provide the data that will eventually be replayed to the client. URLs are noted, and counters are kept for URLs that are requested more than once, in case the response is different throughout the multiple calls to such pages.

When the client requests a page for the nth time, the replaying server serves the nth occurrence of that page from the HTTP dump. This page may be an actual HTML page, a CSS or JavaScript resource, or even a formatted error page recorded from the original run. In all cases, the HTTP headers are reproduced from the old log, including the HTTP response codes.

One complication that was encountered was the propensity on some malicious pages to conduct chain redirects through randomized hosts. The JavaScript contained in the page would generate an IP address based on several parameters and redirect to it, and the resulting page would do likewise until the final malicious page was reached. Since these generated IP addresses can be different from those encountered during Wepawet's original run, we may not necessarily have their responses in the HTTP dump, and the replaying server would be at a loss as to what to respond with. The implemented solution, in this case, checks for that URL across all of the hosts in the dump if the requested host is not present. If it finds a similar URL on a differing host, that request is returned instead. This allows such redirect chains to process through to their ends, allowing DeLorean to handle such a case.

DeLorean can operate in several modes, depending on the client's requirements, making it suitable for multiple environments. The first mode is a proxy-based approach in which the client simply uses the server on which DeLorean is running as its proxy. All requests are thus routed to the replaying server, and the malicious web page is replayed to the client. The advantage of this approach is that it does not interfere with other workload running on the client, since proxies can generally be set on a per-application basis.

However, the proxy approach is sometimes inadequate. Specifically, Flash and Java applets can ignore or detect proxy settings, and certain clients (for example, embedded systems) may not support proxies at all. For these cases, DeLorean provides a proxy-less, transparent IP sinkhole configuration. The client (or the client's router) is configured such that any outbound packets, regardless of their initial destination, are redirected to the machine running the replaying server. This way, DeLorean can receive and respond to all requests. With this approach, applications on the client would be unaware of the change.

To facilitate efficient regression testing, DeLorean also provides support for replaying web pages to multiple clients in a concurrent fashion. This is achieved via a small callback program running on the client. Whenever the replaying server receives a request for a page, it connects back to the client host and requests an identifier for the request. The client examines its host's connection data, identifies the PID of the process which originated the request, and responds to the replaying server with an identifier consisting of the PID and its hostname. DeLorean thus tracks the page request history separately for each process on every host, allowing multiple malicious page replays to be carried out simultaneously.

IV. Related Work

Chen et al. [2] proposed a system for automated collection and replay of web-based malware scenarios called WebPatrol. The main weakness of their approach is their use of proxy authentication to isolate different clients. Our approach to concurrency is an improvement over WebPatrol's method of proxy authentication due to the fact that certain applications, like Google Update [1], though containing proxy support, do not support proxy authentication. As a result, any such applications will fail to work with concurrency through WebPatrol. One should also take into consideration the scalability and deployment issues of using different credentials per client and the fact that using our approach, we can have hundreds of clients (browsers) on a single machine using the regression testing service simultaneously.

References

[1] Google Update lack of proxy authentication. http://www.google.com/support/pack/bin/answer.py?answer=56921.

[2] K. Z. Chen, G. Gu, J. Nazario, X. Han, and J. Zhuge. WebPatrol: Automated collection and replay of web-based malware scenarios. In *Proceedings of the 2011 ACM Symposium on Information, Computer, and Communication Security (ASIACCS'11)*, March 2011.

[3] M. Cova, C. Kruegel, and G. Vigna. Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code. In *Proceedings of the International World Wide Web Conference (WWW)*, 2010.

Generating Applications for the Cloud from Formal Specifications

Christopher Coakley, Peter Cappello
University of California, Santa Barbara
{ccoakley, cappello}@cs.ucsb.edu

I. Introduction

Web Applications are ubiquitous. Unfortunately, so are errors in implementation and design, as highlighted by well-publicized security breaches and system outages. Fortunately, there are substantial efforts to make developing more robust web applications possible.

Model-View-Controller analysis restrictions. Many modern web applications are built using a specific framework assuming a specific architecture. This specificity is used restrict to general software verification techniques to make them scale better or require less developer effort. By assuming an application is built using specific semantics of a Model-View-Controller architecture, application behavior is analyzed by looking at only a small subset of the total application code, as in [1]. They analyze Rails model definitions and translate them into Alloy [2] for bounded verification of manually specified properties. Unfortunately, properties that cannot be inferred from the model relationship definitions cannot be used in the verification process. For example, if a delete function implemented cascade delete behavior in Ruby code instead of simply using the :cascade symbol in its Rails relationship definition (a pathological example), the analyzer cannot know that this behavior exists.

Model Driven Architecture. By specifying application behavior with domain-specific languages, the OMG hoped to ease the analysis of applications. UML supports the introduction of *stereotypes*, enabling tool implementers to improve its expressiveness. WebAlloy [3] uses *declarative* models and access control restrictions in Alloy to generate web applications. Unfortunately, the semantics of the domain-specific language and the tight coupling with the implementation eliminated implementation overrides, resulting in a J2EE application that disallowed Java code from the developer. All custom application logic was restricted to the jsp pages, violating a central principle in the MVC architecture.

Design-by-Contract. Design by Contract, coined by Bertrand Meyer in the development of the Eiffel language, is adapted for Java projects [4], and received recent attention [5]. Design-by-Contract enables the use of preconditions and postconditions for methods and invariants for classes. Preconditions, postconditions, and invariants are boolean predicates. If the caller of a method satisfies the preconditions, the method implementation guarantees that the postconditions will be satisfied. A class invariant is a predicate conjoined with every public method's postcondition. Our project uses logic explicitly modeled after JContractor to provide runtime guarantees.

Bounded Exhaustive Testing (BET). Unit testing assures robustness for software. Unfortunately, unit testing requires additional developer effort. Not only must the code be written, but tests must be written to exercise the code. BET tools such as [6] automate unit test generation, ensuring code coverage for bounded domain size. Korat generates all *valid* objects within a bound and exhaustively calls public methods with parameters satisfying method preconditions, checking the resulting objects for validity. Validity is determined by executing the class invariant. Our project uses Korat for test case generation.

II. Methods

Web Applications are specified via a UML model (class diagrams) with OCL constraints. We define UML stereotypes for Model, View, Controller, and Page classes. These can be exported via the UML XML Metadata Interchange (XMI). These are parsed and translated into a smaller XML representation that is both used by our tool and developer-friendly (all test cases are manually generated in this format). This XML format is used to generate a model of a web application. The model, called a Project, internally represents Models, Controllers, Views, Routes, and specific resources for deployment on Google AppEngine.

The Project can generate a set of Alloy Analyzer test scripts. These test scripts help check consistency of the OCL specifications, but also can check whether a bounded verification check of a constraint can hold (any OCL constraint can be generated as a $fact$, a $predicate$, or a $check$ in the Alloy scripts). This allows for analysis of the design prior to implementation. However, there are OCL expressions that do not translate easily into Alloy. For simplicity's sake, we translate expressions using propositional logical operators, first order quantifiers, and equality operators. We issue warnings on general iteration, temporary variable assignment, and side-effect expressions.

For all OCL, we generate Java methods which are used as preconditions, postconditions, and invariants in the generated J2EE web application. For the purpose of generating Java code, iteration and temporary variables are not a problem.

Java contracts also are used to reduce the burden of unit testing. Korat test suites are generated for Bounded Exhaustive

Testing. This helps protect the application against regression during implementation but before deployment (the tests run in a local environment).

III. Application Restrictions

Restricting the application domain to Model-View-Controller web applications running on Google AppEngine provides many benefits in terms of the expressiveness of our specifications. Additional semantics are assumed for common field and method naming conventions. Application robustness features are provided by construction due to additional restrictions and assumptions.

To help reduce the burden on the programmer, we provide some high-level modeling constructs that are common in MVC applications. HasMany, BelongsTo, HasAndBelongsToMany automate relationships and their associated implementation fields, the retrieval via queries, and navigation between related objects in the generated user interface.

MVC shrinks analysis scope to manageable size. Models and their relationships are analyzed independently from the rest of the system (similar to [1]). Controllers are analyzed independent of other Controllers with their associated models and their dependencies.

We assume that all Models support CRUD operations. The semantics of the operations are automatically assumed unless overridden. This reduces the size of the specification and simplifies analysis of the design for the common case.

Restricting to AppEngine trivially invalidates SQL injection attacks as a threat (SQL is not supported). An analogous injection attack for JDOQL is automatically protected against by using only the single-token builder API for JDO queries. The builder pattern properly escapes all inputs by only accepting a single token at a time. The builder API is as expressive as the general String version of the API (though neither are relationally complete when running on AppEngine), which is susceptible to injection attacks. Use of the general String API will raise a critical warning prior to deployment (the class files are analyzed for use of unsafe methods).

IV. Brief Sketch of Use Cases

Several applications have been built from specifications. A magic 8-ball, specified in 15 lines of xml, provides random answers to questions. A blogging application tied to Google user accounts was generated from 21 lines of code for the purpose of demonstrating the orphan record problem and its solution at design time. A polling/questionaire application was specified in 38 lines. One was an invariant to enforce that each user could only vote once (line-breaks not in original):

```
Response.allInstances->forAll(r |
  r.facebookUser=self.facebookUser
  implies r=self)
```

The second was a shortcut for application navigation:

```
self.option.question.poll = self.poll
```

That compact specification of 38 lines generates a web application with about 1000 lines of java code for Models and Controllers and another 1000 lines of jsp code for Views. It also generated a 65 line Alloy script for analysis. The resulting application runs on Google AppEngine and will ultimately be run as a Facebook application, changing only the View code. Runtime validation guarantees that no user can vote in a poll more than once, and Korat testing helps ensure correct implementation before deployment.

V. Conclusions

Using a compact specification language as a high-level domain specific language for model-driven development enables the development of more robust web applications with a net reduction in programmer effort compared to simply using a web application framework. Further restricting the deployment environment can provide large gains in productivity at the expense of limiting the applicability of a tool. However, with over 200,000 active applications deployed on AppEngine, this still presents a significant number of users for even a restricted tool.

References

[1] J. Nijjar and T. Bultan, "Bounded Verification of Ruby on Rails Data Models," in *International Symposium on Software Testing and Analysis*, July 2011.

[2] D. Jackson, "Alloy: a lightweight object modelling notation," *ACM Trans. Softw. Eng. Methodol.*, vol. 11, no. 2, pp. 256–290, 2002.

[3] F. Chang, "Generation of policy-rich websites from declarative models," Ph.D. dissertation, MIT, Cambridge, Massachusetts, February 2009.

[4] M. Karaorman and P. Abercrombie, "jContractor: Introducing Design-by-Contract to Java Using Reflective Bytecode Instrumentation," *Formal Methods in System Design*, vol. 27, no. 3, pp. 275–312, November 2005.

[5] "Contracts for Java," http://code.google.com/p/cofoja/.

[6] C. Boyapati, S. Khurshid, and D. Marinov, "Korat: automated testing based on java predicates," in *International Symposium on Software Testing and Analysis*, 2002, pp. 123–133.

EVILSEED: A Guided Approach to Finding Malicious Web Pages

Luca Invernizzi
UC Santa Barbara
Santa Barbara, CA, USA
invernizzi@cs.ucsb.edu

Marco Cova
University of Birmingham
Birmingham, UK
m.cova@cs.bham.ac.uk

Christopher Kruegel
UC Santa Barbara
Santa Barbara, CA, USA
chris@cs.ucsb.edu

Stefano Benvenuti
University of Genova
Genova, Italy
ste.benve86@gmail.com

Paolo Milani Comparetti
Vienna Univ. of Technology
Vienna, Austria
pmilani@seclab.tuwien.ac.at

Giovanni Vigna
UC Santa Barbara
Santa Barbara, CA, USA
vigna@cs.ucsb.edu

The web has become the medium of choice for people to search for information, conduct business, and enjoy entertainment. At the same time, the web has also become the primary platform used by miscreants to attack users. These attacks can either exploit vulnerabilities in the users' web browser, or use social engineering to trick victims into installing malicious software. The web is a very large place, and new pages (both legitimate and malicious) are added at a daunting pace. Attackers relentlessly scan for vulnerable hosts that can be exploited, they create millions of fake pages and connect them into sophisticated, malicious meshes, and they abuse stolen accounts to break into hosting providers to inject malicious code into legitimate web sites. As a result, it is a challenging task to identify malicious pages as they appear on the web, in order to protect web users. For example, one can leverage information about web pages that compromise visitors to create blacklists. Blacklists prevent users from accessing malicious content in the first place, and they have become a popular defense solution that is supported by all major browsers. Moreover, the ability to quickly find malicious pages is necessary for vendors of anti-virus (AV) products who need to timely obtain newly released malware samples to update their signature databases.

Searching for malicious web pages is a three-step process: First, one has to collect pointers to web pages (URLs) that are live on the Internet. This is typically done via web crawlers. The purpose of the second step is to filter these URLs for subsequent, detailed analysis. The number of pages discovered by a crawler might be too large to allow for in-depth analysis. Thus, one requires a fast, but possibly imprecise, prefilter to quickly discard pages that are almost certain to be legitimate. For the third step, we require detection systems that can determine with high accuracy whether a web page is malicious. To this end, researchers have introduced honeyclients. Some of these systems use static and/or dynamic analysis techniques to examine the HTML content of a page as well as its active elements, such as client-side scripting code (typically JavaScript scripts or Java applets). The idea is to look for signs of well-known exploits or anomalous activity associated with attacks [**?**]. Other detection systems look for changes to the persistent state of the operating system once a page has been loaded (such as additional files or processes) [**?**]. These changes often occur as a result of a successful exploit and the subsequent execution of a malware binary.

The resources for identifying malicious pages are neither infinite nor free. Thus, it is essential to perform this search in an efficient way and to optimize the process so that we can find as many malicious pages as possible in a fixed amount of time. In this paper, we propose an approach that improves the efficiency of the first step of the search process. More precisely, we propose a system, called EVILSEED, which complements the (essentially random) web crawling approach with a *guided search* for malicious URLs. EVILSEED starts from a set of known pages that are involved in malicious activities. This set contains malicious pages that were directly set up by cybercriminals to host drive-by download exploits or scam pages. The set also includes legitimate pages that were compromised and, as a result, unknowingly expose users to malicious code or redirect visitors to dedicated attack pages. In the next step, EVILSEED searches the web for pages that share certain similarities with the known malicious pages. Of course, these searches are not guaranteed to return only malicious pages. Thus, it is still necessary to analyze the search results with both prefilters and honeyclients. However, the key advantage of our approach is that a result of our guided search is *much more likely* to be malicious than a web page found by a random crawler. Thus, given a fixed amount of resources, our approach allows us to find more malicious pages, and we do so quicker.

The general architecture of EVILSEED is shown in Figure 1. The core of our system is a set of gadgets. These gadgets consume a feed of web pages that have been previously identified as malicious (as well as other data feeds, such as DNS traffic). Based on their input, the gadgets generate queries to search engines. The results returned by the search engines are then forwarded to an (existing) analysis infrastructure. In the current implementation of EVILSEED, the honeyclient consists of three components: Google's Safe Browsing blacklist,

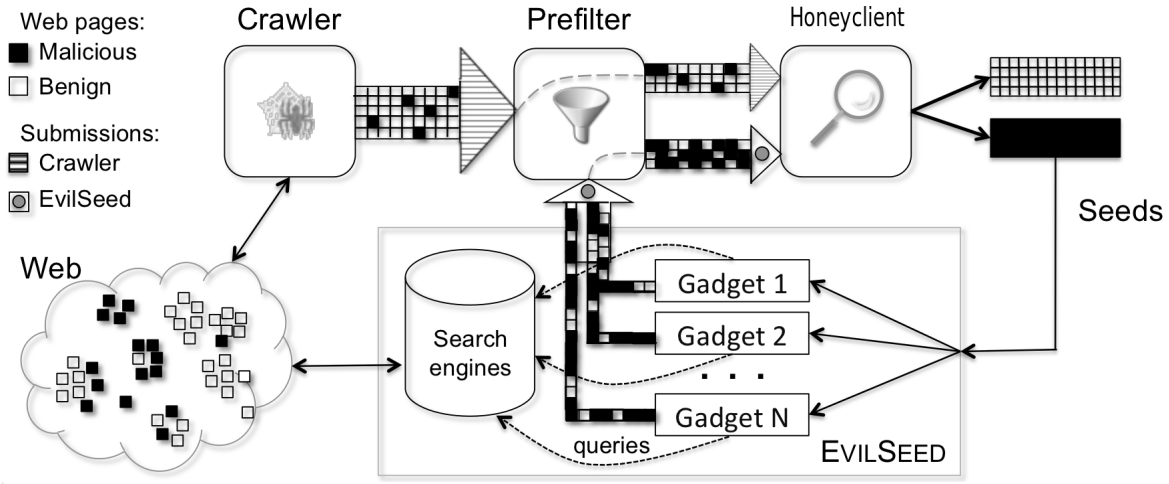


Fig. 1. EVILSEED overview.

Wepawet [**?**], and a custom-built tool to detect sites that host fake AV tools.

We have implemented four gadgets. The *links gadget* leverages the web topology (web graph) to find pages that link to several malicious resources. In our experience, these pages can be grouped in two categories: vulnerable sites that have been infected multiple times (this is typical, for example, of unmaintained web applications), and pages that catalog (and link to) web malware (this is the case of certain malware discussion forums, such as `malwareurl.com`). The *content dorks gadget* aims at discovering vulnerable and exploited web applications that contain the same word n-grams or relevant keywords, based on examples from the seeds. For example, "powered by PhpBB 2.0.15" indicates that a website can be easily compromised, and "calendar about pregnancy" is a query that returns pages that are part of the same injection campaign about pharmaceutical products. The *SEO gadget* identifies pages that belong to blackhat Search Engine Optimization campaigns, crawling the whole campaign. These pages can be identified because they present a different content to search engines spiders, and exploitable browsers. The *DNS queries gadget* analyzes traces of DNS requests to locate pages that lead to a domain serving malicious content. Typically, users reach the malicious domain via a chain of redirections that starts from a legitimate website. By looking at the series of DNS requests coming from an host that visits a known malicious domain, we discover the exploited benign site at the beginning of the chain.

We evaluated our gadgets both offline, using a known set of malicious pages (Table I), and online, in a feedback loop with a live Wepawet instance, using as seed the pages that Wepawet recognized as malicious (Table II, as in Figure 1). Our results show that EVILSEED is able to retrieve a set of candidate web pages that contains a much higher percentage of malicious web pages, when compared to a random crawl of the web. Therefore, by using EVILSEED, it is possible to improve the effectiveness of the malicious page discovery process.

Source	Seed	Visited	Malicious	Toxicity	Expansion
EVILSEED					
Links	1,440	169,928	2,618	1.541%	1.818
SEO	248	12,063	11,384	94.371%	45.903
DNS queries	115	4,820	171	3.548%	1.487
Content Dorks	443	76,254	1,314	1.723%	2.966
Total		263,065	15,487	**5.887%**	
Crawler		791,975	1,094	**0.138%**	
Random Search		24,973	76	**0.303%**	
Google Dorks		4,506	17	**0.377%**	

TABLE I
PERFORMANCE OF INDIVIDUAL GADGETS.

Source	Visited	Malicious	Toxicity
EVILSEED			
Content Dorks			
n-grams selection	42,886	924	2.155%
term extraction	3,977	156	3.923%
Links	246	33	13.415%
EVILSEED **(Total)**	47,109	1,113	**2.363%**
Crawler	433,672	274	**0.063%**

TABLE II
EVILSEED IN ONLINE MODE AND CRAWLING.

The Phantom Anonymity Protocol

Johannes Schlumberger[§], Magnus Bråding and Amir Houmansadr[‡]

[§]University of California, Santa Barbara js@cs.ucsb.edu
magnus.brading@fortego.se
[‡]University of Illinois at Urbana-Champaign ahouman2@uiuc.edu

***Abstract*—To address known issues with today's anonymity networks (such as scalability, performance and ease-of-use), we propose a set of design goals for a new, decentralized and scalable anonymity network that we call the Phantom anonymity protocol. Based on the proposed design goals we developed and implemented a protocol aimed at easy user adoption, low latency and high throughput. Our design is fully IP compatible, decentralized, easy to use and has no single prominent point to be targeted by legal or technical attacks.**

I. Introduction and Motivation

In recent years, there has been a massive upswing in surveillance and censorship laws across many states and nations, following the fear of terrorist attacks or similar threats. To counter these risks to personal freedom and privacy, several anonymous communication networks such as Tor [1] and I2P [2] have been proposed, which enable anonymous communication through the Internet. In particular those services are important for whistleblowers, citizens of states with heavy censorship laws and typical Internet users that aim to keep their activities private. We believe most of the Internet users would choose to protect their privacy, given the chance to do so without taking big performance penalties.

Unfortunately, the existing anonymous networks are subject to different attacks that try to compromise their anonymity promises or availability. Examples of such attacks are traffic analysis [3], [4], [5] or denial of service attempts [6], [7].

Moreover, various lawsuits have been leveraged against people involved in these networks, which shows that a successful anonymous network has to tackle some non-technical issues as well.

Previous solutions either fail at being easily usable for non-technical users or do not provide good throughput. Also due to their complicated usage, they are not widely implemented within the Internet.

We propose a new anonymity protocol, Phantom, that is specifically designed as a protocol to overcome the weaknesses of its predecessors, providing regular Internet users with anonymity, being transparent to use for different applications and being highly scalable.

II. Design Goals

In this section we state the design goals for the Phantom anonymity protocol.

Complete decentralization: A centralized anonymous network is fragile to attacks on its central part and does not scale with the number of users.

Resistance to denial of service (DoS) attacks: Resisting different DoS attacks that try to exhaust its resources is necessary for anonymous networks to serve its users.

Provable anonymity: The security of the anonymity provided by the network should be rigorously proven.

Confidential communication: An anonymous network should guarantee the communication confidentiality of its users by deploying end-to-end encryption.

Isolation from the Internet: To protect the deploying nodes against legal actions, an anonymous network should be isolated from the Internet. However, some nodes can act similar to Tor-exit-nodes to access content on the Internet.

Protection against protocol identification: A protocol must not be easily identifiable from the outside, so no one can easily and selectively drop the protocol traffic.

High throughput capacity: Any network should provide a high throughput to its users.

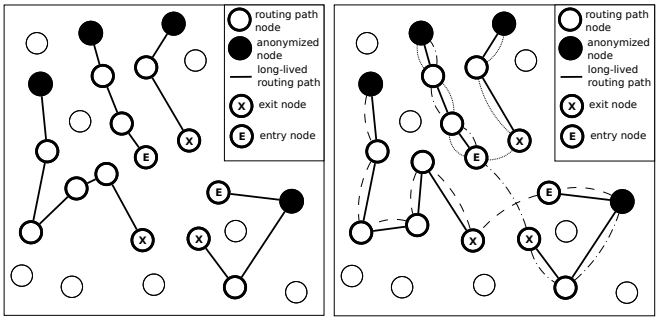
Generic, well-abstracted and backward compatible design: A backward compatible design makes it easier for a new protocol to be used with already existing applications, greatly boosting the acceptance rate and variety of services offered over the protocol.

III. High level sketch of the protocol

Having described our design goals we give an overview of the Phantom protocol abstraction used to achieve these goals. The protocol overview given here is extremely high level, a reader interested in technical detail can find more such on the project web page [8], [9], [10].

The Phantom anonymity network is a completely novel overlay network, where nodes relay traffic for each other in a peer-to-peer manner and store data in a distributed hash table (DHT). Anonymized nodes communicate with each other through proxy nodes forwarding IP-traffic, making the protocol fully transparent to all existing applications.

The Phantom protocol abstraction consists of three logical parts. First, long lived routing paths, which can be seen as a chain of nodes relaying incoming and outgoing traffic to and from an anonymized node, with a designated entry or exit node acting as a proxy for incoming or outgoing requests concerning the anonymized node. Second, shorter lived routing tunnels, which form the logical equivalence of a point to point connection. Third, a DHT, containing various pieces of information, such as public keys of the participants and information about the current entry nodes for a given anonymized node offering a service.



Anonymized nodes with entry and exit routing paths (left) communicating with each other using routing tunnels (right)

A routing path is the central anonymity concept of the Phantom design. It consists of a number of nodes relaying traffic for an anonymized node. The main purpose of a routing path is to decouple an anonymized node from its IP address and replace it by a fully compatible AP (Anonymous Protocol) address. An AP address is just like an IP address in many ways, but the person communicating using this AP cannot be deduced from it.

Routing tunnels, initiated by an anonymized node controlling an exit routing path, are used to communicate between two Phantom nodes. They are logically constructed along routing paths. A finished routing tunnel consists of the nodes forming the entry and the exit routing path of two communication endpoints. Once established, the tunnel can be seen as a bidirectional point-to-point connection between two network participants enabling them to exchange arbitrary data over it. This connection is used to transmit an onion-encrypted IP-datagram stream between the two communication endpoints.

A DHT is used to store all the information needed by the protocol. This information must include, at least, a mapping that makes it possible to get the entry nodes for an AP address and information describing other Phantom nodes, specifically their IP addresses and ports along with some cryptographic keys. By using a DHT instead of a central storage for this information, we eliminate every central point of authority and control in the network and keep it scalable.

If a node wants to participate, he first joins the DHT, retrieves information about other nodes from it and uses these to construct a routing path under his control. He then uses his routing path to receive incoming and establish outgoing connections to IP-based services offered by other Phantom nodes.

IV. Implementation

A prototype implementation of Phantom is publicly available[8], consisting of the full routing path and tunnel logic. It is based on a kademlia[11]-like DHT design and usable on a Linux operating system.

V. Evaluation

To show that our implementation meets the criteria for low latency and high throughput, we ran a set of experiments [9] on a test network. The time to create a routing path of length three is roughly three seconds on middle class machines; these measurements do not take churn into account which.

To test the throughput of a tunnel between two Phantom nodes, we copied several large files via scp. The throughput achieved was very close to the theoretical throughput limit of the underlying 100Mbit network, which shows that the overhead introduced through cryptographic operations is easily bearable on today's machines and the protocol itself adds only very small overhead on the wire.

VI. Issues

The protocol design is as of today a work in progress and such has some unresolved issues, that we are still working on.

A new DHT needs to be designed that avoids information leakage common to all current designs used in other protocols. Depending on the new design of the DHT, defense mechanisms against Sybil, predecessor and DoS attacks need to be carefully chosen.

To evaluate our design further with respect to the effect of churn, usability and resistance against aforementioned attacks, we plan to set up a reasonably sized real world test network to collect data that allows comparison with previous approaches.

VII. Conclusions and Future work

Analyzing existing anonymous networks and the growing need for anonymity in modern societies, we have developed a set of design goals for a new overlay network protocol that is easy to use, decentralized and fast. Based on these goals, we designed a novel anonymity network protocol. By implementing the design, we have shown that it is feasible and our evaluation measurements indicate the high-throughput and low latency goals are met.

Future work should concentrate on improving our implementation and expanding the design to address the unresolved issues discussed above.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13 th Usenix Security Symposium*, 2004.

[2] T. Schweyer, http://www.i2p2.de/.

[3] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy. IEEE CS*, 2005, pp. 183–195.

[4] A. Houmansadr and N. Borisov, "Swirl: A scalable watermark to detect correlated," in *Proceedings of the 18th Annual Network & Distributed System Security Symposium*, 2011.

[5] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems."

[6] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, "Denial of service or denial of security?" in *Proceedings of the 14th ACM conference on Computer and communications security*, ser. CCS '07, 2007, pp. 92–102.

[7] N. S. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on tor using long paths," 2010.

[8] J. Schlumberger and M. Bråding, http://code.google.com/p/phantom/.

[9] J. Schlumberger, "Implementing the phantom protocol," http://wwwcip.cs.fau.de/~spjsschl/da.pdf, 2010.

[10] M. Bråding, "The phantom protocol," http://www.magnusbrading.com/phantom/phantom-design-paper.pdf, 2008.

[11] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 2002, pp. 53–65.

A Framework for Modeling Trust in Collaborative Ontologies

Byungkyu Kang, John O'Donovan, Tobias Höllerer
Department of Computer Science
University of California, Santa Barbara
{bkang, jod, holl}@cs.ucsb.edu

I. Introduction

At the heart of both social and semantic web paradigms is the support for any user to become an information provider. While this has huge benefits in terms of the scope of information available, it raises two important problems: firstly, the well researched problem of information overload, and secondly, the problem of assigning trustworthiness to a piece of information, or an information source. Given the small window of information available for us to make decisions about trust on the web, relative to real-world trust decisions, this becomes a challenging problem. This paper presents a framework for harnessing available information in the domain of collaborative/shared ontologies on the Semantic Web.

In this paper we distinguish between semantic data that is personally created, and data that has been pulled out of public database or other shared resource. Various semantic databases currently exist. some of which are integrated into a single database network, such as DBpedia for example. DBpedia is based on structured information from the Wikipedia dataset, and supports complex relational queries over its contents. As of January 2011, this database contains more than 3.5 million entries, out of which 1.67 million are classified in a consistent ontology [2]. DBpedia is a good representative example of how credibility can play an important role in the Semantic Web. It is inevitable that when anonymous data is derived from a massive data set, errors and inconsistencies, whether malicious or otherwise will begin to manifest. This problem is worsened in cases where there are large amounts of data and provenance is difficult to source. To address this class of problems, this paper describes initial steps towards a framework for modeling trust in shared ontologies. The model incorporates elements of contextual relevance to a query or task, global reputation of a data source and history of previous interactions between the information producer and consumer.

II. Collaborative Ontologies

In general, a web ontology is created by a group of people who share the same notion or idea. Accordingly, we propose a cloud based collaborative ontology which can be freely updated or modified by any client in a network. Our system is based on a voting mechanism and each client has permissions to assert ontological information as well as to provide ratings of previously updated ontological entry into the semantic cloud. Once an assertion has been made into the cloud by any client, this affects all the other users of the system. *Fig 1* provides a high level overview of the system, focusing on its role in the semantic web cloud network. When ontological data is thought of in a collaborative context, it is crucial to consider factors of trust and provenance of shared information prior to integration with local ontologies. As Huang and Fox [5] claim, "borrowed data should be trustworthy in order to yield reliable information using it".

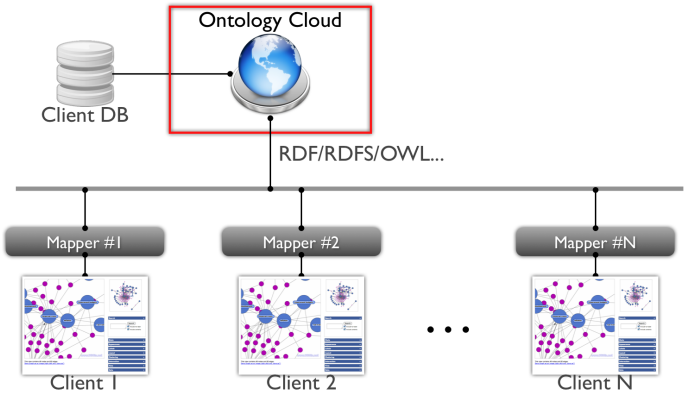


Fig. 1. Collaborative Ontology Network with Multiple Clients

III. Trust, Credibility and Expertise

In this paper, we adopt three major variables–*Trust, Credibility and Expertise*– to show three different reliability metrics. These variables are used in the mathematical model in the following section to calculate what we term the *Assertion Weight Model*. A Trust variable represents an average value of interpersonal credibilities between two clients based on a history of their previous communications. The previous communication can be referred as a directional reference such as *following* or *retweeting* in the Twitter network. For example, client C_i and client C_j may have ranked multiple scores to each other if both clients had asserted more than one ontological information such as triple into the server(cloud). The notation for trust from client C_i to Client C_j can be expressed as in Equation 1 below. *n* in this following equation represents total number of ranks or communications made by client C_i as to the entries created by client C_j.

$$T_{ij} = \frac{\Sigma_n T_{ijn}}{n} \tag{1}$$

Note that T_{ij} are not commutatative since client C_i is a rater and client C_j is ontology information creator due to directional voting mechanism.

While the Trust T_{ij} represents interclient reliability, We view credibility as a more global "reputation-like" concept, representing an overall evaluation of a client based on a composition of individual trust values. This variable can be expressed as follows.

$$Cred_j = \frac{\Sigma_n[\frac{\Sigma_i T_{ijn}}{i}]}{n} \quad (2)$$

To calculate the credibility that client C_j has, we assume that a client C_j has created n ontological entries in the network and each entry has i scores ranked by different i clients.

We define *expertise* as third factor used in our model to assess the weight of an information source. Expertise is related to the contextual relevance of an information source to some target information task, based on analysis of content, for example TFIDF, LDA or other statistical text based comparison metric.

$$E_{jk} = s_{jk} + o_{jk} \quad (3)$$

Note that E_{jk} is the expertise of the $Client_j$(creator) on his/her $k-th$ ontological information. s_{jk} and o_{jk} stand for the number of entries(triplets) this client created before which contains the same *subject* or *object* of selected ontology entry Ont_{jk}.

IV. Assertion Weight Model

As each client of the system ranks relevant scores on the previous ontological entries created by other clients, the system updates an Assertion Weight Model in the ontology cloud. In this section, we explain the Assertion Weight Model. Mathematical notations for the Assertion Weight Model W on the item k is as follows.

$$W_k = f(E_{jk}, T_{ij}, Cred_j) \quad (4)$$

where the weight model funtion $f()$ is,

$$f(E_{jk}, T_{ij}, Cred_j) = \alpha E_{jk} + \beta T_{ij} + \gamma Cred_j \quad (5)$$

$$\alpha + \beta + \gamma = 1 \quad (6)$$

Note that the three coefficients – α, β, γ, respectively – represent contribution weight coefficients which can be defined based on a particular learning or optimization function, a next step on this research agenda.

Our Assertion Weight Model can be explicitly distinguished from previous literatures such as Gil, et al. [3] and Golbeck, et al. [4] since these literatures propose trust semantic network in the scope of social network setting. In other words, they describe trust of the semantic web based on the users or agents. On the contrary, our model describes more as a collaborative tool in cloud computing like environment based on individual ontologies. Another benefit of our framework is that this model can be utilized in any form of expression(RDF, RDFS, OWL, and so on) with different types of database(RDB, TDB, SDB, and so on). Figure 2 shows a visualization of trusted information sources in the collaborative ontology. The visualization was generated using the WiGis toolkit (www.wigis.net).

V. Preliminary Experiments

To evaluate our model, we propose to conduct a preliminary experiment based on integration of data from multiple sources, with a mechanism for eliciting feedback on information from a community of users. Jena is a Java-based open source semantic web toolkit. It supports most ontology languages with several different rule-base reasoners, SPARQL query support and several persistent graph stores. SDB is a component of Jena

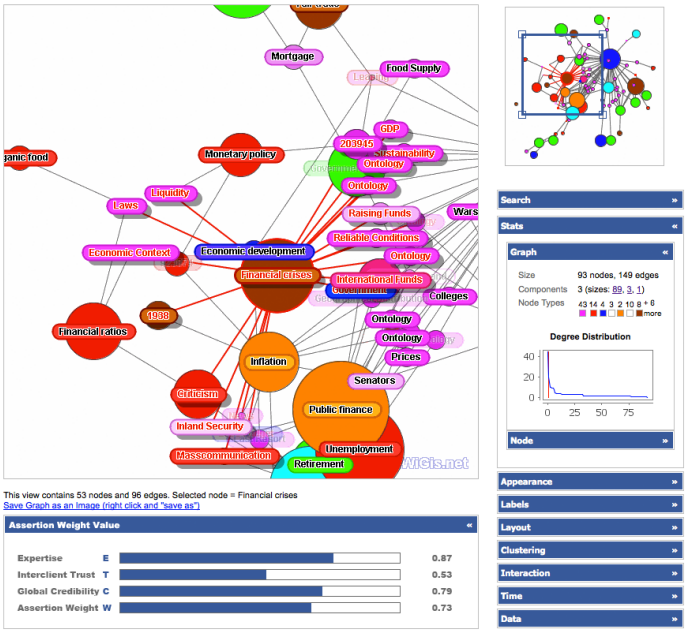


Fig. 2. Visualization Example of Semantic Credibilities using WiGis Framework

toolkit and this unique database has the ability to easily parse SPARQL query between Jena API and semantic database.

In this early stage of research, a specific testbed for evaluation has not been targeted, however, the evaluation will combine facets from different ontologies using the standard JENA triple-store model and they will be accessed through the JENA API, in keeping with W3C standards.

VI. Conclusion

A novel framework for modeling trust in collaborative ontologies was presented in this paper. The model is based on three trust-based weightings for information providers in collaborative ontologies: Global reputation of a source, history of interactions between an information producer and consumer, and contextual relevance of a source to a particular target task. In addition we have proposed a visual interface for eliciting feedback from users by putting them "in the computational loop" through simple interactive visual feedback mechanisms. We believe that as the semantic web expands and becomes more widely used, the need for such reliable mechanisms for computing credibility and trust of the information providers increases therein.

References

[1] Berners-Lee, Tim; James Hendler and Ora Lassila (May 17, 2001). "The Semantic Web". Scientific American Magazine. Retrieved March 26, 2008.

[2] http://wiki.dbpedia.org/Datasets, 2. Content of the DBpedia Data Set

[3] Yolanda Gil, Varun Ratnakar. Feb 25, 2009 - Trusting Information Sources One Citizen at a Time,
International Semantic Web Conference pp.162-176, 2002

[4] Golbeck, Jennifer, Bijan Parsia, James Hendler, "Trust Networks on the Semantic Web," *Proceedings of the Seventh International Workshop on Cooperative Information Agents*, August 2003, Helsinki, Finland.

[5] J. Huang and M. S. Fox. An ontology of trust - formal semantics and transitivity. *In Proceedings of The Eighth International Conference on Electronic Commerce*, pages 259-270. ACM, 2006.

[6] B. Gretarsson, S. Bostandjiev, J. O'Donovan, and T. Höllerer. "WiGis: A Framework for Web-based Interactive Graph Visualizations." (International Symposium on Graph Drawing 2009, Chicago, USA)

SIGMA: A Statistical Interface for Graph Manipulation and Analysis

Greg Meyer, Brynjar Gretarsson, Svetlin Bostandjiev, John O'Donovan, Tobias Höllerer

Department of Computer Science, University of California, Santa Barbara
meyer.greg.pro@gmail.com, {brynjar,alex,jod,holl}@cs.ucsb.edu

***Abstract*—In this paper, we present a statistical approach for gaining deep understanding of a graph visualization. The approach follows Shneiderman's vision that "visualizations simplify the statistical results, facilitating sense-making and discovery of features such as distributions, patterns, trends, gaps and outliers."[3]. Thus, the addition of statistical metrics within a graph visualization tool efficiently improves exploratory data analysis and allow analysts to discover new interesting relationships between entities. In addition, we believe that a statistical interface can play a role as a navigation control for a large graph visualization. This paper presents a discussion of design and implementation of the *SIGMA* statistics visualization module for WiGis [2], and example use cases showing how statistical views help to increase a user's understanding of graph visualizations.**

I. Introduction

With the advancement of rich internet technologies, and explosion of network data available on the web, interactive graphs are becoming more common as a means to explore, navigate and understand network data. However, scalability of these tools remains a weak point for most existing systems. For node-link graphs of more than a few hundred connected entities, most client-side graph visualization tools begin to slow down considerably. Usually network visualization tools use layout or clustering algorithms in order to clarify a chaotic visualization. However, this approach traditionally does not scale well. Now that the need for statistical graph representation has been motivated, we provide a brief discussion of related work and describe the design challenges and implementation choices used in the development of the *SIGMA* statistical interface.

A. Statistics in Graph Visualizations

Graph visualization tools abound, some popular examples include TouchGraph Navigator [5], Tom Sawyer [4] , GraphDice [1], and IBM's ManyEyes. These tools base their principal functionalities either on network visualization or statistical analysis but still do not support visualization of a broad scope of statistical functions in an interactive manner. Our novel statistical viewer and navigator, SIGMA, is implemented as a module for our existing graph visualization toolkit known as WiGis [2]. *SIGMA* is focused on the coupling of statistical views and node-link representations of data. WiGis was chosen as a supporting platform for this work because it has a modular design, making plug-in development easy. Addition of a statistics module supports novel research involving interactive analysis of large scale data, graph comparisons, classification and decomposition.

By enabling statistical analysis and control in a graph visualization tool such WiGis, we aim to provide a user with a rapid overview of data contained in a graph. Moreover, the goal is to highlight important components, nodes, edges or relationships, and anomalies that are not obvious in the traditional network view. Statistics allow for simplification/abstraction of a view, which can then support on-demand dynamic focus based on simple interactions, such as moving, deleting, clustering, zooming or running various algorithms to give the user a deeper understanding of the underlying data.

B. SIGMA Module

There is a huge variety of statistical analysis methods that can support graph analysis. In this initial work, we categorized candidate statistical methods into four groups to better organize the analytical process for the end user as they interact with a graph visualization. A *Global* statistics panel persists in every view, containing statistical metrics that apply to the entire graph. If two nodes are selected, a *Pairwise* panel appears containing metrics related to the selected node pair, such as Dijkstra's shortest path algorithm, for instance. Upon selection of a group of nodes, a *SubGraph* panel, gathering statistics for all selected nodes appears in the view. Similarly, when a single node is selected, a *Node* panel appears, giving statistical metrics for the current node. The following list describes the currently implemented statistics in the module :

1) Global Statistics:

- *Size*: Information about graph size in terms of nodes, edges and content.
- *Components*: Listing of distinct components.
- *Node Types*: List and number of nodes for each different type.
- *Degree Distribution*: Shows an interactive chart of degree distribution for the entire graph.
- *Average Path Length*: Average length over all paths.
- *Average Degree Centrality*: Average degree over all nodes.
- *Average In-Out Degree (only for directed graphs)*: Average number of in and out degree over all nodes.

2) Subgraph Statistics:

- *Average Path Length*: Average path length of the selected nodes.
- *Average Degree Centrality*: Average degree over all selected nodes.
- *Degree Distribution*: Presents a real-time degree distribution chart over all selected nodes.

3) Pairwise Statistics:

- *Shortest Path Distance*: Minimum hops number between two nodes (Shortest Path highlighted on the graph).

4) Node Statistics:

- *Degree*: Node's degree.
- *Neighbors*: List of all neighbors for a selected node.
- *In-Degree (only for directed nodes)*: Number of in-coming edges.
- *Out-Degree (only for directed nodes)*: Number of out-coming edges.

II. Use Cases

Since this work is in an early stage, a full live user evaluation is not yet available. To motivate our approach, two practical use cases from different domains are presented here. Each case describes a concrete example of the statistical

viewer supporting discovery of previously hidden information, not easily detectable though the standard graph visualization interface.

A. NSF Dataset

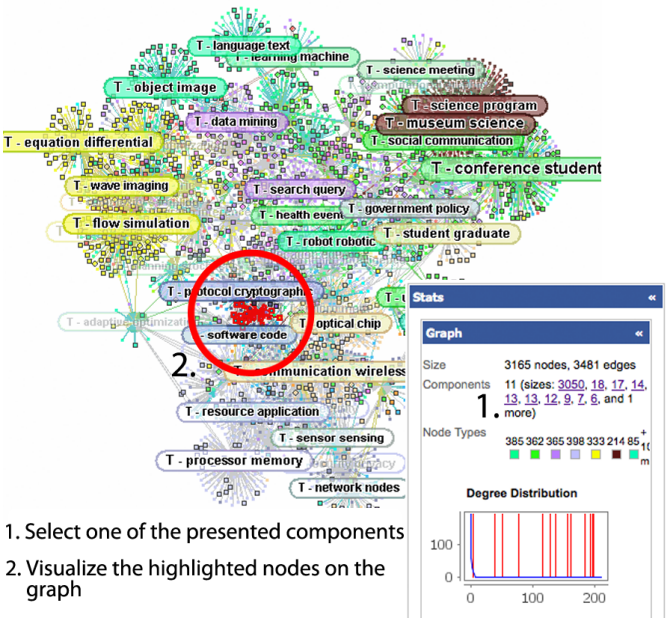


Fig. 1. Use of *Graph* statistics panel to discover isolated graph components

Figure 1 shows a visualization of a collection of awarded NSF grant proposals, showing the documents and topics that they relate to, based on Latent Dirichlet Analysis (LDA) modeling over their contents. The initial view shows the full graph. By clicking on elements in the *Graph* statistics panel, a user can quickly distinguish separate components that were previously hidden. With this feature, it is also possible to click on a disconnected component to highlight each of that node's neighbors, and reveal its graph position. When this occurs, statistical metrics such as the degree distribution chart shown in Figure 1 are automatically recomputed.

B. New York Times Dataset

Figure 2 shows a similar network of documents and topics from LDA analysis, in this case, a collection of New York Times news articles. A user selects a pair of nodes at random and the *Pairwise* panel appears, showing algorithms related to pairwise analysis. The user selects a shortest path algorithm, and in the main graph view, the set of shortest paths between the two selected nodes are highlighted in red, as illustrated in Figure 2. Figure 2 a.) and b.) shows a comparison between two different interaction methods for finding the shortest path between two nodes. In view a.), a user has selected the node pair and dragged them to the right of the screen. An *interpolation-based layout* moves all of the other nodes by a relative amount in the same direction, based on their graph distance from the moved node. The result highlights the shortest path on the right side of Figure 2 a.). Note that this view is specific to the two target nodes only, and all other nodes become clustered based on their graph distance from

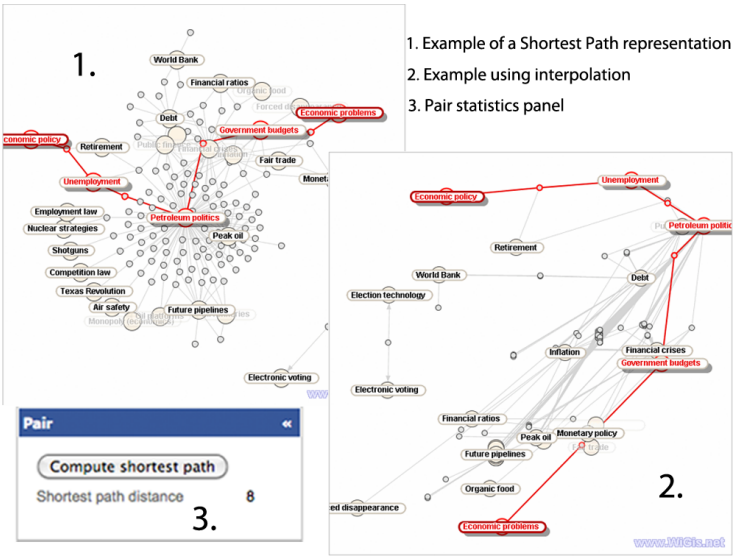


Fig. 2. Two different representations of a shortest path algorithm a.) WiGis interpolation based view (pair-specific). b.) force-directed in a global view using SIGMA.

the selected node pair. Contrastingly, the view in 2 b.) shows a global *force-directed* layout of the graph, in which the shortest path is highlighted between two nodes. This view was arrived at simply by clicking on the button provided in the pairwise statistics panel that appeared on selection of the two nodes. Both views support shortest path analysis, but enable discovery of very different information about the graph.

III. Conclusion

In this research abstract, we have discussed initial work on *SIGMA*, a statistical analysis tool for the WiGis visualization framework. Design details and two use cases on diverse data sets have been presented. Application of statistical analysis and navigation mechanisms to graph visualization tool such as *WiGis* improves a users understanding of the underlying graph. The *SIGMA* module allows a user to focus in on data that may have otherwise remained hidden in a traditional visualization such as force-directed node-link layout. In addition, the visualization tool improves and simplifies the comprehension of statistical metrics by allowing a user to see the results of a statistical method appear on the graph.

References

[1] Anastasia Bezerianos, Fanny Chevalier, Pierre Dragicevic, Niklas Elmqvist, and Jean-Daniel Fekete. Graphdice: A system for exploring multivariate social networks. *Computer Graphics Forum (Proc. EuroVis 2010)*, 29(3):863–872, 2010.

[2] Brynjar Gretarsson, Svetlin Bostandjiev, John ODonovan, and Tobias Höllerer. Wigis: A framework for scalable web-based interactive graph visualizations. In David Eppstein and Emden Gansner, editors, *Graph Drawing*, volume 5849 of *Lecture Notes in Computer Science*, pages 119–134. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-11805-0-13.

[3] Adam Perer and Ben Shneiderman. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 265–274, New York, NY, USA, 2008. ACM.

[4] Tom Sawyer Software. Tom Sawyer Visualization, 2009. Available at www.tomsawyer.com.

[5] Touchgraph. Touchgraph llc, http://www.touchgraph.com, 2004.

Detecting Service Outages via Social Media Analysis

Eriq Augustine, Cailin Cushing, Kim Paterson, Matt Tognetti, Alex Dekhtyar Department of Computer Science
Cal Poly, San Luis Obispo
{eaugusti, ccushing, klpaters, mtognett, dekhtyar}@calpoly.edu

***Abstract*—We consider the problem of anomaly detection related to specific online services via analysis of social media (Twitter) messages related to the service provider. Results of the initial study are documented here.**

***Keywords*-anomaly detection, data mining, classification**

I. Introduction

It is frustrating for product owners and online service providers who offer a product or service but sometimes do not receive much feedback from their customers. The customers may, in fact, be providing feedback, but the feedback is frequently transmitted in a way that is not useful: it is sent to the wrong receiver or to a receiver that is not looking for it. Netflix recognizes that its customers are providing feedback about their online streaming applications on social media sites, but currently employs no sophisticated system for processing that feedback. After Netflix sends a file to the customer, they have limited capabilities for detecting if the users are having problems receiving or viewing it. As a result, if a movie file is corrupted or one of their channels is down, it may take some time for Netflix to realize there is a problem. The time delay between an error occurring and Netflix's recognition and response negatively affects their customers perception of the company as a reliable service provide. Consequently, the faster Netflix becomes aware of a problem, the more quickly they can resolve it and maintain their reputation. To ensure error detection happens quickly, Netflix needs to employ a reliable listener than can process the feedback already available on the social networking site Twitter.

Our work concentrates on tapping such feedback for the purpose of detecting, in real time, the situations when Netflix experiences service problems.

We decided to use Twitter as our primary source of user feedback because Twitter is one of the more public social networking sites, allowing almost everyone to view Twitter posts or tweets. tweets are generated quickly and in a high volume, providing us with a wealth of data to process. This paper documents our initial progress in processing tweets for anomaly detection.

II. Contributions

We are studying whether it is possible to detect Netflix service outages by analyzing Twitter traffic related to the term '`Netflix`.' Anecdotal evidence (observations of Twitter traffic during some such outages) suggests that some Netflix customers report the problems they experience on Twitter when streaming Netflix movies and shows. In our initial study, we asked whether we could detect situations when the quantity of such reports would significantly increase.

Tweets are set apart from traditional Internet documents like articles and formal movie reviews because they are limited to 140 characters for English tweets. This means that each document is very short compared to articles and other documents typically used in natural language processing, text classification or information retrieval. Tweets also tend to use informal language, containing deliberate typos, slang, SMS-style abbreviations, and emoticons that carry significant meaning. Informal language can contribute to greater ambiguity in the meaning of phrases. Our challenge is to detect outage reports under such conditions. In our initial study, described here, we look at the ability of traditional text classification techniques to do so.

III. Experiment

We collected every tweet mentioning the word '`Netflix`' from January 1, 2011 to June 1, 2011. For each tweet, in addition to its text, we store the metadata provided by the Twitter API, including the time it was posted. From these messages, we randomly selected 800 to form a training set of tweets.

By observing Netflix-related Twitter traffic, we determined that there were, in general, two major reasons for significant increases in the number of Netflix-related posts: a breaking news story related to Netflix and a Netflix service outage. In addition to these two types of traffic, Netflix-related Twitter traffic included posts that specified which movies the posters watched using Netflix, expressed joy/gratitude to Netflix, or commented on someone else's Netflix-related post. As such, we chose to use three broad categories for Netflix-related tweets:

- Media: news story-related tweets;
- Bad: service outage reports and general complaints;
- Other: regular Netflix-related chatter.

Each of the tweets from our training set was classified manually into one of these three categories. We used the training set to train a number of classifiers as described below.

We used three different ways of processing individual tweets prior to submitting them to classifiers:

- No Filtering – the raw text was sent into the classifier

- Simple Filtering – stop words, non-English characters, and punctuation were removed, and links were replaced with a placeholder
- Full Filtering – simple filtering enhanced but with movie/show title detection and replacement placeholders.

We used various off-the-shelf classifiers from the WEKA machine learning package[**?**]. In this paper, we report on the use of the Naïve Bayes, Bayes Net and J48 classifiers[**?**].

In addition to those individual classifiers, we used a "committee" classifier. The committee classifier used Simple-Filtered and Full-Filtered version of Naive Bayes, Bayes Net, and K-Nearest-Neighbors[**?**] classifiers. The Committe classifier then chose the plurality class.

IV. Results

Netflix provided us with the list of days and times between January 1, 2011 and June 1, 2011, during which it experienced various service outages related to video streaming for different platforms, or web site availability.

In order to determine the accuracy of our classifiers, we tagged any period of significant (beyond 3 standard deviations from the mean) increase in Bad tweets as an indicator of a Netflix service outage. We then compared our estimations to the log of known outage times. Figure 1 depicts tweet volumes during a known outage time. The green line is the volume of all Netflix-related tweets. The red line is the number of tweets in the "Bad" group as classified by the Committee classifier.

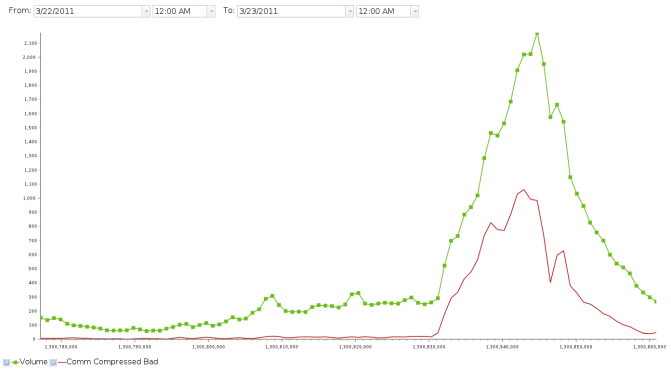


Fig. 1. Outage Period

Individually, our classifiers performed moderately well, detecting about 57% of the known outages. The Committee classifier, however, produced excellent results; correctly catching nearly 86% of the outages (Table I).

Classifier	Percent Outages Caught
Bayes Net	57.14
Naive Bayes	57.14
J48	57.14
Committee	85.71

TABLE I
Outage Detection Results

In addition, the Committee classifier produced very few false positives for Bad tweets, improving the accuracy and confidence of its outage detection. Table II shows a confusion matrix of the Committee classifier when ran on the training set. "Bad" false positives are shown in red.

	Media	Bad	Other
Media	97	0	6
Bad	2	187	111
Other	17	16	393

TABLE II
Committee Classification Results

V. Conclusions

Our best classifier was able to obtain 85% accuracy. Another experiment on classification of social media posts was done by Sarah Schrauwen on the Dutch social networking website, Netlog. Schrauwen was only able to obtain 65% accuracy when trying to classify the valence (mood) of social media posts[**?**]. Schrauwen dealt with data sets that had similar restrictions and informal language. We believe that our work is a good starting point.

VI. Future Work

In the future, we plan to incorporate methods that look at the "valence" of words in the tweet. We use valence to mean a numeric measure of the happiness of a word.

Filtering out the titles of movies and shows will also improve results since movie and show titles like "Death at a funeral" or "System Crash" can easily throw off our classifiers.

We also intend to collect information about authors and use it to weight their posts and improve accuracy.

References

[1] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *SIGKDD Explor. Newsl. 11*, 1 (2009), 10–18.

[2] Schrauwen, S. Machine learning approaches to sentiment analysis using the dutch netlog corpus. In *Machine Learning Approaches to Sentiment Analysis Using the Dutch Netlog Corpus* (Antwerp, Belgium, 2010), CLiPS Technical Report Series, Computational Linguistics & Psycholinguistics.

Chronology-Sensitive Hierarchical Clustering of Pyrosequenced DNA Samples of E. coli

Aldrin Montana, Alex Dekhtyar
Computer Science Department
{amontana, dekhtyar}@calpoly.edu
Emily Neal, Michael Black, Chris Kitts
Biology Department
{erusch, ckitts, mblack}@calpoly.edu
California Polytechnic State University
San Luis Obispo, United States

***Abstract*—Hierarchical clustering is used in computational biology to compare sequenced bacterial strain DNA and to determine bacterial isolates that belong to the same strain. However, at times, the results of the hierarchical clustering are difficult to read and interpret. This paper is a case study for the use of an improved hierarchical clustering algorithm, which takes into account the underlying structure of the bacterial DNA isolate collection to which it is applied.**

***Keywords*-bioinformatics, clustering; pyrosequence; primer; pyrogram;**

I. Introduction

E. coli is a common member of the mammalian and avian intestinal microbiota, and is frequently used as an indicator for fecal contamination in watersheds, lakes, beaches, and recreational water. Because dangerous interspecific pathogens can be transferred through contaminated water, it is necessary for health agencies and environmental protection to be able to track the source of a fecal contamination at the species level. The general process linking microbes (in this case *E. coli*) to a host source is called microbial source tracking (MST).

Our lab is currently developing a cost-effective and efficient MST method to create DNA fingerprints for different strains of *E. coli*. In a pilot study, this method was used to investigate the variation in *E. coli* by utilizing sequence differences in the 23S rRNA - 5S rRNA Intergenic Transcribed Spacer (ITS) region to distinguish between *E. coli* strains. This region is non-coding and is assumed to accumulate more mutations than regions of coding DNA. Assuming that ITS sequences vary for different strains of *E. coli*, generated patterns act as a DNA fingerprint (or pyroprint) for each strain. In this paper we report on the use of traditional hierarchical clustering method (primer5) and an improved method, sensitive to the structure of the data collection under study to determine and track *E. coli* strains found in a human host over a period of time.

A. Pilot Study

Over a period of 14 days a fecal sample and two anal swabs, one immediately following the collection of a fecal sample, and one a few hours later, were collected from a single human host (except on day 7). Up to four bacterial isolates per sample were drawn, and bacterial cultures were grown from each isolate for up to 12 bacterial isolates per day extracted from the host.

B. Data Description

For each isolate, a mix of the 23S rRNA - 5S rRNA[1] was extracted and amplified using the traditional PCR process. The intragenic region was then *pyrosequenced*[1] and the obtained *pyrograms* were compared to each other.

For the purposes of this paper, a *pyrogram* is a sequence of real numbers representing light intensities emitted during the pyrosequencing process when a reagent for a specific nucleotide (A, T, C or G) is introduced in the pyrosequencing reaction. In our experiments, pyrograms of length 104 were generated following the same dispensation sequence of nucleotides for each bacterial isolate. Pearson correlation coefficient was used as the similarity measure between pyrograms. The goal of the study was to determine which bacterial isolates belong to the same strain.

C. Primer5

Primer5 is a program that is capable of analyzing data in many ways[2]. Primer5 is commonly used by biologists and utilizes hierarchical clustering. Applied to the *E. coli* data collected, Primer5 produced the output shown in Figure 1. Hierarchical clustering works by iteratively combining clusters until there is one cluster remaining[3]. This approach discounts the context of the pilot study where the collected isolates were organized by day in chronological order, and by the type of sample within a single day. The output of Primer5 in such situations is hard to read and organize.

II. Chronology-Sensitive Clustering

Our approach clusters the results of the study in a way that takes into account the order in which the isolates were collected. Intuitively, we want to change the order in which a hierarchical clustering algorithm combines clusters. In our

[1]These two genes and their intragenic region are found in the *E. coli* genome in septuplicate.

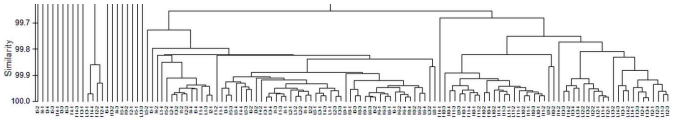


Fig. 1. Primer5 Clustering Results

Fig. 2. Cluster 1

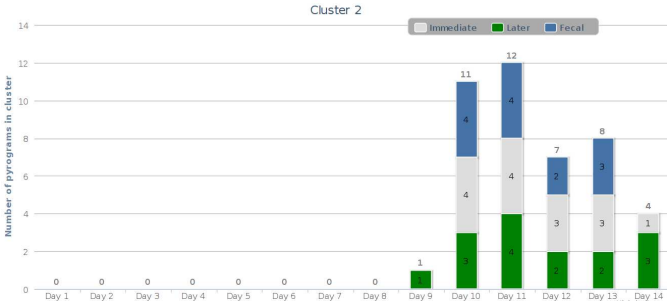


Fig. 3. Cluster 2

algorithm, clusters are first formed out of isolates collected on the same day (and further subdivided by collection method), and then are grown chronologically across days. Our algorithm takes as input two parameters $\alpha > \beta$ representing similarity thresholds. Pyrogram similarity scores above the value α are replaced with 1 and are considered to be the same. Similarity scores below β indicate that the two pyrograms are definitely dissimilar; such scores are replaced with 0. The similarity score transformation is performed before hierarchical clustering commences.

A. Algorithm

Our clustering algorithm works as follows. Isolates collected in the same day are clustered together, then single-day clusters are combined in chronological order. When clustering isolates in the same day, isolates are first clustered for each collection method, then are combined across collection methods. For our algorithm we use average link inter-cluster distance.

III. Results

Results of our study are shown in Figures 2, 3, 4. We used threshold values of $\alpha = 0.997$ and $\beta = 0.95$ provided to the computer scientists by the biologists in the group. As seen from the results, three key clusters were detected spanning across multiple days; additionally, a significant number of

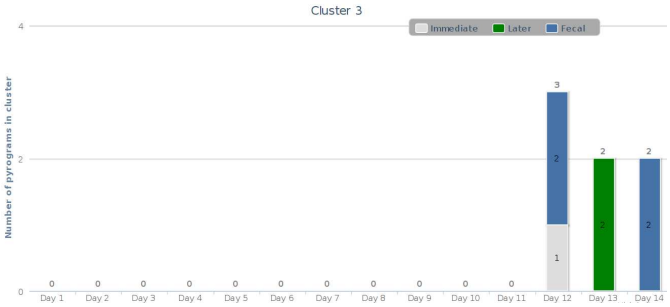


Fig. 4. Cluster 3

individual isolates (not depicted) on different days appeared to represent completely separate *E. Coli* strains.

The largest cluster, Cluster 1, pictured in Figure 2 represents a dominant strain that is seen on days 1 - 9. A second dominant strain, pictured in Figure 3, appears around the time when the host experienced uneasiness in the stomach. The host's ailment was reportedly most uncomfortable around days 9, 10, and 11. The presence and time at which these first two dominant strains are seen seems to correspond to the host's experience. Figure 4 may indicate a third dominant strain becoming established. However, as this cluster is seen in the last three days of the experiment there is not enough data to suggest anything conclusive. Additionally, a total of 17 pyroprints formed their own single-pyroprint clusters and an additional two pyroprints formed another cluster.

IV. Conclusions and Future Work

The proposed algorithm produces a more intuitive hierarchical cluster organization for biological data collections possessing a distinct internal structure. This paper describes briefly our pilot case study for the algorithm, which provided necessary insight into the data for the biologists. The conceived algorithm is applicable to any data collection with internal structure, but the current implementation is specific to the pilot study described in the paper. Work is underway on the development of a general version, and on applying it to more biological studies.

References

[1] R. M., "Pyrosequencing sheds light on dna sequencing," *Genome Research*, vol. 11, pp. 3 – 11, january 2001.

[2] K. Clarke, "Non-parametric multivariate analyses of changes in community structure," *Australian Journal of Ecology*, vol. 18, pp. 117 – 143, 1993.

[3] B. Liu, *Web Data Mining-Exploring Hyperlinks, Contents, and Usage Data*. Springer, 2006.

Closest Pair and the Post Office Problem for Stochastic Points

Pegah Kamousi Timothy M. Chan Subhash Suri

***Abstract*—Given a (master) set M of n points in d-dimensional Euclidean space, consider drawing a random subset that includes each point $m_i \in M$ with an independent probability p_i. How difficult is it to compute elementary statistics about the closest pair of points in such a subset? We obtain hardness results and approximation algorithms for stochastic problems of this kind.**

I. Introduction

Many years ago, Knuth [2] posed the now classic *post-office* problem, namely, given a set of points in the plane, find the one closest to a query point q. This fundamental which arises as a basic building block of numerous computational geometry algorithms and data structures [1], is reasonably well-understood in small dimensions. In this paper, we consider a *stochastic* version of the problem in which each post office may be *closed* with certain probability. A given set of points M in d dimensions includes the locations of all the post offices but on a typical day each post office $m_i \in M$ is only open with an independent probability p_i. Therefore, given a query points q, we ask for the *expected* distance from q to its closest neighbor in M. Similarly we ask: how *likely* is it that the closest pair of points are no more than ℓ apart? In this paper, we study the complexity of such elementary proximity problems and establish upper and lower bounds.

II. The Stochastic Closest Pair Problem

The stochastic closest pair problem asks for the probability that the closest pair has distance at most a given bound ℓ. We show that this basic problem is intractable, via reduction from the problem of *counting vertex covers* in planar unit disk graphs (UDGs). In order to show that even the *bichromatic* closest pair problem is hard, we also prove that a corresponding vertex cover counting problem is hard for a bichromatic version of the unit disk graphs.

A. Counting Vertex Covers in Unit Disk Graphs

We first prove that the minimum vertex cover problem is hard for planar unit disk graphs of maximum degree 3 using which we then prove that counting the vertex covers is also hard for 3-planar UDGs.

Lemma 2.1: The minimum vertex cover problem is NP-hard for planar unit disk graphs of maximum degree 3.

We define the class of *rectilinear unit disk graph* as the unit disk graph with maximum degree 3, which can be embedded in the plane such that the length of each edge is $\geq 2/3$, and the edges lie on the integer grid lines.

We have the following corollary.

Corollary 2.2: The minimum vertex cover problem is NP-hard for rectilinear unit disk graphs.

Theorem 2.3: It is NP-hard to count the vertex covers in a rectilinear unit disk graph. Moreover, the number of vertex covers cannot be approximated to any multiplicative factor in polynomial time assuming $\mathrm{P} \neq \mathrm{NP}$.

Proof: We will prove the inapproximability, which shows the hardness as well. Let $G = (V, E)$ be a rectilinear UDG. Suppose we have an α-approximation algorithm for counting the vertex covers in G, i.e., if $c(G)$ is the number of vertex covers, the algorithm outputs a value $\tilde{c}$ such that $(1/\alpha)c(G) \leq \tilde{c} \leq \alpha c(G)$.

Let G_p be the stochastic graph obtained from G by assigning the probability $p = 1/(2^n\alpha^2)$ of being present to each of the nodes in G. Since this probability is the same for all the nodes, an α-approximation algorithm for counting the vertex covers in G readily provides an α-approximation algorithm for computing the probability that a random subset of nodes in G_p is a vertex cover. Let $\Pr(G_p)$ denote this probability, and $\tilde{r}$ be an α-approximation to $\Pr(G_p)$.

The key observation is that $\tilde{r} \geq p^k$ if and only if G has a vertex cover of size k or less. To see this, suppose G has a vertex cover C of size k or less. Then the probability that a random subset of nodes of G_p is a vertex cover is at least p^k, i.e., the probability that all the nodes in C are present. In this case, $\tilde{r} \geq p^k/\alpha$. Otherwise, at least $k+1$ nodes must be present to constitute a vertex cover, which happens with probability at most $2^{|V|}p^{k+1} < p^k/\alpha^2$. In this case $\tilde{r} < p^k/\alpha$.

Corollary 2.2, however, shows that the minimum vertex cover problem is hard for G, and therefore $\Pr(G_p)$ cannot be approximated to any factor α in polynomial time assuming $\mathrm{P} \neq \mathrm{NP}$. This completes the proof. ■

B. Bichromatic Unit Disk Graphs

We introduce the class of *bichromatic unit disk graphs* as the graphs defined over a set of points in the plane, each colored as blue or red, with an edge between a red and a blue pair if and only if their distance is ≤ 1. We will show that counting the vertex covers is NP-hard for bichromatic UDGs. Consider the gadget $\mathcal{H}$ in Fig. 1 (a), which consists of l paths between u and v, for a given l. Let $G = (V, E)$ be an instance of a rectilinear UDG. Let $G' = (V', E')$ be the graph obtained from G by replacing each edge $uv \in E$ with the graph $\mathcal{H}$. We color u, v and the b_i's red, and the remaining nodes blue.

Lemma 2.4: The graph G' is a bichromatic unit disk graph.

Finally we arrive at the following theorem.

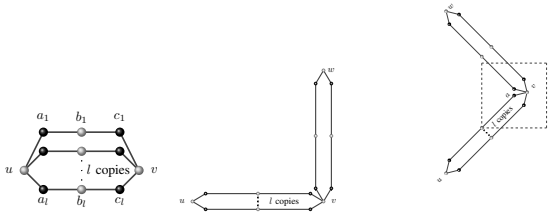


Fig. 1. (a) The gadget $\mathcal{H}$ (b) Two orthogonal gadgets (c) Two rotated gadgets.

Theorem 2.5: It is NP-hard to count the number of vertex covers in a bichromatic unit disk graph even if the distances are measured in the L_∞ metric.

C. Complexity of the Stochastic Closest Pair

In a unit disk graph $G = (V, E)$ defined over the full set M of points, a random subset S of nodes is a vertex cover if and only if in the complement of that subset, no two nodes are at distance ≤ 1. (In other words, all the edges are covered by S.) Therefore, computing the probability that a random subset of nodes is a vertex cover in G amounts to computing the probability that the closest pair of present points in a random subset S are at distance > 1. But as discussed in Theorem 2.3, counting the vertex covers in a unit disk graph is NP-hard. Therefore we have the following theorem.

Theorem 2.6: Given a set M of points in the plane, where each point $m_i \in M$ is present with probability p_i, it is NP-hard to compute the probability that the L_2 or L_∞ distance between the closest pair is $\leq \ell$.

The next theorem considers the bichromatic version of this problem.

Theorem 2.7: Given a set R of red and a set B of blue points in the plane, where each point is only present with an independent, rational probability, it is NP-hard to compute the probability that the closest L_2 or L_∞ distance between a bichromatic pair of present points is less that a given value ℓ.

III. Linearly Separable Point Sets under the L_∞ Norm

We also show that when the red points are linearly separable from the blue points by a vertical or a horizontal line, the stochastic bichromatic closest pair problem under L_∞ distances can be solved in polynomial time using dynamic programming.

Theorem 3.1: The stochastic bichromatic closest pair problem under L_∞ norm can be solved in polynomial time when the read and blue point are linearly separable.

Next Theorem considers the problem for $d > 2$.

Theorem 3.2: Given a set R of red and a set B of blue points in a Euclidean space of dimension $d > 2$, each being present with an independent probability, it is NP-hard to compute the probability that the L_∞ distance between the closest pair of bichromatic points is less than a given value r, even when the two sets are linearly separable by a hyperplane orthogonal to some axis.

IV. Stochastic Approximate Nearest Neighbor Queries

Given a stochastic set M of points in a d-dimensional Euclidean space, and a query point q, what is the expected (L_2) distance of q to the closest present point of M? In this section we target this problem, and design a data structure for approximating the expected value of $d(S, q) = \min_{p \in S} d(p, q)$ with respect to a random subset S of M, assuming that d is a constant. We obtain a linear-space data structure with $O(\log n)$ query time.

A. Approximation via a modified distance function

Assume that the points lie in the universe $\{0, \ldots, 2^w - 1\}^d$. Fix an odd integer $k = \Theta(1/\varepsilon)$. Shift all points in M by the vector $(j2^w/k, j2^w/k, \ldots, j2^w/k)$ for a randomly chosen $j \in \{0, \ldots, k-1\}$.

Given points p and q, let $\mathcal{D}(p, q)$ be the side length of the smallest quadtree box containing p and q. Let $B_s(p)$ be the quadtree box of side length $\lfloor\!\lfloor s \rfloor\!\rfloor$ containing p, where $\lfloor\!\lfloor s \rfloor\!\rfloor$ denotes the largest power of 2 smaller than s. Let $c_s(p)$ denote the center of $B_s(p)$. Let $[X]$ be 1 if X is true, and 0 otherwise.

Definition 1: (a) Define $\ell(p, q) = d(B_s(p), B_s(q)) + 2\sqrt{d}s$ with $s = \varepsilon^2 \mathcal{D}(p, q)$. Let $\ell(S, q) = \min_{p \in S} \ell(p, q)$.

(b) r is said to be *q-good* if the ball centered at $c_{\varepsilon^2 r}(q)$ of radius $2r$ is contained in $B_{12kr}(q)$.

(c) Define $\widetilde{\ell}(S, q) = [\ell(S, q) \text{ is } q\text{-good}] \cdot \ell(S, q)$.

Lemma 4.1: (a) $\ell(S, q) \geq d(S, q)$. Furthermore, if $\ell(S, q)$ is q-good, then $\ell(S, q) \leq (1 + O(\varepsilon))d(S, q)$.

(b) $\ell(S, q)$ is q-good for all but at most d choices of the random index j.

(c) $\widetilde{\ell}(S, q) \leq (1 + O(\varepsilon))d(S, q)$ always, and $\mathbb{E}_j[\widetilde{\ell}(S, q)] \geq (1 - O(\varepsilon))d(S, q)$.

By (c), $\mathbb{E}_j[\mathbb{E}_S[\widetilde{\ell}(S, q)]]$ approximates $\mathbb{E}_S[d(S, q)]$ to within factor $1 \pm O(\varepsilon)$. It suffices to give an exact algorithm for computing $\mathbb{E}_S[\widetilde{\ell}(S, q)]$ for a query point q for a fixed j; we can then return the average, over all k choices of j.

B. The data structure: a BBD tree

We use a version of Arya et al.'s balanced box decomposition (BBD) tree. We form a binary tree T of height $O(\log n)$, where each node stores a cell, the root's cell is the entire universe, a node's cell is equal to the disjoint union of the two children's cells, and each leaf's cell contains $\Theta(1)$ points of M. Every cell B is a difference of a quadtree box (the *outer box*) and a union of $O(1)$ quadtree boxes (the *holes*). Such a tree can be constructed by forming the compressed quadtree and repeatedly taking centroids, as described by Arya et al. The total space is $O(n/\varepsilon^{O(1)})$. The main algorithm is excluded frim this abstract.

V. Conclusion

We show that even elementary proximity problems become hard under the stochastic model, and point to a need for new techniques to achieve approximation bounds. We believe that our results can serve as building blocks for a theory of geometric computation under a stochastic model of input.

References

[1] M. De Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational geometry: algorithms and applications*. Springer, 2008.

[2] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

http://gswc.cs.ucsb.edu